UNIVERSIDADE FEDERAL DE CAMPINA GRANDE - UFCG
CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO - DSC
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA - COPIN

MEDIWEB: UM INTEGRADOR SEMÂNTICO DE DADOS NA WEB BASEADO EM MEDIADOR

Ladjane Silva de Arruda

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba – Campus II, como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Banco de Dados

Cláudio de S. Baptista
(Orientador)

CAMPINA GRANDE, AGOSTO DE 2002

MEDIWEB: UM INTEGRADOR SEMÂNTICO DE DADOS NA WEB BASEADO EM MEDIADOR

Ladjane Silva de Arruda

Dissertação aprovada em 16/08/2002

Cláudio de Souza Baptista, Ph.D

Orientador

Ulrich Schiel, Ph.D

Componente da Banca

Fernando da Fonseca de Souza, Ph.D

Componente da Banca

Campina Grande, 16/08/2002

Ficha catalográfica

ARRUDA, Ladjane Silva de

A773M

MediWeb: Um Integrador Semântico de Dados na Web Baseado em Mediador

Dissertação (Mestrado) - Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande, Paraíba, Agosto de 2002.

103 p. Il.

Orientador: Cláudio de S. Baptista, Ph.D

Palavras Chaves:

1. Banco de Dados
2. Integração de Dados
3. Web
4. Mediadores
5. Web Semântica

CDU – 681.3.07B

"MediWeb: UM INTEGRATOR SEMÂNTICO DE DADOS NA WEB"

LADJANE SILVA DE ARRUDA

DISSERTAÇÃO APROVADA EM 16.08.2002

PROF. CLAUDIO DE SOUZA BAPTISTA, Ph.D
Orientador

PROF. ULRICH SCHIEL , Dr.
Examinador

PROF. FERNANDO DA FONSECA DE SOUZA, Ph.D
Examinador

CAMPINA GRANDE – PB

"A coisa mais bela que o
homem pode experimentar é o
mistério. É essa emoção
fundamental que está na raíz de
toda ciência e toda arte."

(Albert Einstein)

Dedico este trabalho aos meus pais, César Celestino de Arruda e Maria das Neves S. Arruda; e à minha avó Maria Xavier.

Agradecimentos

Mais um degrau alcançado... Como tantos outros, contei com a ajuda, direta ou indiretamente, de pessoas especiais. Gostaria de citar cada uma delas e mostrar meus sinceros agradecimentos. Entretanto, por ser inviável, agradeço a todas de coração.

Agradecimentos mais que especiais a meu pai César Arruda e a minha mãe Nevinha, que me ensinaram o real significado de amor, perseverança e dignidade e que me deram um apoio inestimável.

Agradeço a meus irmãos, Oldair e Odailton (Toninho), pelos conselhos e principalmente pelo exemplo de responsabilidade e dedicação.

Sinceros agradecimentos a meu orientador Cláudio Baptista, pelo empenho, paciência, grande incentivo e acima de tudo, pela amizade e confiança.

Agradecimentos ao pessoal do Laboratório de Sistemas de Informação, pelas inúmeras trocas de idéias. Em especial quero agradecer ao amigo Benitz, meu companheiro de administração do laboratório, aos amigos André Vinícius, Alberto e aos gaúchos (Éder, Fábio e Josué) pela força e amizade. Agradecimentos essenciais ao colega Carlos Alexandre pelo seu trabalho na implementação da interface do nosso protótipo.

Agradecimentos especiais a todos os funcionários e professores do Departamento de Sistemas e Computação. Em especial a Aninha e Jacques, pela amizade e incentivo; a Vera; Zeneide e ao pessoal da cantina (Inês, Romildo, Maria, Jô e Larissa) pelos momentos de descontração.

Agradeço às amigas Glaucimar, Eliane, Juliana Camboin, Fabiana, Beti, Rakel, Juliana Medeiros, Adna, Isabel, Marcela, Elaine, Alcieli e aos amigos Sergio (Espanha), Rob (EUA), Osman Jr. e Gustavo Cirne pelo grande apoio e amizade. Agradeço também ao "Recanto Zen" (Cláudia, Juliana, Érick, Érica, Amancio, Leandro, Kyller e Sandro) pelos bons momentos de alegria. Ao Amancio, agradeço o carinho, compreensão e grande apoio principalmente na fase final deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

A Web Semântica é uma visão da próxima geração da *web*, onde as informações são integradas, processadas, interpretadas e intercambiadas entre máquinas, com o objetivo principal de prover aplicações mais sofisticadas para o usuário *web*. Tal visão, considerada como a terceira geração da *web,* trouxe à tona grandes desafios para a comunidade científica. Um dos principais desafios a ser tratado é a heterogeneidade semântica, o que representa atualmente o maior empecilho para interoperabilidade entre sistemas e, conseqüentemente, um grande desafio para a integração de dados na *web*. Neste trabalho, especificamos um integrador semântico de dados na *web* baseado na arquitetura de mediadores, utilizando como suporte às questões semânticas, a abordagem de ontologias. Utilizamos linguagens como XML e DAML+OIL a fim de propor uma arquitetura, distribuída em n-camadas, que possa prover interoperabilidade entre dados estruturados e semi-estruturados.

Abstract

Semantic Web is a next web generation vision, where information is integrated, processed, interpreted and interchanged among machines in order to provide more sophisticated applications for web users. This vision, considered to be the third web generation, creates large challenges for the scientific community. One of the main obstacles that has to be overcome is semantic heterogeneousness, which creates major problems for interoperability and data integration on the web. In this dissertation, we specify a semantic data integrator on the web based on mediator architecture using the ontological approach to support the semantic tasks. We have also used XML and DAML + OIL languages to propose a n-layer architecture that can provide syntactical, structural and semantical interoperability as defined in the Semantic Web specification.

Capítulo 1

Introdução

Tradicionalmente, os Sistemas de Gerência de Banco de Dados (SGBDs) têm usado modelos de dados baseados em estruturas regulares. Como exemplo, tem-se o modelo relacional (introduzido nos anos 70 e largamente usado), o modelo orientado a objetos e o modelo objeto-relacional (mais recente) [O´Neil, 2000]. Todos os dados destes modelos possuem um esquema estrita e previamente definido.

Esta **cultura de banco de dados** fez surgir técnicas de armazenamento e linguagens de consultas que proporcionam acesso eficiente a grandes bases de dados altamente estruturadas. Também, trouxe modelos de dados e métodos para estruturar dados e mecanismos para manter a integridade e consistência de dados [Abiteboul *et al*, 2000]. Os modernos SGBDs implementam funcionalidades como as atualizações e extrações eficientes de grande quantidade de dados, processamento de transações altamente sincronizado, recuperação de falhas, índices, controle de integridade, "*triggers"* e consultas complexas com boa performance [Ullman, 1997; Elmasri, 2000].

Atualmente, com o advento, consolidação e grande crescimento da *web,* muitos dados dispostos na *web* não são mantidos em bancos de dados. Esta característica se deve à própria natureza dos dados *web*, sobretudo pela sua organização bastante heterogênea. Esta heterogeneidade de estrutura enquadra os dados da *web* no que foi denominado de dados semi-estruturados, pois estes nem são totalmente não-estruturados e nem estritamente tipados [Abiteboul, 1997]. Pode-se afirmar também que dados semi-estruturados são dados em que o esquema de representação está presente, explícita ou implicitamente, juntamente com o dado, ou seja, são *autodescritivos* [Buneman, 1997]. Tipicamente, quando se trabalha em programas com armazenamento de dados, estabelece-se uma estrutura (tipo, esquema) do dado e depois se cria, por exemplo, instâncias do tipo de esquema. Com os dados semi-estruturados, é preciso uma análise do dado para identificar sua estrutura [Abiteboul *et al*, 2000]. O gerenciamento dos dados semi-estruturados pelos SGBDs tradicionais não pode ser feito de forma direta, tornando-se, assim, um problema para a cultura de banco de dados [Bourret, 2000].

A cultura web proporcionou a criação de uma infraestrutura global e um conjunto de padrões para suportar a troca de documentos, um formato de apresentação para hipertexto (HTML) e técnicas para recuperação de informações. Porém, a alta heterogeneidade, autonomia e ampla distribuição dos dados *web* sem uma padronização mínima, tornam a consulta dos dados complexa, já que não se tem um esquema uniforme para se fazer uma consulta. Hoje em dia, as consultas são efetuadas geralmente através da navegação exaustiva (que dificilmente resulta em sucesso, devido ao grande volume de dados a ser consultado e que acarreta, muitas vezes, a perda do encadeamento da navegabilidade), ou busca por palavras-chaves (full-text search) onde são utilizadas técnicas de recuperação de informações através das ferramentas de busca, tais como Altavista (www.altavista.com) e Google (www.google.com). Estas ferramentas de busca apresentam como provável desvantagem respostas de consultas que abrangem diferentes categorias de informações, pois deste modo, cabe ao usuário a análise de relevância das informações.

Devido a estas deficiências, a *web* vem se tornando um desafio para a comunidade científica, a qual portanto, focou muitos estudos em soluções para os problemas emergidos com a consolidação e crescimento de sistemas *web,* principalmente no que tange à integração, intercâmbio e entendimento semântico das informações. Várias iniciativas concentraram esforços na criação de padrões, modelos, linguagens, arquiteturas de metadados, dentre outros, para atender aos novos requisitos da *web*. Alguns projetos desenvolvidos pela W3C (http://www.w3.org), tais como a metalinguagem XML (*Extensible Markup Language*) e a arquitetura RDF (*Resource Description Framework*) [Lassila & Swick, 1999], por exemplo, visam dar suporte à codificação, ao transporte e ao intercâmbio dos dados e metadados. Não obstante, este intercâmbio deve possibilitar a interoperabilidade *semântica, sintática e estrutural.*

A arquitetura RDF define um mecanismo para descrever recursos de uma forma genérica, ou seja, não vinculados a um domínio específico de aplicação. Possui também a característica de determinar o relacionamento entre entidades através de triplas (sujeito, predicado e objeto) e de prover a interoperabilidade estrutural, a qual especifica a organização dos recursos com seus tipos e os possíveis valores para cada tipo.

A linguagem XML é derivada do SGML (*Standard Generelized Markup Language*) e, por não definir "*tags"* (instruções embutidas no corpo de documentos) nem a gramática para uma linguagem, é completamente extensível [W3C, 2000c]. Atualmente, está sendo usada na representação de estruturas de dados estruturados e semi-estruturados em documentos próprios para o processamento entre máquinas e intercâmbio na *web*. Possibilita a interoperabilidade sintática, a qual determina como os dados e metadados devem ser codificados para a troca de informações.

Entretanto, apesar de se resolver grandes problemas com a padronização e modelagem de dados, alguns outros problemas ainda são pertinentes. XML é definida apenas em nível sintático e RDF apesar de prover estruturas para descrições semânticas, é muito limitada. Assim, por exemplo, não é possível fazer consultas semânticas devido à falta de um modelo conceitual e portanto, também não é possível que máquinas determinem sem ambigüidades o significado correto das *tags*. Ademais, conhecimentos implícitos não podem ser explorados quando os documentos são acessados diretamente e estes conhecimentos não são disponibilizáveis como informações para consultas [Erdmann, 1999].

Este "gargalo" semântico na *web* veio à tona principalmente pela necessidade de uma comunicação não unicamente humana, ou seja, a necessidade do auxílio de uma comunicação independente entre as máquinas. Porém, um dos aspectos que dificulta o atendimento desta necessidade é o fato de que até hoje a maioria das informações na *web* é projetada para a concepção humana. Toda esta situação contribuiu para o surgimento da Web Semântica (*Semantic Web)*, considerada atualmente como a terceira geração da *web*. A Web Semântica tem como objetivo proporcionar estruturas conceituais, como o desenvolvimento de linguagens, para expressar informações que possam ser lidas e interpretadas por máquinas [Berners-Lee et al, 2001; SemanticWeb, 2002]. Ela visa prover acesso automatizado à informação através de processamento semântico dos dados, ampla utilização de metadados e heurísticas feitas por máquinas. É importante ressaltar que a Web Semântica não está apenas ligada ao conteúdo dos recursos, mas também, na forma como eles se relacionam na *web.* Logo, é importante que estes recursos disponibilizados devam ser expressivos o bastante para que possam ser processados por máquinas. Ademais, como citado anteriormente, os sistemas deverão atender à tripla da interoperabilidade (semântica, sintática e estrutural) para que a Web Semântica funcione de forma efetiva.

Para atender a interoperabilidade semântica (essencial nesta nova visão), a qual possibilita compreender o significado de cada elemento do recurso com suas associações, constatou-se essencial o uso de vocabulários específicos, ontologias e/ou padrões de metadados. Uma ontologia, tipicamente, contém um vocabulário de conceitos ou classes, estrutura de grafo, relacionamentos entre conceitos, atributos de conceitos e um conjunto de axiomas que definem as regras do negócio [Erdmann, 2000], servindo como uma solução para a recuperação semântica de informações nas fontes de dados semi-estruturados.

Através das ontologias, dentre outras coisas, é possível explicitar a semântica dos dados e um consenso de interpretação semântica sobre um domínio. Em [Hendler, 2001], Hendler expõe a visão de que a Web Semântica será uma teia de pequenos componentes ontológicos que apontam entre si e que serão criados pelos usuários da *web* do mesmo modo que atualmente o conteúdo desta está sendo criado.

A partir das tecnologias mencionadas anteriormente (XML e RDF), juntamente com as ontologias, a Web Semântica oferece uma rede enorme de conhecimento humano, que pode ser processada pelas máquinas e conseqüentemente inferir novos conhecimentos. A figura 1.0 mostra a relação de camadas na Web Semântica proposta por Tim Berners-Lee.

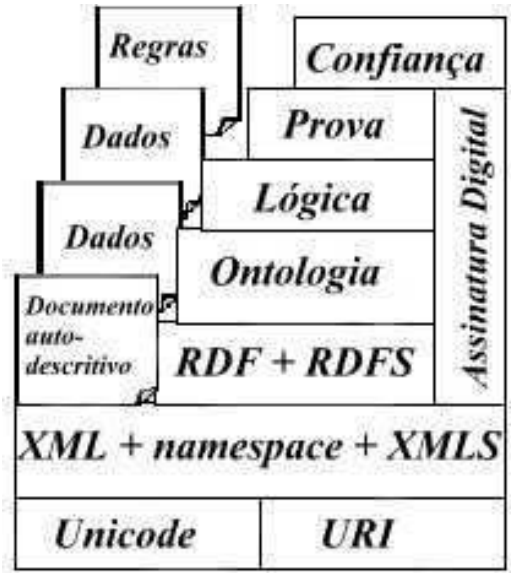

Figura 1.0: Camadas da Web Semântica segundo Tim Berners-Lee

A heterogeneidade semântica é considerada atualmente como o principal empecilho para a interoperabilidade semântica, gerando um grande desafio para integração de dados na *web*. Conseqüentemente, o clássico problema da integração de dados vem ganhando cada vez mais atenção pela comunidade de pesquisa, demandando assim, soluções que tornem possível a integração de dados nestes sistemas de informações altamente heterogêneos (dados semi-estruturados e estruturados) considerando as novas tendências dos serviços "inteligentes".

Também no mundo *web,* enfrentam-se muitos outros problemas que não eram considerados há alguns anos. Os sistemas distribuídos geograficamente devem funcionar com eficiência e precisão e os dados de diversas fontes devem ser transferidos sem haver perda de informação. As aplicações devem estar aptas a se comunicar, não apenas com componentes de negócios, mas também com outros sistemas de negócios, como, por exemplo, com aplicações B2B (*Business to Business*). Os clientes podem ser navegadores web ("*browsers*") que suportam HTML, telefones móveis que suportam WML, ou PDAs (*Personal Digital Assistent*) com alguma linguagem de marcação [Wait, 1999; Hall, 2000].

Como a pesquisa dos principais recursos apresentados para a problemática da integração de dados na *web* ainda é incipiente, principalmente no que tange a arquiteturas de integração para a composição de ferramentas que suportem a visão da Web Semântica, a gerência da transformação e apresentação dos dados no ambiente *web*, vislumbra-se uma interessante área de pesquisa.

1.1 - Objetivos da dissertação

Como visto na seção anterior, o problema de gerenciamento e consulta de caráter semântico dos dados semi-estruturados é uma questão em aberto, além disto, o problema de integração destes dados com os dados estruturados tem recebido bastante atenção pela comunidade de banco de dados [Abiteboul *et al*, 1997; Lahiri *et al*, 1999; Bergamaschi *et al*, 1999].

O objetivo geral desta pesquisa é propor uma solução que ofereça a integração de dados semi-estruturados e dados estruturados através de consultas com caráter semântico, com suporte de ontologias, visando atender aos requisitos da Web Semântica [Arruda *et al*, 2002].

1.2 - Estrutura da Dissertação

Os capítulos seguintes deste documento estão estruturados de acordo com a ordem descrita a seguir.

No capítulo 2, apresentamos alguns conceitos relacionados com a solução proposta. Na seção 2.1 tem-se uma breve explanação sobre os sistemas de integração de dados, apresentando as principais abordagens, arquiteturas e os problemas pertinentes. A seção 2.2 apresenta conceitos e principais características dos dados semi-estruturados e XML. Na seção 2.3 descrevemos as ontologias, seu emprego na integração semântica de dados, sua relação com os dados semi-estruturados e as principais linguagens utilizadas para representá-las atualmente. A seção 2.4 mostra os principais trabalhos relacionados, suas abordagens e uma breve comparação com nossa solução.

No capítulo 3, apresentamos a análise da nossa proposta de um integrador semântico de dados. A seção 3.1 mostra a arquitetura proposta, já a seção 3.2 detalha o funcionamento dos componentes desta arquitetura. Os requisitos levantados são apresentados na seção 3.3. O modelo de caso de uso (*use case)* e o modelo conceitual são expostos respectivamente nas seções 3.4 e 3.5.

O capítulo 4 apresenta o projeto e implementação de um integrador semântico de dados analisado no capitulo 3. O funcionamento de classes e interfaces será explorado na seção 4.1. A seção 4.2 mostra a interface do protótipo desenvolvido. O processo de correspondência sintática e mapeamento utilizado pelos *wrappers* é detalhado na seção 4.3. A seção 4.4 mostra um estudo de caso com o intuito de exemplificar a utilização do protótipo desenvolvido.

Por fim, no capítulo 5, temos a conclusão deste trabalho de pesquisa e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo apresentamos alguns conceitos que balizam a nossa pesquisa. A seção 2.1 ressalta o clássico problema da integração de dados, mostrando as principais abordagens e arquiteturas utilizadas, detalhando também as dificuldades ainda encontradas no desenvolvimento de um sistema de integração.

Na seção 2.2 apresentamos conceitos e características gerais dos dados semi-estruturados e da linguagem XML. Também são documentadas as principais características das linguagens de consulta para dados semi-estruturados e as propostas deste tipo de linguagens para documentos XML.

A seção 2.3 introduz conceitos, classificação e aplicações das ontologias. Algumas propostas de linguagens para especificação de ontologias também são apresentadas nesta seção.

Por fim, na seção 2.4, citamos alguns dos diversos trabalhos relacionados com a integração semântica de dados, procurando ressaltar os principais pontos que diferenciam estes da nossa proposta.

2.1- Sistemas de Integração de Dados

Um dos principais aspectos a ser considerado em um sistema de integração de dados é prover uma visão de acesso unificado aos dados, independente da localização desses dados. Essas fontes de dados podem ser independentes e dinâmicas. Independentes no sentido de manter normalmente suas aplicações locais e dinâmicas por poderem modificar e/ou atualizar tanto seus dados quanto seus esquemas para o sistema de integração. São geralmente fontes de dados pré-existentes, que foram criadas independentemente e que contêm dados numerosos e que mudam constantemente. As fontes também podem ser incompletas ou parcialmente completas.

Tudo isto implica na necessidade de um sistema de integração que tenha um mecanismo flexível para o tratamento destas particularidades supracitadas.

Uma das principais vantagens de um sistema de integração de dados é de poder proporcionar um ambiente em que os usuários especificam o que querem pesquisar e não precisam pensar como obter as respostas. Conseqüentemente, isso evita que o usuário tenha a tarefa tediosa de pesquisar as fontes relevantes, interagir com cada uma delas e combinar os dados resultantes das múltiplas fontes [Levy, 1999].

Percebe-se dois tipos de abordagens utilizadas na construção de um sistema de integração de dados:

- Materializada. As informações relevantes das fontes são extraídas e armazenadas em um repositório central do sistema. Uma implicação muito importante nesta abordagem é a manutenção da consistência dos dados armazenados no repositório com os dados das fontes de origem. Portanto, é mais adequada para informações integradas com pouca necessidade de atualizações, como informações históricas por exemplo. O principal representante desta abordagem é o Data Warehouse. A próxima sessão explana melhor a arquitetura de Data Warehousing, porém, para maiores informações sobre tais tipos de sistema, recomendamos a leitura de [Kimball, 1996; Kimball *et al*, 1998 e Dodge, 2000].
- Virtual. As informações são extraídas dinamicamente das fontes integradas conforme os requisitos das consultas sobre o sistema. No entanto, estas fontes deverão sempre estar disponíveis para o acesso a seus dados.

2.1.1 - Arquiteturas de Integração de dados

Há diversas maneiras de se fazer um sistema de integração. Porém as abordagens mais comuns segundo Garcia-Molina [Garcia-Molina *et al* , 2000] são:

- Arquitetura Federada. Segue a abordagem virtual. Os bancos de dados são independentes e cooperantes. Na Figura 2.0 tem-se uma situação exemplo, onde existem quatro bancos de dados na federação. Cada banco de dados precisa de três componentes, um para cada um dos outros três bancos de dados.

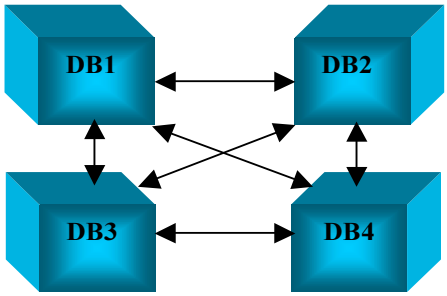

Figura 2.0: Arquitetura Federada

- Warehousing. Um único banco de dados, chamado *warehouse*, armazena dados de diversas fontes. Esta arquitetura está associada à abordagem materializada. Os dados são recuperados, integrados e armazenados em um repositório de dados para consultas posteriores. A atualização dos dados deve ser efetuada para manter a consistência do repositório com as fontes de origem. Existem basicamente três maneiras de tratar o problema da atualização: i) O *warehouse* é periodicamente reconstruído a partir dos dados nas fontes da integração. Este processo deverá ser feito em um período que ocorra o mínimo de transações, pois requer a paralisação do *warehouse* e conseqüentemente as consultas não podem ser realizadas, ii) O *warehouse* é atualizado periodicamente por um processo chamado *atualização incremental*. Neste processo, as mudanças nas fontes locais são refletidas incrementalmente para o *data warehouse*, iii) O *warehouse* é atualizado imediatamente quando uma ou mais fontes são mudadas. A figura 2.1 mostra um exemplo de uma arquitetura de *data warehouse.*

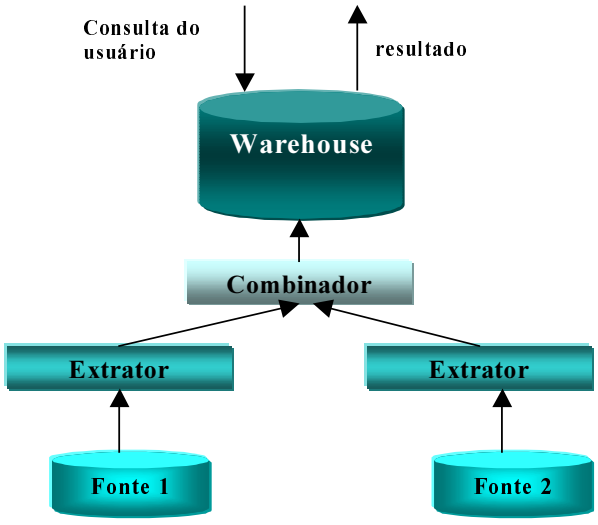

Figura 2.1: Arquitetura de Data Warehouse

- Mediadores. Os mediadores são componentes de software que suportam um banco de dados virtual. Eles integram várias fontes e possuem um esquema global, provendo uma interface uniforme de acesso aos dados. Entretanto, os mediadores não armazenam dados nem os usuários realizam as consultas da forma materializada como é feita no *data warehouse.* Um mediador traduz as consultas nos termos das fontes distribuídas através de *wrappers*, integra os resultados e devolve a resposta ao usuário [Wiederhold, 1994]. A figura 2.2 mostra um exemplo de integração feita por mediador.

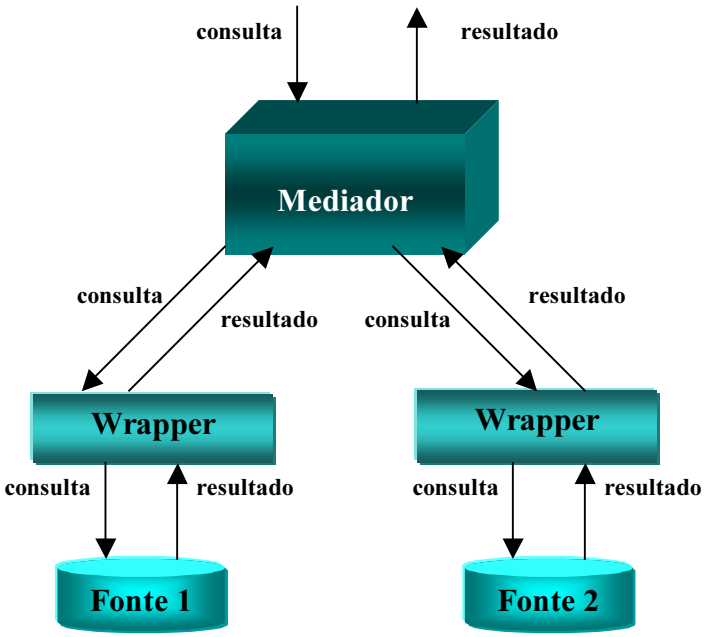

Figura 2.2: Arquitetura de Mediadores

Percebe-se que a arquitetura de *data warehouse* e a arquitetura de mediadores possuem uma abordagem "*multitier*", ou seja, de múltiplas camadas. Esta abordagem é própria para a *Web*. Geralmente formada por uma camada de mais alto nível, a camada do cliente, que possui serviços de interface para os usuários finais, por uma camada de mais baixo nível, que compreende os servidores de dados e um conjunto de camadas intermediárias denominadas de "*middleware*", *software* responsável por transformar e integrar dados.

2.1.2 – Problemas e desafios da integração de informação

Como descrito anteriormente, o objetivo principal de um sistema de integração é prover uma interface uniforme para uma coleção de diferentes fontes. Em [Garcia-Molina *et al* , 2000] citam-se alguns dos problemas mais comuns encontrados quando se trabalha com um sistema de integração:

- Diferentes tipos de dados entre as diferentes fontes;
- Diferente convenção de valores. As fontes podem usar diferentes nomes ou formatos para referenciar o mesmo objeto. Como exemplo, temos as diferentes convenções para especificar datas ou endereços.
- Diferenças semânticas. Muitos termos podem ter diferentes interpretações nas fontes. Por exemplo, uma determinada fonte relacional pode armazenar todos os atributos de um livro em uma única tabela, enquanto que uma outra fonte pode representar os atributos através de relacionamentos. Além disto, os nomes das tabelas e dos atributos provavelmente serão diferentes de uma fonte para outra.
- Valores ausentes. A informação de um tipo pode não ser definida em uma fonte, mas outras podem defini-lo.

Alguns outros desafios são levantados por [Levy, 1999] na construção de um sistema de integração de dados usando mediadores:

- Modelagem de dados. Um esquema mediador é um dos primeiros passos que um projetista deve considerar ao desenvolver uma aplicação de integração. Este esquema descreve os dados das fontes integradas que são importantes para os usuários. Existem duas abordagens para definir este esquema global:
  - Visão Global. O esquema global é uma visão dos esquemas locais. Tem a vantagem de que a tradução da consulta no esquema global em consultas nos esquemas locais é mais simples. Por outro lado, para suportar modificações das fontes locais, é preciso construir um novo esquema global considerando todo o conjunto de fontes modificadas.

- Visão Local. A visão local consiste em definir as fontes locais como visões do esquema global. Em contraste com a visão global, modificações nas fontes locais são tratadas nas definições isoladas para cada fonte. Entretanto, nesta abordagem é preciso que as consultas descritas no esquema global sejam reescritas, ou seja, reformuladas para cada esquema local.

Adicionado ao esquema mediador, são necessários descritores (metadados) das fontes de dados, os quais devem especificar, por exemplo, seus conteúdos, atributos, regras e capacidades de processamento de consulta. Esta tarefa é mais complexa quando consideramos a integração de dados na *web*, pois seu esquema, entre outras características, pode ser muito extenso e de rápida evolução. Na seção 2.2 do presente capítulo mostramos com detalhes estes tipos de dados (semi-estruturados). Além dos descritores, deve-se especificar as correspondências entre os elementos do esquema do sistema integrado e do esquema das fontes locais. Geralmente adiciona-se ao processo de identificação de correspondências dicionários comuns, para resolver problemas de semântica, ou técnicas como thesaurus e/ou ontologias, para descrever termos e relacionamentos de um domínio [Salgado, 2001].

- Reformulação da consulta. Um sistema de integração de dados da abordagem virtual deve possuir um módulo que traduza a consulta do usuário, a qual é baseada no esquema mediador, numa consulta adequada ao processamento nas fontes locais. É importante assegurar que fontes irrelevantes não são acessadas e que a reformulação seja semanticamente correta.
- *Wrappers*. Um *wrapper* é um programa específico para cada fonte de dados. Tem como função geral encapsular o acesso a uma única fonte de dados, tornando-a mais utilizável. Num sistema de integração, *wrappers* têm como objetivos principais traduzir consultas de uma camada superior em consultas específicas das fontes correspondentes na camada inferior do sistema e converter os dados de uma fonte local para o formato de um modelo de dados comum [Wells, 1999; Hammer *et al*, 1998].
- Otimização e execução de consulta. Os processos de otimização e execução de consultas apresentam diferenças significantes nas abordagens de um sistema tradicional de banco de dados e em um sistema de integração. Entre elas podemos citar:

- Em um sistema de integração, as fontes são autônomas, portanto o otimizador pode não ter estatísticas confiáveis sobre as fontes.
- As fontes de um sistema de integração possivelmente têm diferentes capacidades de processamento, portanto um otimizador de consultas em um sistema de integração precisa considerar a possibilidade de explorar as capacidades de processamento das fontes.
- Nos banco de dados tradicionais pode-se estimar o tempo de transferência de dados de um disco para a memória. Entretanto nos sistemas de integração, os dados são geralmente transferidos em ambientes de rede e os atrasos podem ocorrer por diversos motivos. Isto pode tornar ineficiente um plano baseado em estimativas de custos.

2.2 - Dados semi-estruturados e XML

Na seção anterior, apresentamos os principais aspectos e desafios de projeto encontrados nos sistemas de integração de dados. Um outro ponto importante que devemos considerar na integração de dados é a presença de dados semi-estruturados. Isso se deve pelas seguintes razões:

- Atualmente, principalmente com a consolidação da *web,* muitas fontes de dados são semi-estruturadas;
- A integração de diversas fontes heterogêneas, onde suas estruturas são parcialmente conhecidas e podem mudar sem aviso e a usual convenção de um esquema comum a todas elas, desencadeia a manipulação de dados semi-estruturados.

Dados semi-estruturados são dados em que o esquema de representação está presente, explícita ou implicitamente, juntamente com o dado, por isso, geralmente são chamados de auto-descritivos. Pode-se dizer ainda que estes dados não são nem totalmente não-estruturados nem estritamente tipados. O principal exemplo de dados semi-estruturados são os dados *web* [Buneman, 1997; Abiteboul *et al*, 2000].

Dados semi-estruturados aparecem naturalmente na integração de dados, mesmo quando as fontes de dados são estruturadas [Levy, 1999]. Entretanto, com a notória necessidade da integração de dados envolvendo o ambiente *web,* as pesquisas sobre os dados semi-estruturados tornaram-se mais intensas. Algumas razões contribuíram para emergir muitos estudos nesta área [Buneman, 1997; Abiteboul, 1997]:

- Há fontes de dados como a *web*, as quais desejam-se tratar como BDs, mas que não podem ser restritos por um esquema;
- É desejável ter um formato extremamente flexível para troca de dados entre BDs;
- Mesmo usando dados estruturados, poderá ser útil ter uma visão semi-estruturada desses para navegação.

Como características gerais dos dados semi-estruturados, pode-se citar [Abiteboul, 1997; Suciu, 1998a; Florescu *et al*, 1998]:

- Definição à *posteriori*: geralmente os esquemas para dados semi-estruturados, são definidos posteriormente aos seus dados, baseando-se em uma investigação de suas estruturas particulares e da análise de similaridades e de diferenças;
- Estrutura irregular: não existe um esquema padrão para dados semi-estruturados, diferentes instâncias que podem ser classificadas como de mesmo tipo, podem ter estruturas diferentes. Como exemplo, podemos ter um endereço estruturado como uma simples *string* em uma fonte, e em outra, estruturado com uma tupla;
- Estrutura implícita: geralmente uma estrutura básica existe para os dados, porém de maneira implícita. É necessária uma computação (*parsing*) para obter essa estrutura;
- Estrutura parcial: nem todos os dados possuem uma estrutura. Por exemplo, componentes de objetos que são arquivos *bitmaps* são não-estruturados. Já um catálogo de produtos pode ter uma estrutura (implícita ou explícita). Deste modo, um esquema para estes dados nem sempre é completo (semanticamente) e nem sempre todas as informações esperadas estão presentes;
- Estrutura extensa: devido a sua grande heterogeneidade (diferentes formatos para um mesmo assunto), uma união de atributos significativos em cada formato pode produzir um esquema extenso em relação à quantidade de informações;

- Estrutura evolucionária: freqüente modificação de estrutura. Como bom exemplo, temos os dados *web*, onde geralmente ocorrem muitas atualizações.
- Estrutura descritiva e não prescritiva: devido às características evolutivas e irregulares dos dados, ao contrário das aplicações de banco de dados tradicionais, o esquema estático normalmente se restringe a descrever o estado corrente dos dados. A estrutura não estabelece tipos rigorosos que devem ser seguidos pelas instâncias, apenas indica um tipo. Todos dados novos são aceitos e violações do esquema são toleradas.
- Distinção não clara entre a estrutura e os dados: geralmente a distinção lógica entre estrutura e valor não é clara, devido a estrutura estar embutida na representação dos dados. Por exemplo, pode-se manter um número de telefone como um valor em uma fonte (inteiro) e em outra fonte como um tipo (classe telefone).

As características apresentadas acima apresentam várias diferenças, se comparadas com as características de dados mantidos em BDs tradicionais.

Na tabela 2.0 faz-se um comparativo resumido entre os dados tradicionais e os dados semi-estrututrados:

Dados tradicionais	**Dados semi-estruturados**
Esquema predefinido	Inexistência de esquema predefinido
Estrutura homogênea	Estrutura heterogênea
Estrutura independente dos dados	Estrutura embutida no dado
Estrutura reduzida	Estrutura extensa
Estrutura pouco evolutiva	Estrutura fortemente evolutiva
Estrutura prescritiva	Estrutura descritiva
Distinção entre estrutura e dado é clara	Distinção entre estrutura e dado não é clara

Tabela 2.0: Comparativo entre os dados semi-estruturados e os dados estruturados [Melo *et al*, 2000].

2.2.1- Modelagem

Idealmente, um modelo para dados semi-estruturados deve ser ao mesmo tempo um modelo simples (não apresenta muitos construtores semânticos para representar um esquema) e rico (com estruturas particulares como tipos atômicos, coleções e tipos compostos) [Abiteboul, 1997].

Algumas das formas usuais de modelar dados semi-estruturados é através de grafos direcionados rotulados e através de estrutura de árvore. No modelo de grafos, geralmente os vértices representam os objetos e as arestas são arcos para outros objetos que fazem parte da sua estrutura. A estrutura particular de cada dado está embutida no próprio grafo, tornando-se assim, bem apropriado para dados semi-estruturados. Geralmente faz-se a representação dos grafos em estrutura de árvore. [Florescu et al, 1998; Buneman, 1997].

Um modelo pioneiro para dados semi-estruturados é o OEM (Object Exchange Model), definido com o objetivo de ser simples e de propósito geral. Outro exemplo de modelo é o ADM (Araneus Data Model), que é um modelo de dados baseado no padrão ODMG (Object Database Management Group) de representação de objetos, voltado para o contexto da *web* [Abiteboul *et al*, 2000]. Atualmente, XML vem sendo também usada na representação de dados semi-estruturados. A seguir, descrevemos suas principais características e tecnologias relacionadas.

2.2.2 - XML

XML (eXtensible Markup Language) é uma metalinguagem de marcação derivada do SGML que foi desenvolvida pelo consórcio W3C (World Wide Web Consortium) e foi consolidada como um padrão para publicação, combinação e intercâmbio de documentos. Atualmente é usada também na representação de estruturas de dados estruturados e semi-estruturados e seu intercâmbio na *web* [W3C, 2000c]. As principais características de XML podem ser citadas como:

- Independência de conteúdo e de apresentação. XML refere-se apenas ao conteúdo. A apresentação de XML pode ser feita pela CSS (Cascading Style Sheets) ou pela XSL (Extensible Stylesheet Language), ambas da W3C;

- Extensível. Pode-se definir as próprias “tags” de acordo com a utilidade na aplicação, diferenciando da linguagem HTML que já possui “tags” pré-definidas;
- Validação. Os documentos XML podem estar associados com uma linguagem de esquema, que define estruturas e restrições para um documento. Assim, os dados podem ser validados conforme esta estrutura.

Além das características vistas anteriormente, a linguagem XML vem se tornando muito importante por duas principais razões [Rosenthal, 1999]:

- Tem suporte em grandes corporações comerciais, por exemplo, Microsoft, IBM, Sun, Oracle e Netscape. Como resultado, há um grande mercado para as ferramentas XML;
- Há vários padrões relacionados ao XML que aumentarão ainda mais o seu impacto.

Os documentos XML possuem uma estrutura básica, composta de um cabeçalho, chamado de *prólogo*, e do restante do documento, chamado de *instância.* No prólogo, ficam informações que identificam o documento como sendo XML, tais como a versão da linguagem e o tipo de documento ao qual ele está associado. A instância do documento segue o prólogo, contendo os dados propriamente ditos, organizados como uma hierarquia de elementos. O principal conceito de XML é o elemento. O elemento é o tipo mais comum de marcação em XML, e é delimitado pelo rótulo (“tag”) inicial e pelo rótulo de fim, que envolvem a informação. Os atributos são qualificadores dos elementos, estes sempre são representados como um par atributo-valor, sendo que o valor deve, obrigatoriamente, estar entre aspas. O texto a seguir mostra um exemplo de um documento XML.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<-- DTD externa -->
<!DOCTYPE notaPedido SYSTEM “pedido.dtd”>
<! -- Aqui começa a instância -->
<notaPedido>
        <destino>
                <nome>César Celestino Arruda</nome>
                <end>Vigário Calixto, 34 Catolé</end>
                <cidade>Campina Grande</cidade>
                <pais>Brasil</pais>
        </destino>
        <itens>
                <item cod = 001>
                        <titulo>ICEIS 2002</titulo>
                        <quantidade>1</quantidade>
                        <preço>14.90</preço>
                </item>
                <item cod = 301>
                        <titulo>Bon Jovi alive</titulo>
                        <quantidade>1</quantidade>
                        <preço>9.90</preço>
                </item>
        </itens>
</notaPedido>
```

Um documento XML é considerado bem-formatado quando está coerente sintaticamente com a especificação da linguagem XML. Pode-se também criar algumas regras que descrevem a estrutura de um documento XML, os elementos que podem participar do documento e os atributos que podem estar associados a estes elementos. A partir dessas regras, um documento XML pode ser classificado como válido ou não. Um documento é válido se estiver bem formatado e de acordo com o esquema que define sua estrutura. Algumas linguagens para definição de esquemas para XML têm sido propostas [Lee, 2000], entre as mais usuais tem-se a DTD [Bradley, 1998; W3C, 2000c] e XML Schema [W3C, 2000b].

DTD (Document Type Definition) é uma definição que estabelece um conjunto de restrições para um documento XML. Ela define a maneira como um documento XML deve ser construído, através da definição de uma hierarquia de elementos.

A validação do documento XML para uma DTD é feita por um "*parser"* XML, ele verifica principalmente se os elementos presentes nos documentos XML estão definidos na DTD. O texto a seguir, mostra uma DTD para o exemplo do documento XML visto anteriormente.

```
<!ELEMENT destino (nome, end, cidade, pais)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT end (#PCDATA)>
<!ELEMENT cidade (#PCDATA)>
<!ELEMENT pais (#PCDATA)>
<!ELEMENT itens (item+)>
<!ELEMENT item (titulo,quantidade,preço)>
<!ATTLIST item cod ID #required>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT quantidade (#PCDATA)>
<!ELEMENT preço (#PCDATA)>
```

DTD é a forma mais utilizada para se classificar documentos XML. Porém, apresenta alguns problemas e deficiências, apontados por [Walsh, 1999]:

- Não é escrita em XML;
- Não possui suporte para *namespace*s, ou espaço de nomes (descritos posteriormente);
- Oferece recursos muito pobres para se definirem tipos de dados, principalmente no que se refere a números, datas, etc;
- Possui um mecanismo de extensão complexo e frágil, baseado em pouco mais do que substituição de cadeias de caracteres.

XML-Schema é uma proposta da W3C por auxiliar e eventualmente substituir o DTD, pois promete ser mais expressiva que a DTD e mais útil por uma grande variedade de aplicações [Lee, 2000]. Ela tem muitos mecanismos que permitem expressar tipos de dados, herança, tipos abstratos, unicidades e chaves, etc [W3C, 2000b]. Suporta *namespaces* e é escrita em XML. No texto seguinte, tem-se um exemplo de XML *Schema* equivalente à DTD vista anteriormente.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="notaPedido" type="pedido"/>
        <xs:complexType name="pedido">
                <xs:sequence>
                        <xs:element name="destino" type="entregaEnd"/>
                        <xs:element name="itens" type="cdItens"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="entregaEnd">
                 <xs:sequence>
                        <xs:element name="nome" type="xs:string"/>
                        <xs:element name="end" type="xs:string"/>
                        <xs:element name="cidade" type="xs:string"/>
                        <xs:element name="pais" type="xs:string"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="cdItens">
                <xs:sequence>
                        <xs:element name="item" type="cdItem"
                                maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="cdItem">
                <xs:sequence>
                        <xs:element name="titulo" type="xs:string"/>
                        <xs:element name="quantidade" type="xs:positiveInteger"/>
                        <xs:element name="preço" type="xs:decimal"/>
                </xs:sequence>
                <xs:attribute> name = “cod” type =  “xs:ID” use = “required” />
        </xs:complexType>
</xs:schema>
```

Alguns padrões foram propostos em função do XML. Entre eles temos:

- ***XML Namespace.*** Descreve a sintaxe de “namespace”, ou espaço de nomes, e que serve para criar prefixos (xmlns:*prefixo*), como etiquetas, para os nomes de tags, evitando confusões que possam surgir com nomes iguais para “tags” que definem dados diferentes [W3C, 1999; W3C, 2002c].

- ***A linguagem XSL.*** XSL (Extensible Stylesheet Language) transforma e traduz dados XML de um formato para outro. XSL é organizada em duas partes:
  - *XSL Transformation* (XSLT): Responsável por especificar transformações de documentos XML;
  - *XSL Formatting Objects* (XSLFO): Similar a CSS. Descreve como exibir os elementos de um documento, especificando fontes, cores, indentação, etc.

  Por exemplo, um documento XML pode ser transformado para HTML, PDF ou Postscript [McLaughlihn, 2000; W3C, 2001b; Cover, 2002b]. Com o XSL, tem-se uma completa separação entre dados (ou conteúdo) e a apresentação. Tornando-se uma excelente solução para mapeamento do XML para uma outra representação.
- ***SAX.*** O SAX (Simple API for XML) provê um "framework" baseado em eventos para o "parser" dos dados XML. A implementação de suas interfaces permite um controle completo sobre o processo de acesso do "parser" sobre o conteúdo de um documento XML. Mais adequado quando o processamento do documento é seqüencial, pois não permite acesso randômico [McLaughlihn, 2000; SAX, 2002; Brownell, 2002].
- ***DOM.*** DOM é uma API para o Document Object Model. Enquanto o SAX provê acesso aos dados dentro de um documento XML, o DOM é designado para prover uma maneira de manipular os dados. Ele representa um documento XML como uma árvore. O DOM também lê o documento inteiro na memória, armazenando os dados em nodos, o que torna o acesso rápido ao documento todo. Porém dependendo do tamanho e complexidade do documento, isto pode exigir demais dos recursos ou mesmo incapacitar a aplicação [Abiteboul *et al,* 2000; W3C, 2002b].

Tanto um documento XML quanto descrições textuais de objetos OEM apresentam uma estrutura aninhada. Além disto, as características dos dados semi-estruturados apresentadas anteriormente nesta seção (estrutura irregular, parcial, implícita, evolucionária, etc) são aplicadas também aos documentos XML [Abiteboul *et al,* 2000].

2.2.3 – Linguagens de Consultas

A consulta a fontes de dados semi-estruturados é um problema que ainda precisa de soluções definitivas, devido à dificuldade de se definir uma estrutura de representação uniforme que sirva de base para as pesquisas. Muitas linguagens estão sendo desenvolvidas. Segundo Buneman [Buneman, 1997], existem duas abordagens gerais para criar as linguagens de consultas a dados semi-estruturados:

- Tomar as linguagens SQL ou OQL como ponto de partida e adicionar novas características suficientes para atender aos desafios propostos pelos dados semi-estruturados.
- Começar uma linguagem baseada em alguma noção formal de dados semi-estruturados.

Segundo [Abiteboul *et al*, 2000], alguns requisitos genéricos para estas linguagens de consultas podem ser citados:

- Poder de expressão. Ainda não está claro qual poder de expressão uma linguagem de consulta para dados semi-estruturados deverá ter, mesmo sendo possível listar alguns tipos de operações que a linguagem deveria expressar em enfoques específicos;
- Semântica. É necessária uma semântica precisa, pois só assim se pode discutir transformação ou otimização de consultas. Ademais, precisa-se definir que tipo de semântica se pode extrair (ou adicionar) da sintaxe na qual os dados estão expressos. Este último ponto é especialmente importante para XML, cuja especificação é puramente sintática;
- Composição. O resultado de uma consulta deve poder ser utilizado como entrada para outra consulta, sendo bastante importante para a construção de visões, por exemplo;
- Esquema. As linguagens de consultas deverão ser capazes de identificar um esquema (caso este seja implícita nos dados) e explorá-lo a fim de fazer verificação de tipos, otimização, etc;

Diversas linguagens de consulta para dados semi-estruturados vêm sendo desenvolvidas. Muitas delas estão voltadas para consultas a dados XML, devido a grande tendência de padronização de XML para intercâmbio de informações na *web*.
Entretanto, fazer consultas em documentos XML representa um novo desafio para a comunidade de bancos de dados, devido à perspectiva de que grandes volumes de dados sejam disponibilizados nesse formato, gerando a necessidade de ferramentas de consulta que sejam, ao mesmo tempo, flexíveis o suficiente para compreender a heterogeneidade dos documentos e poderosas a ponto de extraírem informações com eficiência e eficácia [Fernandez *et al*, 1999; W3C, 2000a; Deutsch e*t al*, 1999a e Deutsch *et al*, 1999b].

De acordo com Baru [Baru *et al,* 1998], as linguagens de consultas para dados semi-estruturados até agora desenvolvidas pertencem a dois paradigmas:

- O paradigma de Banco de dados. Apropriada para selecionar e integrar objetos, permitindo também uma fácil navegação recursiva. São linguagens SQL-like ou OQL-like, como XML-QL e Lorel. Ou seja, que efetuam consultas do tipo *select-from-where*, típicas de bancos de dados relacionais e orientados a objetos. Como resultado, essas linguagens sempre fornecem documentos XML;
- O paradigma da programação funcional. Exemplificado por XSL, XQL, XQuery e XPath. Particularmente apropriada para transformações de documentos XML.

Algumas outras características gerais desejáveis em uma linguagem de consulta para XML são apontadas por [Maier, 1998; W3C, 2000a]:

- Espera-se que toda linguagem produzida para consulta em XML também produza como resultado um documento XML;
- Preservar ordem e associação entre os elementos;
- Manipulação programática, ou seja, ser fácil para criação e manipulação por programas;
- Ser representada em XML;
- Uma linguagem para consulta a documentos XML deve oferecer pelo menos as seguintes operações sob uma perspectiva de banco de dados [Marchiori, 1999]:

- Seleção: que permite incluir no resultado somente os elementos necessários;
- Extração: para extrair elementos específicos de um documento;
- Redução: para remover sub-elementos de um elemento;
- Reestruturação: para construir um novo conjunto de instâncias de elementos para armazenar os dados resultantes da consulta.
- Combinação: que consiste em combinar dois ou mais elementos em um só, expressando um "join".

Apresentaremos a seguir, algumas das principais linguagens propostas para XML. Entretanto, para um melhor aprofundamento sobre o assunto, recomendamos a leitura das referências [Zachary, 2000; W3C, 2002a e Cover, 2002a]. Em [Bonifati, 2000] e [Fernandez et al, 1999] encontram-se comparações entre as principais linguagens de consultas a XML.

2.2.4.1- XML-QL

XML-QL é uma linguagem de consulta para XML, proposta por [Deutsch et al, 1999a] e que utiliza como base os conceitos desenvolvidos em linguagens para os dados semi-estruturados, combinando com a sintaxe XML. Ela suporta construção, transformação e integração de dados XML. É uma linguagem declarativa, relacionalmente completa, suporta o critério de ordenamento (ou não) de um documento, não é rigidamente estruturada (uma vez que o esquema existe juntamente com os dados do documento) e segundo os autores, é simples de aprender e usar.

As consultas em XML-QL se baseiam em *padrões de elementos e "path expressions"* para buscar dados em um documento XML, podendo tanto recuperar dados XML quanto construir novos documentos. Utiliza também variáveis e modelos, os quais mostram como a saída de dados XML é construída. O formato geral da consulta é `WHERE seleção IN origem CONSTRUCT [resultado]`, onde `seleção` são os dados a serem buscados nos documentos XML, `origem` é o endereço dos documentos, e `resultado` é o documento XML construído como resultado da consulta [Abiteboul *et al,* 2000; Deutsch et al, 1999a; Deutsch *et al*, 1999b].

2.2.4.2 - Lorel

Lorel (Lore language) é a linguagem de consulta do SGBD Lore [Abiteboul *et al*, 2000; Abiteboul, 1997; Abiteboul *et al*, 1997]. desenvolvida na Universidade de Stanford e destinado ao gerenciamento de dados semi-estruturados. Utiliza como base a OQL, com modificações e extensões para suportar dados semi-estruturados, sendo que, originalmente, baseava-se em um modelo de dados chamado Object Exchange Model – OEM, que serve como base para várias outras linguagens. Hoje, a linguagem trabalha com XML [Goldman, 1999]. Suas características principais são o uso de coerção em predicados de seleção, a flexibilidade na especificação de expressões de caminho e a possibilidade de atualização de dados. Permite também o uso de variáveis. [Abiteboul *et al*, 1997].

2.2.4.3 - XQL

XQL (XML Query Language) é uma linguagem do paradigma funcional proposta para estender os padrões da XSL (vista anteriormente que é a linguagem de definição de estilos para XML), sendo adequada portanto, para transformações profundas nos documentos [Robie, 1998].

A sua sintaxe é semelhante à sintaxe utilizada para navegar em diretórios de um sistema operacional, baseando-se na hierarquia definida no documento XML. Tomando como exemplo o documento XML da página 28; uma consulta para se obter todos os elementos que são "destino", é escrita apenas como *destino*. Para se obter as cidades de todos os destinos, basta escrever *destino/cidade.* Pode-se citar alguns de seus recursos:

- Expressões booleanas;
- Equivalência;
- Métodos para manipulação de coleções;
- Indexação em coleções;
- Métodos de agregação.

Algumas similaridades funcionais entre XML-QL e XQL são apontadas em [Schach *et al*, 1998]:

- Ambas as linguagens são estruturadas em blocos e orientadas a modelos;
- Retornam estruturas na forma de árvores ou grafos;
- Criam novos elementos de saída;
- Consultam XML;
- Recuperam dados de uma fonte e constroem uma nova fonte de dados a partir dos dados recuperados.

Entretanto a linguagem XML-QL possui várias características que não são suportadas pela XQL, como as variáveis de consulta, OIDs, integração de dados de múltiplas fontes XML, "joins" e formatação do documento resultante da consulta.

2.2.4.4 – XQuery

Mais recentemente, a W3C resolveu propor uma linguagem padrão – XQuery. A XQuery é derivada de uma linguagem chamada Quilt [Chamberlin *et al*, 2000], a qual tem características de outras linguagens de consultas como:

- XPath [W3C, 2001d]: Sintaxe de *path expression,* ou seja, expressões de caminho*;*
- XML-QL [Deutsch *et al*, 1999a]: Noção de *binding* de variáveis para criar novas estruturas;
- OQL [Cattell *et al*, 1996] Noção de uma linguagem funcional composta de diferentes tipos de expressões;
- Influências das linguagens Lorel e YALT [Cluet *et al*, 1999].

Dois importantes aspectos de XQuery é que ela é **funcional,** permitindo vários tipos de **expressões** aninhadas e combinadas com grande generalidade, e **fortemente tipada**, na qual os operandos de várias expressões, operadores e funções devem estar de acordo com os tipos designados [W3C, 2001c]. Uma outra cracterística importante da XQuery é a presença da expressão FLWR (pronuncia-se "*flower*") construída com as cláusulas FOR, LET, WHERE e RETURN. As cláusulas ***for-let*** fazem as variáveis reiterarem sobre o resultado de uma expressão ou sobre variáveis para expressões arbitárias, a cláusula ***where*** permite especificar restrições nas variáveis e a cláusula ***return*** pode construir novos elementos XML como saída da consulta.

2.3 - Ontologias

No capitulo 1 foi apresentada a introdução deste trabalho, na qual explanamos a motivação e os objetivos da nossa proposta. Citamos no referido capítulo que utilizamos a abordagem de ontologias no nosso projeto. Para um maior esclarecimento, nesta seção, apresentamos alguns aspectos conceituais sobre as ontologias e suas aplicações de maneira geral, tentando também contextualizá-las na área de banco de dados. Na subseção 2.3.2, apresentamos a relação entre ontologias e dados semi-estruturados, motivando o uso destas na solução do problema da heterogeneidade semântica para dados XML.

O termo "***Ontologia***" já é conhecido e aplicado há bastante tempo nas áreas da Filosofia e da Epistemologia, significando, respectivamente, um "*sujeito de existência*" ("uma contabilização sistemática da Existência") e um "*conhecimento e saber*". Na década de 90, esse conceito passou a ser utilizado na Ciência da Computação, mais especificamente na área de Inteligência Artificial, para descrever conceitos e relacionamentos utilizados por um agente ou uma comunidade de agentes [Uschold, 1993]. Nesta mesma área, segundo [Gruber, 1993], uma ontologia é definida como "*uma especificação de uma conceitualização*". [Guarino, 1998] amplia esta definição: "*uma ontologia é uma teoria lógica que corresponde ao significado intencional de um vocabulário formal".*

Em [Mello et al, 2000], do ponto de vista de BD, uma ontologia é "*uma especificação parcial de um domínio ou meta-domínio, descrevendo entidades, relações entre eles e regras de integridade*". Desta forma, uma ontologia pode ser considerada como um **modelo conceitual** [Guarino, 1998]. Apesar disto, dentro do contexto específico de dados semi-estruturados, tem-se uma sutil diferença entre uma ontologia e um modelo conceitual. Um modelo conceitual descreve, dentre outras coisas, a estrutura dos dados do BD em um alto nível de abstração. Uma ontologia, tipicamente, contém um vocabulário de conceitos ou classes, estrutura de árvore, relacionamentos entre conceitos, atributos de conceitos e um conjunto de axiomas que definem as regras do negócio [Erdmann, 2000]. Portanto, uma ontologia não representa a estrutura das fontes de dados associadas a ela, apenas propõe uma estrutura de consenso para conceitos e relações que são úteis para grupos de especialistas.

Em [Hwang, 2000] são apontadas características fundamentais que devem ser consideradas na construção de uma ontologia:

- Aberta e dinâmica: deve ser capaz de adaptar-se a mudanças e aprimoramentos no domínio associado, em relação à estrutura e comportamento do domínio. Idealmente, essa evolução deve ser a mais automatizada possível;
- Escalável e interoperável: uma ontologia deve ser facilmente escalável para um amplo domínio e adaptável a novos requisitos. Deve ser possível integrar múltiplas ontologias em uma única ontologia. Essa característica exige que a ontologia seja simples e clara;
- De fácil manutenção: a sua manutenção deve ser fácil. Se sua definição é simples e clara, mais facilmente ela pode ser mais compreendida por especialistas humanos;
- Semanticamente consistente: deve manter conceitos e relacionamentos coerentes com o domínio;
- Independente de contexto: uma ontologia não deve conter termos muito específicos em um certo contexto. Isso dificulta a associação da semântica de cada fonte com os conceitos da ontologia e a integração de ontologias.

O problema de desenvolvimento de ontologias tornou-se amplamente estudado. Conseqüentemente, um número significativo de metodologias tem sido proposto. Uma análise comparativa destas metodologias pode ser encontrada em [Jones *et al,* 1998] e em [Gómez-Pérez *et al*, 2000]. Recomendamos também, a leitura de [Noy, 2001], que apresenta um guia básico de desenvolvimento de uma ontologia e [Pulvermacher *et al*, 2000], que apresenta um processo de desenvolvimento de uma ontologia de um domínio espacial.

2.3.1 - Classificação

[Guarino, 1998] classifica as ontologias em quatro tipos (Figura 2.3), de acordo com sua dependência em relação a uma tarefa específica ou a um ponto de vista:

- Ontologias de Alto Nível ou Nível Superior: descrevem conceitos bem gerais, como espaço, tempo, objeto, ação, etc. Estes são independentes de um problema particular ou domínio.
- Ontologias de Domínio: descrevem um vocabulário relacionado a um domínio genérico (como automóveis, conferências).
- Ontologias de Tarefas: descrevem uma tarefa ou uma atividade (como venda, leitura de um artigo).
- Ontologias de Aplicação: descrevem conceitos que dependem tanto de um domínio específico como de uma tarefa específica, e geralmente são especializações de ambos (como avaliação de um artigo).

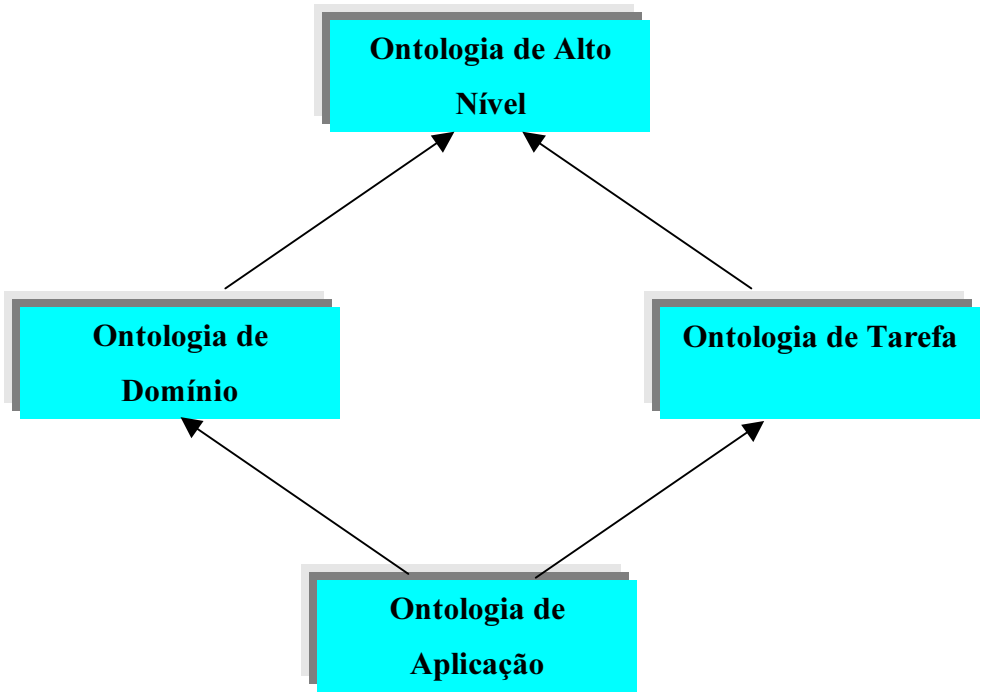

Figura 2.3: Tipos de ontologias e seus relacionamentos de acordo com sua dependência em relação a uma tarefa específica ou a um ponto de vista [Guarino, 1998].

Outra distinção ente ontologias feita por [Guarino, 1998] é entre ontologias refinadas e não-refinadas, ou ainda, *off-line* e *on-lin*e. Uma ontologia não-refinada tem um número mínimo de axiomas, seu objetivo é ser compartilhada por usuários que concordem sobre uma determinada visão de mundo. Uma ontologia refinada tem um grande número de axiomas e precisa de uma linguagem de alta expressividade. Ontologias não-refinadas têm mais chance de serem compartilhadas e deveriam ser usadas *on-line* para dar suporte à funcionalidade de sistemas de informação. As ontologias refinadas deveriam ser usadas *off-line* apenas para referência.

2.3.2 - Ontologias e dados semi-estruturados

O uso de ontologias em sistemas de informação tem por finalidade a manutenção de conceitos (em um alto nível de abstração e de modo específico) para aplicações, banco de dados e interfaces com o usuário.

Na área de BD, ontologias têm sido aplicadas principalmente em BDs heterogêneos e "*Data Warehouses"* como modelos conceituais globais, resultantes de uma concordância na definição de entidades e relacionamentos, integrando informações [Guarino, 1998; Wiederhold, 1994].

Considerando os dados semi-estruturados, ontologias são importantes no sentido de dar **suporte semântico** para o acesso a determinadas informações presentes em um conjunto de fontes destes dados. Tendo-se como grande vantagem, uma interpretação semântica unificada de múltiplas representações de dados semi-estruturados para um mesmo domínio.

Tomando-se como exemplo documentos XML, é importante salientar que com esta representação de dados semi-estruturados, estamos especificando a estrutura de documentos e suas dimensões sintáticas. A estrutura pode também representar propriedades semânticas (devido à flexibilidade do uso de quaisquer "tags"), mas não está claro como isto pode ser usado para aplicações de outros propósitos. Para permitir uma consulta com interpretação semântica, os documentos XML devem ser complementados com um modelo conceitual que adequadamente descreva as semânticas das suas "tags" [Andersen, 2001]. Este papel não pode ser feito pelas DTDs ou XML Schemas. Mesmo estes representando uma visão esquemática

dos documentos, eles apenas suportam as estruturas desses documentos e não definem um modelo de um domínio que seria necessário para as semânticas e os relacionamentos entre os termos [Erdmann, 2000]. [Cui *et al*, 2001] afirmam que o problema da heterogeneidade semântica ainda existirá em um mundo onde todos os dados são trocados usando XML estruturados de acordo com as especificações de esquemas padrões (DTDs ou XML Schemas). Associado a isto, os autores afirmam também que a solução para o problema é equipar os sistemas de software, heterogêneos e autônomos, com a habilidade de trocar informações de uma maneira semanticamente consistente.

Uma ontologia, tipicamente, contém um vocabulário de conceitos ou classes, relacionamentos entre conceitos, atributos de conceitos e um conjunto de axiomas que definem as regras do negócio [Erdmann, 2000], servindo como uma solução para a recuperação semântica de informações nas fontes de dados semi-estruturados. A ontologia oferece também uma interpretação semântica unificada, resolvendo o problema de múltiplas representações dos dados semi-estruturados para um mesmo domínio.

2.3.3 – Linguagens para ontologias

Juntamente com as propostas metodológicas para desenvolvimento de ontologias, têm sido criados também, editores e propostas de linguagens para especificação das ontologias. Em [Corcho, 2000; Gómez-Pérez et al, 2000 e Gómez-Pérez, 2002] encontra-se um estudo sobre as principais linguagens e comparações entre elas, envolvendo características como: conceitos, taxonomias, relações, funções, axiomas e instâncias.

Algumas linguagens já têm sido usadas amplamente na comunidade de ontologias, podendo assim, ser classificadas como tradicionais. São elas: Ontolingua [Farquhar *et al*, 1996], OKBC [Chaudhri *et al*, 1997], OCML [Motta, 1999], Flogic [Kifer *et al*, 1995] e LOOM [MacGregor, 1991].

Nos últimos anos, foram criadas algumas linguagens apropriadas para a *web*, como XML, RDF [Lassila, 1999] e RDF(s) [Brickley, 1999]. Com base nelas, surgiram outras linguagens para especificação de ontologias no contexto *web*.

Apresentamos a seguir, uma breve introdução de algumas destas linguagens. Enfatizamos no entanto, a linguagem DAML+OIL, a qual utilizamos na nossa proposta. Porém, para maior conhecimento, recomendamos a leitura das citadas referências.

- XML-based Ontology Exchange language (XOL) [Karp *et al*,1999]. Criada por uma comunidade de bio-informática para a troca de definições de ontologias dentro deste domínio. XOL tem como base a linguagem Ontolingua e OML [Kent, 1998]. Seu principal objetivo não é especificar ontologias, mas sim, servir como intermediária para transferir ontologias entre as diferentes fontes de dados, aplicações ou ferramentas de desenvolvimento.
- Simple HTML Ontology Extension (SHOE) [Luke, 2000]. Criada na Universidade de Maryland como uma extensão do HTML e agora adaptada sintaticamente para XML, foi usada para desenvolver OML, anotando conhecimento semântico nas páginas HTML ou em outros documentos da *web* para proporcionar a leitura pelas máquinas (agentes).
- Ontology Markup language (OML). Foi desenvolvida na universidade de Washington e é baseada na linguagem SHOE. Existem quatro diferentes níveis de OML: i) *Core*, relacionado aos aspectos lógicos da linguagem e incluída nas demais camadas. ii) *Simple OML*, que mapeia diretamente para RDF(s), iii) *Abbreviated OML*, inclui características de grafos conceituais. iv) *Standard OML*, a versão mais expressiva de OML.
- Resource Description Framework (RDF) e RDF Schema. RDF foi desenvolvido pela W3C com o propósito de descrever recursos *web*. Permite especificações sobre a semântica de dados XML de forma padronizada e interoperável. Provê também mecanismos para representar serviços, processos e modelos de negócios. Porém não suporta recursos para a especificação de relacionamentos entre atributos e seus elementos. Este papel é feito pelo RDF(S), que oferece primitivas para definir modelos de conhecimento que são próximos as abordagens baseadas em *frames*.
- Ontology Interchange Language (OIL) [Horrocks *et al*, 2000]. Desenvolvida no projeto *OntoKnowledge*, permite a interoperabilidade semântica entre as fontes web e é baseada em XOL, RDF(S) e OKBC. Provê a modelagem de primitivas geralmente usadas nas abordagens baseadas em *frames* para engenharia ontológica (conceitos, toxonomias, relações, etc) e também provê semântica formal e suporte de raciocínio encontrados nas abordagens de lógica de descrição (*description logic*). OIL possui as seguintes camadas:

- *Core*: a qual agrupa todas as primitivas que têm mapeamento direto com RDF(S).
- *Standard:* é o modelo completo de OIL, usando mais primitivas do que os definidos em RDF(S).
- *Instance:* adiciona instâncias e papéis para o modelo anterior. iv) *Heavy*, apropriada para futuras extensões de OIL.

- DARPA Agent Markup Language + OIL (DAML+OIL) [Horrocks, 2001]. Foi desenvolvida por um comitê de junção do *US-based Academic* e da União Européia (*IST- Information Society Technologies*) no contexto de DAML, um projeto do DARPA para permitir interoperabilidade semântica em XML. Uma base de conhecimento DAML+OIL é uma coleção de triplas RDF. DAML+OIL é fortemente relacionada à linguagem OIL, seguindo seus mesmos objetivos, além disto, substitui a DAML-ONT, uma especificação inicial a qual também se baseava na linguagem OIL. Uma ontologia DAML+OIL consiste de zero ou mais *headers*, seguidos de zero ou mais *classes, elements, property elements e instances.* Existem muitos editores que a suportam, por exemplo: OILEd, OntoEdit, Protege e WebODE. Nas comparações feitas por [Gómez-Pérez, 2002] entre as linguagens de definições ontológicas para *web*, pôde-se constatar que DAML+OIL suporta todas as características inerentes à taxonomia (subclasse de, decomposições exaustivas, decomposições de disjunção e não subclasse de), todas características de conceitos (partições, documentação; atributos de instância, atributos de classe, escopo local e global, restrições de tipos e restrições de cardinalidade) exceto valores *default*, algumas características de relações e funções (relações/funções n-árias e restrições de tipos), uma característica de axioma (lógica de primeira ordem) e por fim, suporta todas características de instâncias (instâncias de conceito, fatos e *claims*).

A figura 2.4 mostra a relação entre as linguagens citadas no contexto da *web* semântica.

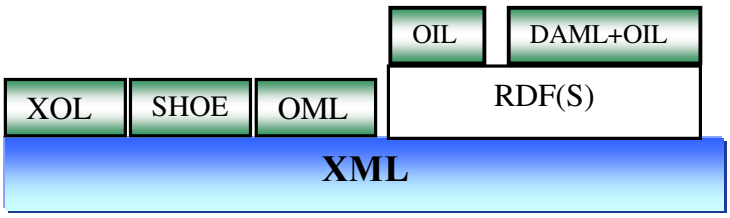

Figura 2.4: Pilha das linguagens na *web* semântica [Gómez-Pérez, 2002].

2.4 – Trabalhos relacionados

Na literatura, existem diversos trabalhos relacionados à integração de dados. Projetos clássicos serviram para nortear as atuais pesquisas, entre os principais podemos citar o TSIMMIS [Chawathe, 1994] e o HERMES [Subrahmanian *et al*, 1995]. Atualmente, existem várias propostas para a integração de dados no contexto *web* que utilizam a abordagem de mediadores e que visam tratar as questões de interoperabilidade semântica. Entre os principais projetos, destacamos:

- Observer [Mena *et al*, 2000]. Sistema de informação global que utiliza ontologias para o processamento de consultas a fonte de dados heterogêneos semi-estruturados e estruturados, criado na Universidade Politécnica de Madrid, Espanha. Possui um gerenciador de relacionamentos entre suas múltiplas ontologias de domínios independentes, que podem estar ou não relacionadas. As ontologias descrevem cada repositório de dados (consideradas neste contexto, como metadados) usando *Description Logics* (DLs), que são então traduzidas para as linguagens de consultas dos repositórios de dados. Como núcleo da sua arquitetura, possui o processador de consultas. Suporta dois tipos de ontologias: alto nível e domínio. Estas provêm suporte semântico para o processamento de consultas e para o esquema conceitual. Observer não considera a substituição de um termo da consulta por uma combinação de *hyponym* e *hypernym*.

- MOMIS (Mediator envirOnment for Multiple Information Sources) [Bergamaschi *et al*, 1999]. Ambiente para integração de fonte de dados estruturados e semi-estruturados desenvolvido na Universidade de Modena, Itália. A sua arquitetura é composta por um *Thesaurus,* que serve como uma ontologia compartilhada, descrito em ODL_I^3, uma linguagem orientada a objetos com um DL subjacente; *wrappers,* responsáveis por traduzir o esquema da fonte local em ODL_I^3 e vive-versa; módulo mediador e processador de consultas, responsáveis por combinar, integrar e refinar o ODL_I^3 e gerar consultas OQL_I^3 para os wrappers. [Momis, 2002].
- Ontobroker [Erdmann, 1999]. Ferramenta de anotação e consulta baseada em ontologias, da Universidade de Karlsruhe, Alemanha, que processa documentos HTML e XML, fazendo uma recuperação inteligente da informação. Suporta ontologia de domínio, descrita em Frame Logic [Kifer *et al*, 1995], a qual provê suporte semântico para o esquema conceitual, anotação de conhecimento e para a inferência de conhecimento. Tem na sua arquitetura: uma máquina de consulta, que recebe as consultas e devolve os resultados para os usuários; um BD, que contém a base de conhecimento; um agente de informação, responsável pela coleta de dados nos diferentes tipos de documentos (HTML anotados com HTML-A, XML, RDF) e uma máquina de inferência, que utiliza regras de inferências sobre as ontologias para derivar novos conhecimentos.

A maioria dos trabalhos encontrados na literatura utiliza as linguagens baseadas em lógica, como *Description Logics* ou *F-Logic*, tornando-os limitados ao poder expressivo deste tipo de linguagens. Além disto, o projeto Observer está apto a tratar apenas com bancos de dados relacionais e bancos de arquivos extensos. O projeto Ontobroker foca o uso de ontologias para anotações de conhecimentos nos documentos *web* e apresenta os resultados também em XML, mas não acessa fontes estruturadas. Além disto, necessita gerar DTDs das ontologias descritas em *F-Logic* para associá-las com XML. Na nossa proposta as ontologias são descritas em DAML+OIL, que é baseada em XML, portanto, não necessita deste processo. O projeto MOMIS expressa o seu esquema global em XML, criado a partir de várias técnicas juntamente com um *Thesaurus* comum entre os esquemas. Porém, acreditamos que ao utilizar uma linguagem que suporte definições semânticas, como DAML+OIL, provemos maior interoperabilidade semântica e flexibilidade para estender agentes inteligentes, atendendo aos requisitos da Web Semântica.

Capítulo 3

Um integrador semântico de dados na web

Como vimos no capítulo anterior, solucionar o problema da integração de dados ainda requer muitas pesquisas, principalmente devido à sua complexidade e aos novos paradigmas provindos da *web*. Atualmente, existe um amplo campo de pesquisa no que tange as arquiteturas de integração para a composição de ferramentas no ambiente *web.*

Em especial, no sentido de fazer com que as ferramentas atendam às especificações de interoperabilidade e às questões de gerenciamento da transformação e apresentação dos dados.

São notórias as aceitações da linguagem XML como padrão para troca de dados na *web* e o uso de ontologias no suporte a buscas por informações de caráter semântico a dados semi-estruturados, especificamente a documentos XML, e o seu poder de incremento de inteligência neste processo de consulta. É a partir destes pontos principais, que propomos um integrador semântico de dados na *web*, cujo objetivo é prover consultas aos dados, e não realizar atualizações nas fontes.

Na seção 3.1, apresentamos uma proposta de arquitetura de integração de dados na *web*. A seção 3.2 detalha o funcionamento dos componentes desta arquitetura. Na seção 3.3 expomos o levantamento de requisitos realizado a partir do estudo e da análise de algumas aplicações de integração de dados e dos problemas e desafios gerais apresentados na análise do domínio do problema, apresentados no capítulo 2. A seção 3.4 mostra o diagrama de *use case* gerado para o processo de realizar uma consulta do usuário. A seção 3.5 mostra um modelo conceitual do integrador de dados que está sendo proposto, o qual se baseia na abordagem de mediadores, vista no capítulo anterior.

3.1 Arquitetura proposta

O principal propósito de nossa arquitetura é prover uma interface unificada a consultas semânticas em fontes de dados heterogêneas (dados estruturados e semi-estruturados), usando a abordagem de ontologias. A figura 3.0 mostra esta arquitetura.

Nossa arquitetura é um sistema de integração de dados baseado em mediador (abordagem virtual) que usa **ontologia** como o esquema comum para sobrepor a heterogeneidade estrutural das fontes de dados. Este esquema comum é descrito na linguagem DAML+OIL e suporta várias ontologias, relacionadas ou não. DAML+OIL é uma linguagem de marcação semântica para recursos da *Web* desenvolvida como uma extensão de XML e RDF [Lassila, 1999; Brickley, 1999], vista no capítulo anterior, na seção 2.3.3.

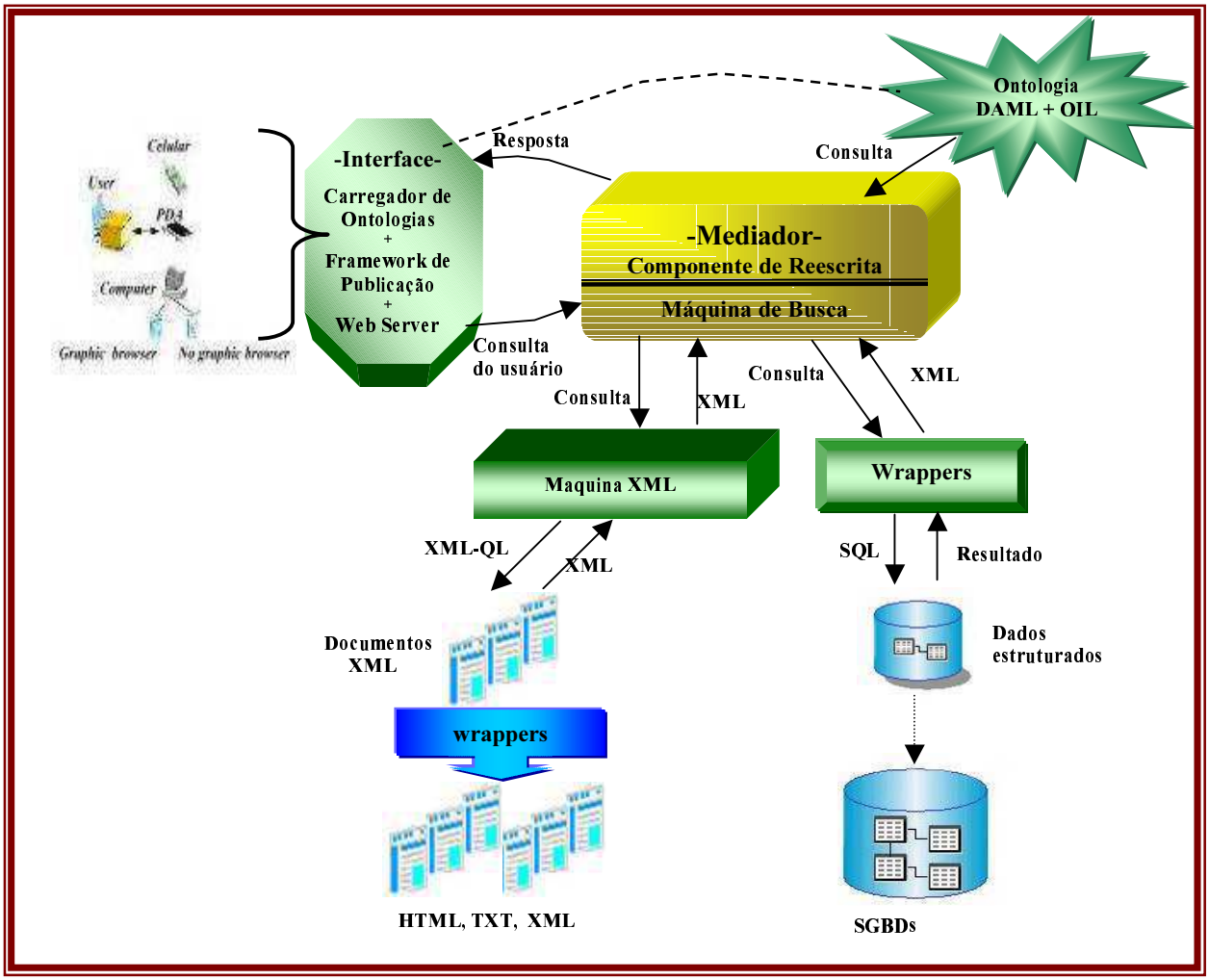

Figura 3.0: Arquitetura proposta.

Os principais componentes da arquitetura são:

- Os **serviços de interface com o usuário**, que são compostos pelo carregador da ontologia, do framework de publicação (publishing framework) [McLaughlihn, 2000] e por um servidor *web.* Estes serviços proporcionam flexibilidade por interagirem com os usuários em múltiplos dispositivos web, representando as ontologias graficamente para posterior navegação.
  - **O carregador de ontologias** é responsável por buscar informações sobre cada ontologia selecionada pelo usuário e montá-la no formato XML.
  - **O *framework* de publicação** é responsável por automaticamente transformar, converter e publicar os documentos XML nos dispositivos web. Para isso, integra algumas ferramentas, como processadores e *"parsers"* de algumas linguagens e separa conteúdo, lógica e estilo para realizar com maior eficiência este processo.
  - **O servidor web** dá suporte essencial de disponibilidade dos componentes da interface para a web.
- **O mediador**, cuja função principal é manter uma visão de esquema integrado (as ontologias) sobre os dados distribuídos nas múltiplas fontes e integrar todos resultados provenientes em formato XML para uma posterior transformação feita pelo *framework* de publicação. O mediador é composto por dois componentes: a máquina de busca e o componente de reescrita de consulta.
  - O componente de reescrita é responsável por estruturar e enriquecer semanticamente a consulta de acordo com conhecimentos extraídos da ontologia escolhida. Por exemplo, o componente pode identificar os relacionamentos hierárquicos das classes da ontologia e a partir disso, reescrever a consulta original adicionando novos elementos correspondentes às subclasses.
  - A máquina de Busca é responsável por receber a consulta do componente de reescrita e distribuí-la entre os *wrappers (*no caso de dados estruturados) e chamar a máquina processadora de consultas a documentos XML (no caso de dados semi-estruturados).

- Os ***wrappers (ou tradutores)*** são específicos para cada fonte estruturada subjacente e traduzem as consultas de aplicação genérica em consultas específicas de cada fonte. Para este processo, os *wrappers* fazem o mapeamento de esquemas através da tabela de correspondências (sintáticas) entre o seu esquema local e o esquema global do sistema (a ontologia). Os ***wrappers*** também traduzem os resultados das consultas realizadas em XML, retornando-os ao módulo mediador.
- A **máquina de consulta a xml** é responsável por processar consultas em fontes semi-estruturadas, ou seja, em documentos xml, retornando o resultado também no formato xml.

As fontes de dados heterogêneas podem ser desde fontes de dados estruturados, como banco de dados relacionais ou orientado a objetos, ou dados semi-estruturados como páginas HTML, documentos textos e XML. Neste trabalho, os dados semi-estruturados são tratados como documentos XML, portanto, é requerido que os repositórios de dados semi-estruturados exportem seus dados no formato XML, estruturados sintaticamente de acordo com alguma ontologia do sistema.

Entretanto, tal requisito supracitado para os dados semi-estruturados não se aplica ao tratamento dos dados estruturados. Isto se deve principalmente às características de sua estrutura: regular; com poucas mudanças, ou seja, fracamente evolutiva; rigidamente definida a priori; prescritiva, etc. *Wrappers* são construídos para tratar cada fonte estruturada. Estes fazem o trabalho de mapeamento de esquemas conceituais, tradução das linguagens, consultas e modelagem dos resultados obtidos em XML.

3.2 Funcionalidade

Baseados na arquitetura descrita, a seguir tem-se alguns passos que demonstram a sua funcionalidade:

1. Os usuários podem acessar o sistema através de quaisquer dispositivos *web* (PDAs, celulares, *browsers*, etc), navegar pela(s) ontologia(s) e então realizar sua consulta em um domínio (ontologia) escolhido. Esta fase é suportada pelos serviços de interface: o **carregador de ontologias**, o **servidor *web*** e o ***framework* de publicação**. No protótipo desenvolvido, cujos detalhes de projeto e implementação serão apresentados no capítulo 4, foi construída inicialmente uma interface para *web browsers.* Portanto, a fim de tornar mais compreensível o funcionamento da arquitetura, apresentamos exemplos baseados neste protótipo. Primeiramente, uma página é apresentada ao usuário com *links* para as ontologias disponíveis no sistema. No momento em que o usuário escolhe uma das ontologias para navegação, o carregador de ontologias monta esta ontologia, definida em DAML + OIL, no formato XML. Logo após, este XML é transformado em HTML e então publicado em estrutura de árvore hierárquica para facilitar a navegação na ontologia. A escolha da estrutura foi feita em função das características das ontologias, vistas no capítulo anterior. A Figura 3.1 mostra exemplos da interface com a ontologia *Documento*. No caso de acesso por outro dispositivo *web*, como um celular por exemplo, o XML gerado pelo carregador de ontologias é transformado em WML (*Wireless Markup Language*) e publicado no dispositivo. Este processo de transformação e publicação é auxiliado pelos serviços do servidor *Web* e do *framework* de publicação (*publish framework*) Cocoon [Cocoon, 2000], este último pode utilizar por exemplo, linguagens como XSL/XSLT para o processo de transformação de formatos.

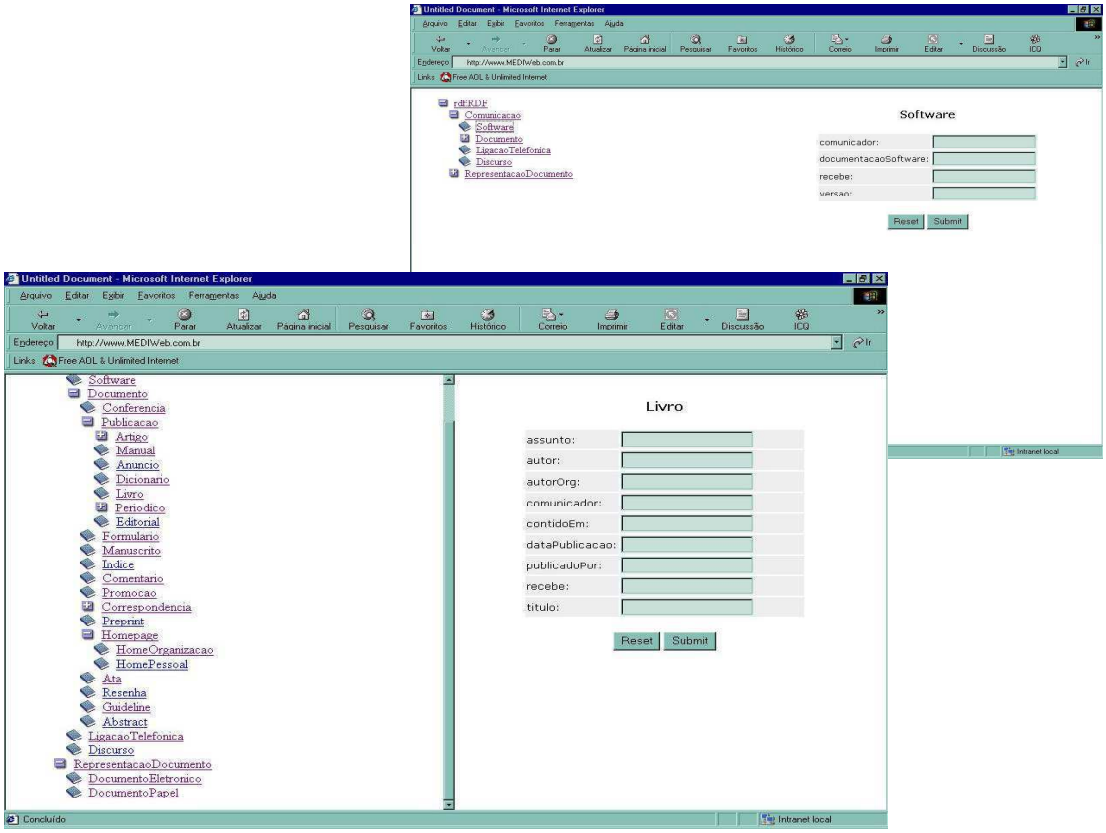

Figura 3.1: Exemplo da Interface com a ontologia Documento.

2. O **mediador** recebe uma consulta genérica, sintaticamente de acordo com os termos ontológicos, ou seja, de acordo com os elementos e suas respectivas propriedades na ontologia. A consulta é então enriquecida pelo **componente de reescrita** da consulta. Este componente utiliza informações adicionais, como por exemplo, regras de inferências e relacionamentos intra e inter ontologias, para enriquecer semanticamente a consulta, porém mantendo a semântica da consulta original feita pelo usuário. Outro componente do mediador – a máquina de busca, mapeia automaticamente a consulta (até então genérica) em consulta para dados semi-estruturados (documentos XML) e em consulta para dados estruturados. Respectivamente, estas são enviadas para a **máquina de consulta a XML** e para os ***Wrappers*** (tradutores).

3. **A máquina de consulta a XML** registra as fontes de todos os documentos XML que pertencem à ontologia escolhida pelo usuário. A linguagem de consulta para XML utilizada em nosso protótipo foi a XMLQL. Porém, outras linguagens poderão ser adotadas, conjuntamente ou não. A escolha da linguagem XMLQL baseou-se principalmente nos seguintes fatores:

- Uma avaliação feita a partir de comparações de XMLQL com outras linguagens (encontradas em [Ives, 2000; Bonifati, 2000 e Bonifati, 2000a]) indica a linguagem como provável padrão a ser adotado pela W3C;
- A linguagem possui características que atendem as necessidades do nosso projeto, como a definição de variáveis, resultado em XML e o poder de expressar *joins;*
- As propostas de linguagens de consulta para XML no período de nossas pesquisas ainda eram incipientes;
- XMLQL foi a primeira proposta pertencente à abordagem de banco de dados, ou seja, "SQL-Like".

Portanto, a máquina de consulta a XML monta a consulta XMLQL com as referências às fontes que deverão ser consultadas e chama o processador XMLQL (Versão 0.9). Este processador retornará resultados em XML, que serão direcionados para a camada de mediação.

4. Paralelamente com a máquina de consulta a XML, os ***wrappers*** presentes no sistema recebem uma consulta sintaticamente coerente com a ontologia. Cada *wrapper* está associado a uma única fonte estruturada e é responsável por mapear a consulta "global" em uma consulta apropriada sintaticamente para a fonte a qual pertence. Este processo de mapeamento é antecedido por outro processo que realiza as devidas correspondências entre os dois esquemas: o esquema ontológico ou global e o esquema da fonte local. O processo de correspondências entre os esquemas é feito analisando um conjunto de regras de correspondências desenvolvidas para cada modelo lógico (relacional, objeto-relacional ou orientado a objetos). Este conjunto auxilia o preenchimento de uma tabela padrão que é utilizada posteriormente no processo de mapeamento feito pelos *wrappers*. Estes processos supracitados serão discutidos no capítulo 4. Com base na sua tabela de mapeamento, cada *wrapper* realiza automaticamente o mapeamento da consulta genérica numa consulta apropriada para sua fonte local. Cada fonte local recebe e processa independentemente sua consulta, retornando para os *wrappers* o conjunto de resultados. Após isto, os *wrappers* formatam os resultados em XML e enviam para a camada do mediador.

5. O **mediador** recebe os resultados que podem ser de dois tipos de fontes (semi-estruturadas e estruturadas) no formato XML, integra estes resultados e chama os serviços da interface para apresentá-los ao usuário em um dispositivo *web*. No protótipo desenvolvido, o *framework* de publicação chama o processador XSL para transformar o documento XML em HTML, para assim, publicá-lo em *web browsers*.

3.3 Requisitos

Nesta seção, encontram-se detalhados os requisitos funcionais e não funcionais de um integrador de dados na *web* e que foram seguidos no decorrer do desenvolvimento do nosso projeto.

3.3.1 Requisitos Funcionais

F1- Deve definir um esquema global para cada domínio de atuação

Como visto no capítulo anterior, um sistema de integração de dados oferece acesso integrado a fontes de dados distribuídas e heterogêneas. Uma das primeiras tarefas a se fazer num sistema deste tipo é definir um esquema único para prover uma visão integrada dos dados distribuídos.

Levando-se em consideração as vantagens do uso de ontologias em sistemas de integração na *web* vistas no capítulo anterior, como por exemplo, o oferecimento de uma interpretação semântica unificada para um domínio, a presente solução deverá ter para cada domínio de atuação, um esquema ontológico assumindo o papel de um esquema global. Estes domínios podem ser relacionados ou podem ser os mais variados possíveis, como por exemplo, um domínio de medicina, de *e-commerce*, de aplicação espacial, de educação, etc. Portanto, deverá ter um esquema global, ou seja, uma ontologia, especificada para cada domínio que o sistema abranger.

F2- Deve disponibilizar graficamente as ontologias para navegação e consulta em dispositivos web.

Atualmente no mundo *web,* enfrentam-se muitos problemas que não eram considerados há alguns anos. Os clientes podem ser navegadores web ("browsers") que suportam HTML, telefones móveis que suportam WML, ou PDAs (Personal Digital Assistent) com alguma linguagem de marcação, como a XHTML (Extensible Hypertext Markup Language). Esta necessidade de lidar com "*thin clients*" na *web* vem crescendo cada vez mais. No que tange a soluções de integração de dados na *web.* O sistema deverá então disponibilizar aos usuários espelhos das ontologias cadastradas no sistema para navegação e conseqüente realização da consulta em diferentes tipos de dispositivos *web.*

F3 - Deve construir consultas coerentes com a ontologia escolhida pelo usuário

Através da interface de um dispositivo *web,* o usuário poderá passar dados para a realização de sua consulta.

O sistema deverá elaborar a consulta requerida pelo usuário com coerência sintática à ontologia que fora escolhida e com um nível de abstração suficiente para facilitar o posterior processo de transformação da consulta para os diversos tipos de fontes de dados.

F4 – Deve reescrever a consulta com enriquecimento semântico

Uma vez criada a consulta genérica a partir dos dados passados pelo usuário, esta deverá conter informações tais como: i) a ontologia a qual pertence, ii) o elemento da ontologia e suas propriedades que estão envolvidos. A partir dessas informações, o sistema deverá enriquecer semanticamente a consulta mediante conhecimentos adquiridos da ontologia. Entretanto, deverá ser mantida a semântica da consulta original.

F5 – Deve reformular a consulta

Como visto no capítulo anterior, devido a sua flexibilidade, XML vem sendo usado atualmente como modelo de dados comum para troca e integração de dados semi-estruturados e estruturados. Considerando que os dados semi-estruturados acessados pelo sistema serão modelados em XML, consultas para este tipo de dado deverão ser formadas a partir da consulta genérica. Do mesmo modo, a consulta deverá ser reformulada para as fontes estruturadas, independentes e heterogêneas, cadastradas no sistema.

F6 – Deve buscar informações em dados semi-estruturados modelados em XML

A consulta já especificada em uma linguagem de consulta para XML, deverá ser processada para a busca de informações nas fontes que pertencem ao domínio da ontologia escolhida para consulta e que estejam sintaticamente baseadas neste esquema ontológico.

F7 – Deve buscar informações em dados estruturados

Para atender a este requisito, outros requisitos são especificamente derivados:

F7.1 - Deverão ser construídos *wrappers* (tradutores)

Cada *wrapper* deve ser construído para uma fonte específica. Sua principal responsabilidade é converter uma determinada consulta coerente com o esquema global na consulta da sua fonte local.

F7.2 – Deve especificar correspondências entre os elementos do esquema global e os elementos dos esquemas locais

Para efetuar o processo de mapeamento da consulta, os *wrappers* necessitam de uma prévia correspondência entre os elementos dos esquemas envolvidos, o esquema global e o seu esquema local. Uma tabela padrão deverá ser preenchida com estas correspondências.

F7.3 – Deve criar regras de correspondências

Considerando a heterogeneidade das fontes envolvidas na integração, mais especificamente o nível lógico delas, e a complexidade do processo de correspondência entre os esquemas; deverão ser criadas regras gerais de correspondências para cada modelo lógico a fim de auxiliar este processo.

F8 – Deve receber os resultados em XML

Tanto na busca a dados estruturados como na busca aos dados semi-estruturados, o resultado deverá ser representado em XML. Isto torna o sistema mais flexível; permitindo troca de dados com outros sistemas, transformações e publicações para diferentes dispositivos *web*.

F9 – Disponibilizar os resultados obtidos em um dispositivo web

O resultado obtido das fontes estruturadas e semi-estruturadas, em XML, deverá ser transformado e publicado em um dispositivo *web* específico (PDA, celular, *browser*). Um *framework* de publicação deverá ser usado para realizar este processo de forma mais eficiente.

3.3.2 Requisitos Não-funcionais

NF1 – Deve ser flexível quanto às linguagens de consulta a XML

A solução deverá ser genérica o suficiente para suportar trocas e/ou adições de diferentes tipos de linguagens de consulta a XML. A justificativa deste requisito está ligada ao fato de que as definições ainda encontram-se imaturas e em fase de muitas pesquisas. Mutações são freqüentes e novas versões surgem com muita freqüência a fim de se chegar aos requisitos desejados de uma linguagem de consulta a XML. Portanto, é desejável que o sistema seja bem flexível às mudanças das linguagens e conseqüentemente de seus processadores.

NF2 – Deve atender à visão da Web Semântica

A Web Semântica representa a evolução da *web* atual. Ela será capaz de prover uma ampla rede de conhecimento, que poderá ser processado por homens e máquinas.

A partir desta convicção, percebe-se que os sistemas desenvolvidos para o ambiente *web* deverão incorporar esta visão, baseando-se em tecnologias que possibilitem a compreensão da informação também por máquinas. Estas tecnologias envolvem principalmente padrões de dados/metadados (XML, RDF, RDFS) e o consenso semântico explícito através de ontologias.

NF3 – Deve ter um bom desempenho

Comparada com a abordagem materializada, a abordagem virtual tem como principal desvantagem o desempenho, pois busca informações diretamente nas fontes de dados locais distribuídas a cada pedido de consulta. Portanto, uma das características mais importantes para um sistema de integração baseado na abordagem virtual, é garantir um bom desempenho. Para tanto, é importante que o sistema tenha acesso apenas às fontes relevantes para cada consulta requerida pelo usuário.

NF4 - Deve ter manutenção facilitada

Tanto as fontes semi-estruturadas quanto as fontes estruturadas que fornecem dados para um sistema de integração são autônomas e dinâmicas. Estas podem sofrer alterações e o sistema deve ter as atualizações necessárias para manter a coerência das respostas para o usuário final. Mesmo requerendo que as fontes semi-estruturadas estejam modeladas esquematicamente de acordo com a ontologia, poderão ser necessárias mudanças, atualizações de novos conceitos nas próprias ontologias. Como também, inclusões de novas fontes estruturadas, ou mesmo as fontes estruturadas existentes podem sofrer alterações de seus esquemas de dados. Portanto, o sistema deve ter essencialmente uma fácil manutenção das ontologias e dos *wrappers*.

NF5 – Usabilidade e disponibilidade

O usuário deverá se sentir confortável em navegar nas ontologias, realizar suas consultas e poder escolher qual dispositivo *web* usar para ter acesso ao sistema. Então, é desejável que o sistema possibilite fácil utilização e disponibilidade de seus recursos.

3.4 – Modelo de *Use Case*

Com o objetivo de explorar melhor os requisitos funcionais e também documentá-los, foi elaborado o modelo de *use case*, onde são descritos os principais eventos de iteração do sistema com um ator (usuário do sistema).

Os *use cases* podem ser categorizados como primários, secundários ou opcionais. Um *use case* primário representa os processos mais comuns e/ou importantes; um *use case* secundário representa processos raros ou de pequena importância e um *use case* do tipo opcional representa os processos que podem não ser considerados [Larman, 1998].

Use cases essenciais são *use cases* expandidos que possuem uma descrição muito breve, sem detalhes de tecnologias e implementações. Ao contrário dos *use cases* reais, os quais descrevem os processos de maneira mais concreta, especificando as tecnologias utilizadas.

A figura 3.2 mostra o modelo de *use cases* expandido para o processo de realização de uma consulta. A tabela 3.0 descreve as características do *use case* RealizarConsulta.

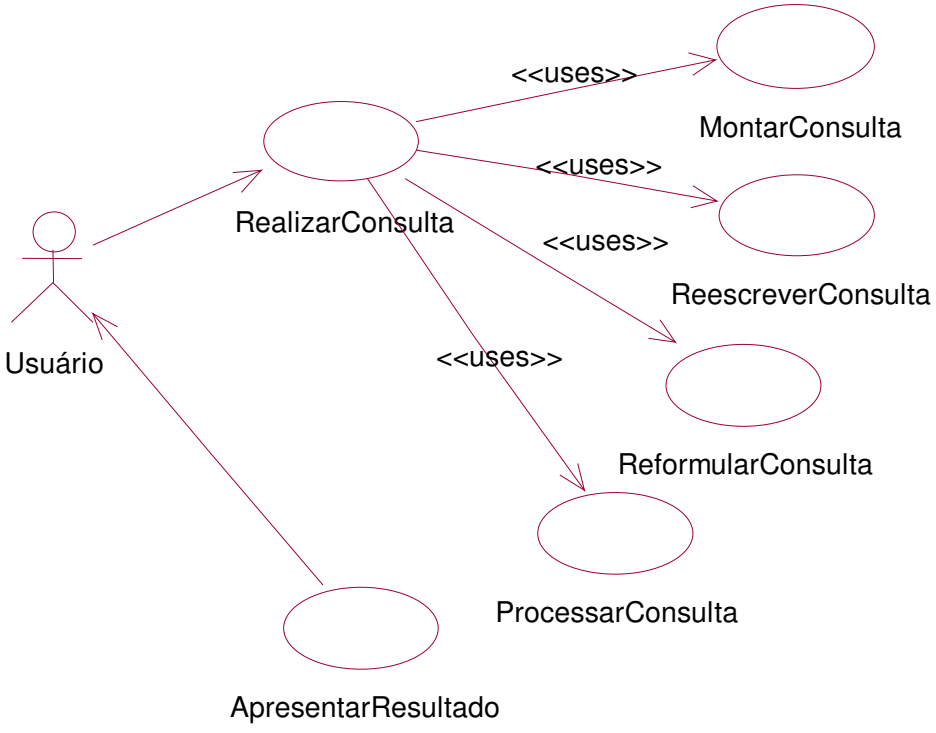

Figura 3.2: Diagrama de *use case* expandido essencial.

Use Case	RealizarConsulta
Ator	Usuário da web
Propósito	Registrar uma consulta ao sistema
Resumo	O usuário acessa o sistema e escolhe uma ontologia para navegar nos seus elementos. Logo após, escolhe um dos elementos da ontologia e preenche um ou mais campos disponíveis na interface com os dados para realizar sua consulta.
Tipo	Primário e Essencial
Referência	Envolve todos os requisitos funcionais do sistema

Tabela 3.0: *Use case* RealizarConsulta

Para efetuar o evento de realizar uma consulta, o sistema precisa executar outros processos imprescindíveis que são transparentes ao usuário. Nas tabelas 3.1, 3.2, 3.3 e 3.4, têm-se as características dos demais u*se cases*:

Use Case	MontarConsulta
Propósito	Montar uma consulta genérica
Resumo	Os dados inseridos pelo usuário e as informações adquiridas sobre a ontologia e os elementos envolvidos são organizados a fim de se montar uma consulta genérica para o sistema.
Tipo	Primário e Essencial
Referência	F3

Tabela 3.1: *Use case* MontarConsulta

Use Case	ReescreverConsulta
Propósito	Reescrever uma consulta, adicionando riqueza semântica
Resumo	A partir da consulta genérica gerada, o sistema obtém informações na ontologia sobre os relacionamentos dos elementos envolvidos na consulta. A partir dessas informações, a consulta é então reescrita de forma a enriquecê-la semanticamente.
Tipo	Primário e Essencial
Referência	F4

Tabela 3.2: *Use case* ReescreverConsulta

Use Case	ReformularConsulta
Propósito	Montar a consulta genérica em uma linguagem específica para fontes estruturadas e semi-estruturadas.
Resumo	A consulta genérica, após ser enriquecida semanticamente, é estruturada sintaticamente de acordo com linguagens específicas de consulta para dados estruturados e para dados semi-estruturados.
Tipo	Primário e Essencial
Referência	F5

Tabela 3.3: *Use case* ReformularConsulta

Use Case	ProcessarConsulta
Propósito	Processar a consulta nas fontes distribuídas
Resumo	A consulta é processada em cada fonte que pertence ao domínio da ontologia escolhida pelo usuário para consulta, seja esta fonte estruturada ou semi-estruturada.
Tipo	Primário e Essencial
Referência	F6 e F7

Tabela 3.4: *Use case* ProcessarConsulta

A tabela 3.5 mostra o *use case* ApresentarResultado:

Use Case	ApresentarResultado
Propósito	Apresentar os resultados da consulta submetida pelo usuário.
Resumo	Depois de processada a consulta nas determinadas fontes distribuídas, todo o resultado é apresentado ao usuário.
Tipo	Primário e Essencial
Referência	F8 e F9

Tabela 3.5: *Use case* ApresentarResultado

3.5 Modelo Conceitual

A figura 3.3 mostra o modelo conceitual do sistema baseado na arquitetura proposta, apresentada na seção 3.1 do presente capítulo.

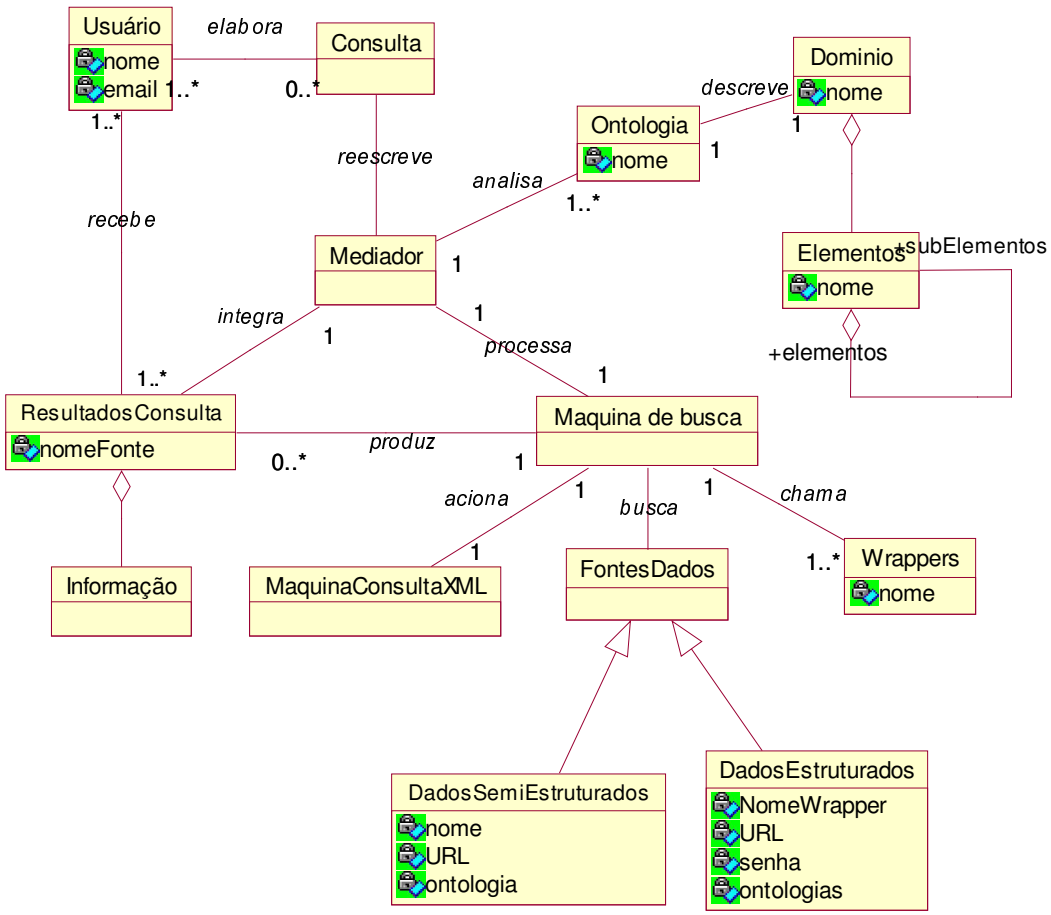

Figura 3.3: Modelo conceitual

Partindo-se de uma visão de alto nível, podemos considerar os principais tipos e relacionamentos descritos no modelo como:

- **Usuário**: Representa qualquer usuário da *web* que utilize o sistema. É o responsável pela elaboração da consulta a ser processada.
- **Consulta**: Representa a consulta feita pelo usuário.
- **Ontologia**: Representa cada ontologia cadastrada no sistema. Uma ontologia é composta por **elementos** relacionados que descrevem um **domínio**.
- **Mediador**: É o componente do sistema de integração responsável por estruturar a consulta recebida, reescrevê-la com base na ontologia escolhida para consulta, distribuí-la através da **máquina de busca** para as fontes heterogêneas e por fim, recolher e integrar os resultados.
- **Máquina de busca**: Componente que pertence à camada de mediação. Auxilia o mediador na busca de informações nas diversas fontes estruturadas e semi-estruturadas, gerenciando os ***wrappers*** e a **máquina de consulta a XML**.
- ***Wrappers***: São tradutores responsáveis pelo mapeamento da consulta gerada pelo sistema de integração numa consulta apropriada para uma fonte estruturada específica.
- **Máquina de Consulta a XML:** Componente responsável por realizar consultas nas fontes semi-estruturadas, ou seja, nos documentos XML.

Capítulo 4

MediWeb: Projeto e implementação

No capítulo três, vimos a análise e a especificação de um integrador semântico de dados na *web.* Foram descritos os requisitos com base na análise do domínio e dos projetos relacionados descritos no capitulo 2. Ademais, foi proposta uma arquitetura pertencente à abordagem virtual, com explanações de alto nível sobre os seus componentes e seu funcionamento.

Neste capítulo, descemos o nível de abstração dos componentes vistos anteriormente, descrevendo-os detalhadamente, chegando assim, ao nível de projeto e implementação da arquitetura proposta. O protótipo foi desenvolvido de forma iterativa e incremental e implementado na linguagem Java [Flanagan, 1997]. A notação usada nos diagramas das diversas fases desse projeto é a *Unified Modeling Language* (UML) [Rumbaugh *et al*, 1999a; Booch *et al*., 1999]. Utilizou-se também a técnica de *refactoring*, a qual tem por finalidade o melhoramento do projeto da solução, realizando modificações no software, sem acréscimo de funcionalidades [Fowler, 1999].

O projeto das classes e interfaces será explorado na seção 4.1, onde destacamos para cada classe, os principais métodos e pormenorizamos suas responsabilidades. Na seção 4.2, discutimos a interface do usuário proposta na nossa arquitetura, mostrando a interface desenvolvida no nosso protótipo. A seção 4.3 contém detalhes do processo de correspondência e mapeamento utilizado pelos *wrappers*. Com o objetivo de tornar válido e também diáfano o funcionamento geral da arquitetura, mostramos na seção 4.4 estudos de caso que exemplificarão a utilização do protótipo desenvolvido.

4.1 Projeto e implementação

Nesta seção, apresentamos um diagrama UML de classes do sistema na figura 4.0 e com base neste, descrevemos a funcionalidade de cada classe e interface.

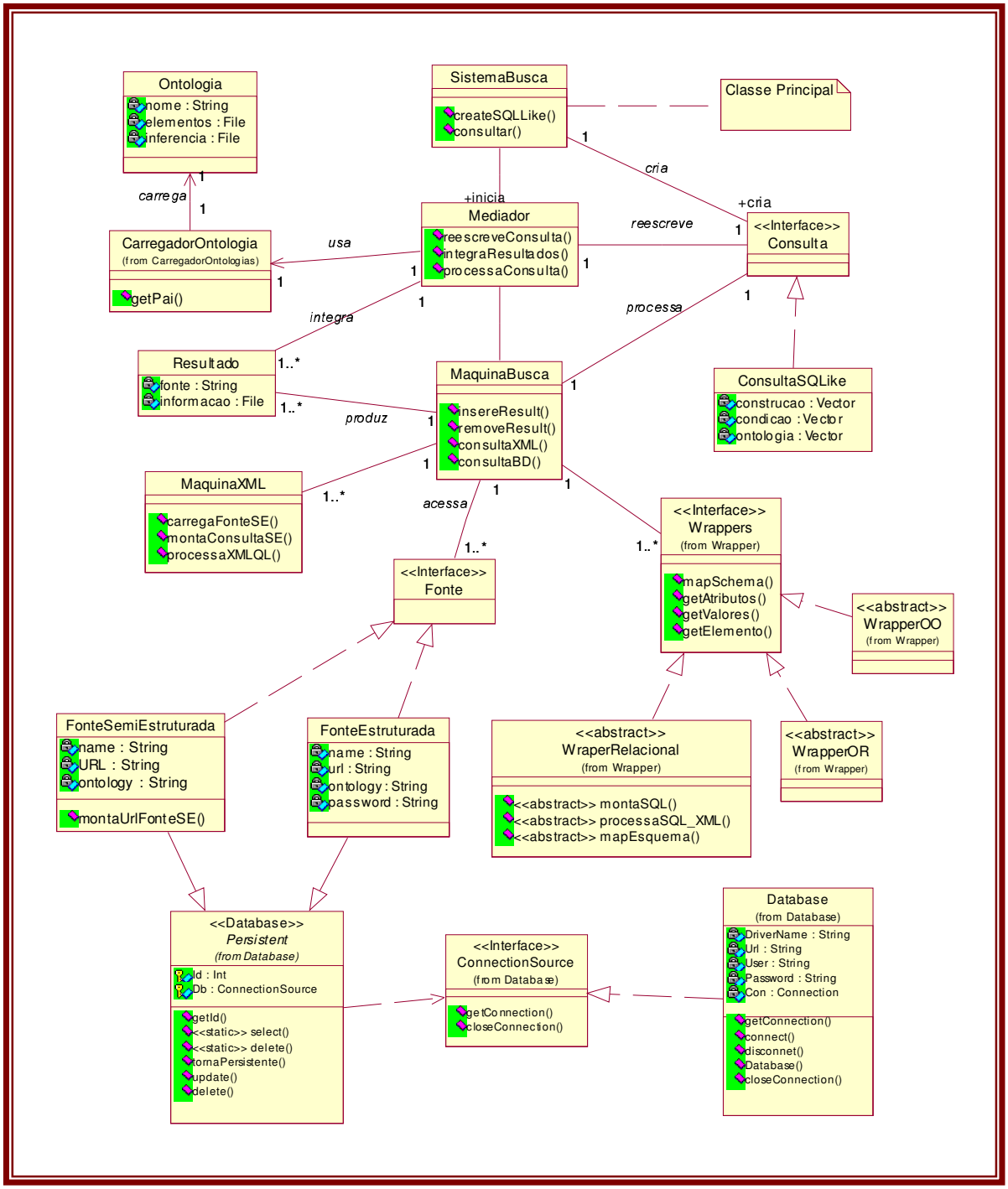

Figura 4.0: Diagrama de classes

Começaremos pela classe principal do sistema, a classe ***SistemaBusca,*** que é responsável por começar o seu funcionamento. O método *createSQLLike()* cria uma instância da classe ***ConsultaSQLike*** com dados passados pelo usuário. Denominamos esta instância criada de consulta genérica, a qual referenciamos no capítulo anterior na seção 3.3. A classe ***ConsultaSQLike*** implementa a interface ***Consulta***. Esta relação provê flexibilidade para mudar ou adicionar qualquer outro tipo de linguagem de consulta. Especialmente quanto se trata das linguagens de consulta para XML, onde podemos ter linguagens de paradigmas bem diferentes. Um exemplo disto é que podemos substituir a linguagem XMLQL, pertencente ao paradigma de banco de dados, pela linguagem XQL, do paradigma funcional ou apenas adicionar esta última sem nenhum impacto na estrutura do projeto do sistema.

Logo depois de criada a consulta, o método *consultar()* instancia um mediador, passando a consulta genérica para ser tratada conforme requerido pelo sistema. A classe ***Mediador*** tem como primeira tarefa reescrever a consulta, depois estruturá-la sintaticamente e enviá-la para as devidas fontes estruturadas e semi-estruturadas e, por fim, recolher os resultados integrando-os no formato XML.

Para a primeira função, a reescrita da consulta, o mediador utiliza o método *reescreveConsulta()*. Este método utiliza uma instância da classe ***CarregadorOntologia***, que possui métodos para navegação nas ontologias. Através da navegação nas ontologias, o método consegue comparar os elementos envolvidos na consulta e portanto, identificar suas relações hierárquicas. A partir disto a consulta é então enriquecida, adicionando alguns elementos "filhos" envolvidos na hierarquia. Por exemplo, suponha que após o usuário utilizar o sistema e realizar sua consulta, tem-se seguinte cenário:

Ontologia: *Documento*

Elemento: *Publicacao*

Propriedade: *titulo*

O método *reescreveConsulta()*, a partir dessas informações, pede para a instância do ***CarregadorOntologia*** caminhar na ontologia *Documento*, identificando os "filhos" da classe Publicacao. Vamos supor que só haja um filho, a subclasse livro. A ontologia define que um livro é uma publicação. Logo, a propriedade *titulo*, pertencente ao elemento *publicacao*, é herdada por *livro*. Conseqüentemente, a consulta é enriquecida adicionando-se outro elemento para a busca de informações. Contudo, a consulta enriquecida ficará semanticamente coerente com a original, atendendo ao requisito funcional F4, visto no capítulo anterior na seção 3.1.

Para realizar a segunda tarefa, o envio da consulta aos dois tipos de fontes, a classe ***Mediador*** instancia a classe ***MaquinaBusca***, que possui dois métodos principais: *consultaXML()* e *consultaBD()*. Ambos métodos recebem um objeto do tipo *Consulta* e enviam, respectivamente, a consulta para as fontes semi-estruturadas e para as fontes estruturadas.

O método *consultaXML()* instancia a classe ***MaquinaXML***, cuja função é chamar os processadores das linguagens de consulta para XML. No protótipo implementado, o processador da linguagem XMLQL é chamado e ativado para processar a consulta nas fontes semi-estruturadas que pertencem à ontologia escolhida pelo usuário. As informações e funcionalidades sobre as fontes semi-estruturadas cadastradas no sistema estão na classe ***FonteSemiEstruturada***.

O método *consultaBD()* tem função semelhante, entretanto, trata da consulta nas fontes estruturadas. Para cumprir sua função, o método instancia objetos da classe ***FonteEstruturada*** que pertencem à ontologia escolhida pelo usuário e, para cada objeto da ***FonteEstruturada*** é instanciado seu w*rapper***.**

O pacote **Wrapper** contém uma interface chamada ***Wrappers***, a qual possui os métodos comuns a todos os *wrappers* que pertencem ao sistema. O principal método definido na interface ***Wrappers*** é o *mapEsquema()*, cuja função é mapear a consulta recebida no esquema global, ou ontológico, para o esquema da sua fonte local através dos dados contidos na tabela de correspondência. Este método é usado pelos *wrappers* concretos das fontes cadastradas no sistema. Na seção 4.3 detalhamos melhor o processo de mapeamento.

Implementando a interface ***Wrappers***, têm-se as classes abstratas ***WrapperRelacional, WrapperOR*** e ***WrapperOO***. Cada uma delas representa respectivamente abstrações do modelo lógico relacional, objeto relacional e orientado a objetos. No nosso protótipo, limitamos a implementação dos *wrappers* para as fontes de dados do modelo relacional. A utilização deste modelo no nosso estudo de caso foi devido à sua ampla utilização nas bases de dados dos sistemas existentes atualmente. A classe abstrata ***WrapperRelacional,*** além de implementar alguns métodos definidos na interface ***Wrappers,*** define mais dois métodos abstratos: *montaSQL()* e o *processaSQL_XML()*. Estes métodos são usados por classes concretas, produzidas para cada fonte estruturada. O método *montaSQL()* é responsável por realizar a estruturação sintática da consulta de acordo com a sua fonte de dados, após esta consulta ser mapeada pelo método *mapEsquema()*. O método *processaSQL_XML()* modela os dados obtidos da fonte em XML.

De maneira geral, quando o método *consultaBD()* recebe a consulta e instancia as devidas fontes estruturadas, ele instancia automaticamente os *wrappers* agregados a cada fonte. A consulta é passada para cada *wrapper* que realiza primeiramente o mapeamento da consulta através da tabela de correspondência, depois faz a estruturação da consulta e por fim a modelagem dos resultados obtidos em XML.

Tanto o método *consultaXML()* quanto o método *consultaDB()* instanciam a classe ***Resultado*** com as informações do resultado da consulta no formato XML. A classe ***Mediador*** portanto, através do método *integraResultados()*, integra o conjunto de resultados em um único resultado, também no formato XML, para ser apresentado ao usuário.

4.2 Interface

Como visto no capítulo anterior, a arquitetura do sistema deve suportar uma interface flexível quanto aos dispositivos *web*. Vimos também que alguns serviços são utilizados para realizar a apresentação da interface: O carregador da ontologia, o *framework* de publicação (*publishing framework*) [McLaughlihn, 2000] e um servidor *web*. Nesta seção, particularizamos e detalhamos a interface desenvolvida no nosso protótipo, mostrando de maneira geral, as principais características do seu funcionamento. A figura 4.1 mostra um exemplo da interface.

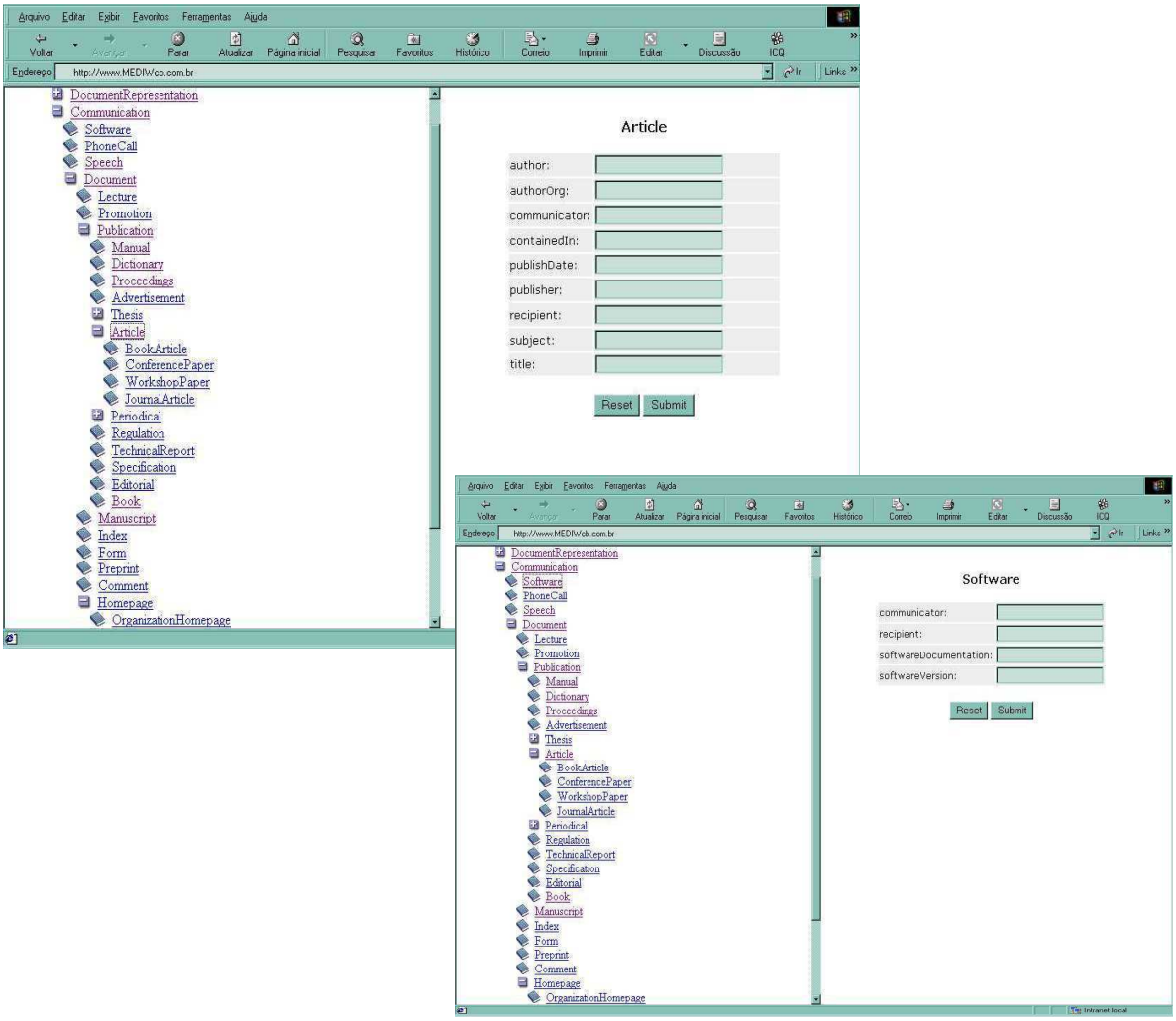

Figura 4.1: Exemplo de interface com a ontologia "Document"

O carregador de ontologias é responsável por acessar a ontologia em DAML + OIL; separar as classes, subclasses e propriedades definidas na ontologia e mapeá-los para a estrutura de elementos e atributos de XML. A partir disto, o documento XML é analisado (*parser*) e é montada então, uma estrutura de árvore para melhor representar a estrutura hierárquica da ontologia.

Na fase de projeto, o carregador de ontologias é representado pelo pacote **CarregadorOntologias.** A figura 4.2 apresenta, de maneira geral, a estrutura de funcionamento deste pacote.

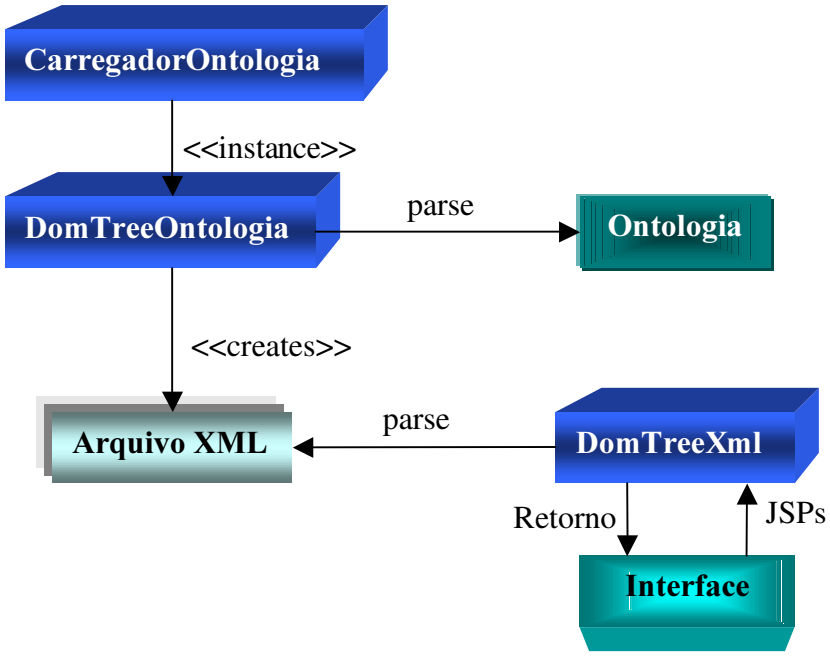

Figura 4.2: Visão geral do funcionamento do pacote CarregadorOntologias

Do exposto, temos as seguintes classes e responsabilidades:

- ***CarregadorOntologia:*** Responsável pela criação da instância *DomTreeOntologia* e pelo controle geral do pacote.
- ***DomTreeOntologia:*** Responsável por fazer o "*parsing*" da ontologia passada como entrada e por armazenar classes requeridas da ontologia em certas estruturas internas para posterior processamento. Os principais métodos são:

  - *getRaizOntologia()*: Recupera a(s) raiz(es) da Ontologia;
  - *getSubclasses(String value):* Retorna todas as subclasses de uma determinada classe da ontologia;
  - getPropriedades(String value): Retorna todas as propriedades de uma determinada classe da ontologia;
  - *performParse(String uri):* Faz o *parser* do arquivo;
  - *walkInTree(Node node):* Realiza o caminhamento recursivo na árvore gerada pelo *parser*;
  - *getDocumentRoot():* Recupera a raiz do documento gerado;

- *writeXml(File f):* Gera um arquivo XML para ser usado na apresentação da árvore hierárquica.

- ***Class DomTreeXml*:** Responsável por fazer o tratamento do arquivo XML gerado pela classe ***DomTreeOntologia***. Entre os principais métodos, estão:

  - *getChildrenElements(String nomeTag):* Retorna todos os elementos "filhos" de um dado elemento da árvore;
  - *getElements():* Retorna todos os elementos da árvore gerada;
  - *getElementAttributes(String field):* Retorna uma coleção de atributos de um dado elemento da árvore;
  - *showTree(Node raiz):* Exibe uma possível coleção de elementos a partir de uma raiz passada como parâmetro;
  - *getEnableExpand(String field):* Informa se um determinado nodo da árvore está expandido ou não. Se o mesmo estiver expandido, significa que existe a possibilidade de visualização de todos os seus filhos.

O *framework* de publicação Cocoon [Cocoon, 2000] é responsável por publicar as informações em XML nos devidos dispositivos. Para fazer tal publicação, o *framework* é auxiliado por um servidor *web.* No nosso protótipo, desenvolvemos a interface para um *browser.* Conseqüentemente, deverá haver uma transformação e publicação do XML em HTML. Em particular, esta transformação é feita pela linguagem XSL. Além do mais, utilizamos páginas JSP (Java Server Pages) para a navegação dinâmica da árvore hierárquica e para a geração dos formulários em HTML próprios para o preenchimento da consulta.

Optamos por usar um *framework* da Apache, o Cocoon2.0 a partir de alguns critérios [McLaughlihn, 2000]:

- Proporciona um conjunto de tecnologias integradas indicadas pela W3C;
- Estabilidade comprovada;
- Suporta uma variedade de "parsers" e processadores para XML;
- Tem código aberto;
- Baseado numa arquitetura popular ("Java Servlet")
- Várias aplicações de produção ("sites") o utilizam .

4.3 O processo de mapeamento

Como visto anteriormente, os *wrappers* são responsáveis pelo processo de mapeamento da consulta no esquema ontológico em uma consulta no esquema interno da sua fonte local. Este processo é automático, porém requer um processo anterior para acertar as correspondências sintáticas entre os dois esquemas.

Uma proposta de solução foi criar uma tabela de correspondências entre os dois esquemas, que deve ser preenchida por um especialista humano devido à complexidade semântica envolvida. O especialista deverá apenas ter conhecimento dos dois esquemas e poderá ser auxiliado por um conjunto de regras básicas para o preenchimento da tabela.

O conjunto de regras de correspondências definido para nosso protótipo foi criado para o modelo relacional. Deixamos para trabalhos futuros o desenvolvimento de regras para os demais modelos lógicos (modelo objeto relacional e orientado a objetos).

4.3.1 Regras de correspondências para o modelo relacional

A seguir, temos as definições das regras de correspondências para cada possibilidade básica de relacionamento no modelo relacional. Exemplos genéricos são utilizados para demonstrar o preenchimento da tabela de mapeamento.

Para todas as regras, utilizamos as seguintes definições:

Entidade **Eq_C1** : Entidade Equivalente à classe C1.

Entidade **Eq_p** : Entidade Equivalente à propriedade p.

FK = Chave estrangeira

PK = Chave primária

null: Nulo

Regra #1: **Quando uma tabela "A", em um determinado esquema, corresponde a uma classe "C1" na ontologia e um atributo "a" desta tabela A, corresponde a uma propriedade "p" da mesma classe da ontologia, o preenchimento da tabela de mapeamento deve seguir o modelo descrito na Tabela 4.0.**

Classe	Propriedade	Tabela	Atributo	Junção
C1	P	Eq_C1	Eq_p	*null*

Tabela 4.0: Tabela de mapeamento da regra #1

Exemplo 1:

Esquema ontológico: Classe *C1* com propriedade *p*

Ex: Classe *C1* = documento e *p* = autor

➔**Consulta: Select * from documento where autor = 'fulano';**

Esquema relacional: Entidade *Eq_C1* com atributo *Eq_p*

Ex: Tabela *Eq_C1* = livro. Ou seja, a tabela correspondente à classe documento é a tabela livro. A mesma lógica aplica-se à propriedade autor: *Eq_p* = autor. Assim, pelo nosso exemplo 1, temos a tabela de mapeamento preenchida (tabela 4.1) e a consulta correspondente na fonte relacional:

Classe	Propriedade	Tabela	Atributo	Junção
documento	Autor	livro	autor	*null*

Tabela 4.1: Tabela de mapeamento do exemplo #1

➔ Consulta: Select * from livro where autor = 'fulano';

Regra #2: **Quando uma tabela "A", em um determinado esquema, corresponde a uma classe "C1" na ontologia; e uma tabela "B" relacionada com a tabela A com multiplicidade (1,*) ou (1,1) corresponde a uma propriedade "p" da mesma classe da ontologia, o preenchimento da tabela de mapeamento deve seguir o modelo descrito na Tabela 4.2.**

Classe	Propriedade	Tabela	Atributo	Junção
C1	p	Eq_C1 x, Eq_p y	y.atributo	y.FK = x.PK

Tabela 4.2: Tabela de mapeamento da regra #2

Exemplo 2:

Esquema ontológico: Classe *C1* com atributo *p*

Ex: Classe *C1* = documento e *p* = autor

➔ **Consulta: Select * from documento where autor = 'fulano';**

Esquema relacional: Entidade *Eq_C1* com Entidade *Eq_p*

Ex: Tabela *Eq_C1* = livro e Tabela *Eq_p* = autor (1,1) ou (1,*).

A tabela 4.3 mostra a tabela de mapeamento para o exemplo 2:

Classe	Propriedade	Tabela	Atributo	Junção
livro	autor	livro l, autor a	a.nome	a.livro = l.cod

Tabela 4.3: Tabela de mapeamento do exemplo #2

➔ Consulta: Select * from livro l, autor a

where a.nome = 'fulano' AND a.livro = l.cod;

Regra #3:

Quando uma tabela "A", em um determinado esquema, corresponde a uma classe "C1" na ontologia; e uma tabela "B" relacionada com a tabela A com multiplicidade (*,1) ou (1,1), corresponde a uma propriedade "p" da mesma classe da ontologia, o preenchimento da tabela de mapeamento deve seguir o modelo descrito na Tabela 4.4:

Classe	Propriedade	Tabela	Atributo	Junção
C1	p	Eq_C1 x, Eq_p y	y.atributo	x.FK = y.PK

Tabela 4.4: Tabela de mapeamento da regra #3

Exemplo 3:

Esquema ontológico: Classe *C1* com atributo *p*

Ex: Classe C1 = documento e p = autor

➔ Consulta: Select * from documento where autor = ‘fulano’;

Esquema relacional: Entidade *Eq_C1* com Entidade *Eq_p*

Ex: Tabela Eq_C1 = livro e Tabela Eq_p = autor (1,1) ou (*,1).

A tabela 4.5 mostra a tabela de mapeamento para o exemplo 3:

Classe	Propriedade	Tabela	Atributo	Junção
livro	autor	Livro l, Autor a	a.nome	l.autor = a.cod

Tabela 4.5: Tabela de mapeamento do exemplo #3

➔ Consulta: Select * from livro l, autor a

where a.nome = ‘fulano’ AND l.autor = a.cod;

Regra #4:

Quando uma tabela “A”, em um determinado esquema, corresponde a uma classe “C1” na ontologia; e uma tabela “B”, que pertence a uma propriedade “p” da mesma ontologia, está relacionada com a tabela A através de uma tabela de junção “AB” (devido a um relacionamento (*,*)) , o preenchimento da tabela de mapeamento deve seguir o modelo descrito na Tabela 4.6:

Classe	Propriedade	Tabela	Atributo	Junção
C1	p	Eq_C1 x, Eq_p y, tabjuncao ab	y.atributo	y.PK = ab.FK_p AND ab.FK_x = x.PK

Tabela 4.6: Tabela de mapeamento da regra #4

Exemplo 4:

Esquema ontológico: Classe *C1* com atributo *p*

Ex: Classe *C1* = documento e *p* = autor

➔ **Select * from documento where autor = 'fulano'.**

Esquema relacional: Entidade Eq_C1 com Entidade Eq_p

Ex: Tabela *Eq_C1* = livro; Tabela *Eq_p* = autor e Tabela *ab* = livroAutor

A tabela 4.7 mostra a tabela de mapeamento para o exemplo 4:

Classe	Propriedade	Tabela	Atributo	Junção
documento	autor	Livro l, Autor a, livroAutor la	a.nome	a.cod = la.codAutor AND la.codLivro = l.cod

Tabela 4.7: Tabela de mapeamento o exemplo #4

➔ Select * from livro l, autor a, livroAutor c where a = 'fulano' AND
a.cod = c.codAutor AND c.codLivro = l.cod.

4.4 Estudo de caso

Como forma de deixar mais claro o funcionamento da arquitetura proposta e o processo de mapeamento de esquemas, serão mostrados exemplos práticos e simples em cima do protótipo desenvolvido para algumas regras de correspondência mostradas na seção anterior.

Suponha que o usuário escolha a ontologia *Document* para realizar sua consulta. Internamente, esta ontologia está especificada em DAML+OIL, como mostra o texto a seguir:

```
<rdf:RDF
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns = "http://www.daml.org/2000/10/daml-ont#"
  xmlns:daml = "http://www.daml.org/2000/10/daml-ont#"
  xmlns:base =
"http://www.cs.umd.edu/projects/plus/DAML/onts/base1.0.daml#"
  xmlns:gen =
"http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.daml#"
>

<Ontology about="">
  <versionInfo>document-ont, v.1.0</versionInfo>
  <comment>An ontology that models documents, particularly
publications</comment>
  <imports
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/base1.0.daml
" />
  <imports
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.d
aml" />
</Ontology>

<daml:Class  ID="Communication">
  <subClassOf
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.d
aml#Event" />
</daml:Class >

<daml:Class  ID="Document">
  <subClassOf resource="#Communication" />
</daml:Class >

<daml:Class  ID="Abstract">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Comment">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Correspondence">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Discussion">
  <subClassOf resource="#Correspondence" />
</daml:Class >

<daml:Class  ID="Email">
  <subClassOf resource="#Correspondence" />
</daml:Class >
```

```
<daml:Class  ID="Letter">
  <subClassOf resource="#Correspondence" />
</daml:Class >

<daml:Class  ID="Postcard">
  <subClassOf resource="#Correspondence" />
</daml:Class >

<daml:Class  ID="Form">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Guideline">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Homepage">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="OrganizationHomepage">
  <subClassOf resource="#Homepage" />
</daml:Class >

<daml:Class  ID="PersonalHomepage">
  <subClassOf resource="#Homepage" />
</daml:Class >

<daml:Class  ID="Index">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Lecture">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Manuscript">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Minutes">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Preprint">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Promotion">
  <subClassOf resource="#Document" />
</daml:Class >
```

```
<daml:Class  ID="Publication">
  <subClassOf resource="#Document" />
</daml:Class >

<daml:Class  ID="Advertisement">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="Article">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="BookArticle">
  <subClassOf resource="#Article" />
</daml:Class >

<daml:Class  ID="ConferencePaper">
  <subClassOf resource="#Article" />
</daml:Class >

<daml:Class  ID="WorkshopPaper">
  <subClassOf resource="#Article" />
</daml:Class >

<daml:Class  ID="JournalArticle">
  <subClassOf resource="#Article" />
</daml:Class >

<daml:Class  ID="Book">
  <subClassOf resource="#Publication" />
</daml:Class >

<Property ID="author">
  <label>is written by</label>
  <domain resource="#Document" />
  <range
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.d
aml#Person" />
</Property>

<daml:Class  ID="Dictionary">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="Editorial">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="Manual">
  <subClassOf resource="#Publication" />
</daml:Class >
```

```
<daml:Class  ID="Periodical">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="Journal">
  <subClassOf resource="#Periodical" />
</daml:Class >

<daml:Class  ID="Magazine">
  <subClassOf resource="#Periodical" />
</daml:Class >

<daml:Class  ID="Newsletter">
  <subClassOf resource="#Periodical" />
</daml:Class >

<daml:Class  ID="Newspaper">
  <subClassOf resource="#Periodical" />
</daml:Class >

<daml:Class  ID="Proceedings">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="Regulation">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="Specification">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="TechnicalReport">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="Thesis">
  <subClassOf resource="#Publication" />
</daml:Class >

<daml:Class  ID="DoctoralThesis">
  <subClassOf resource="#Thesis" />
</daml:Class >

<daml:Class  ID="MastersThesis">
  <subClassOf resource="#Thesis" />
</daml:Class >

<daml:Class  ID="Review">
  <subClassOf resource="#Document" />
</daml:Class >
```

```
<daml:Class  ID="PhoneCall">
  <subClassOf resource="#Communication" />
</daml:Class >

<daml:Class  ID="Software">
  <subClassOf resource="#Communication" />
</daml:Class >

<daml:Class  ID="Speech">
  <subClassOf resource="#Communication" />
</daml:Class >

<daml:Class  ID="DocumentRepresentation">
  <subClassOf
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.d
aml#Artifact" />
</daml:Class >

<daml:Class  ID="PaperDocument">
  <subClassOf resource="#DocumentRepresentation" />
</daml:Class >

<daml:Class  ID="ElectronicDocument">
  <subClassOf resource="#DocumentRepresentation" />
</daml:Class >

<Property ID="softwareDocumentation">
  <label>is documented in</label>
  <domain resource="#Software" />
  <range resource="#Publication" />
</Property>

<Property ID="publisher">
  <label>is published by</label>
  <domain resource="#Document" />
  <range
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.d
aml#Organization" />
</Property>

<Property ID="subject">
  <label>has subject</label>
  <domain resource="#Document" />
  <range
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/base1.0.daml
#SHOEEntity" />
</Property>


<Property ID="publishDate">
  <label>was published on</label>
  <domain resource="#Document" />
</Property>
```

```
<Property ID="communicator">
  <label>is communicated by</label>
  <domain resource="#Communication" />
  <range
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.d
aml#Agent" />
</Property>

<Property ID="authorOrg">
  <label>is written by</label>
  <domain resource="#Document" />
  <range
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.d
aml#Organization" />
</Property>

<Property ID="containedIn">
  <label>is contained in</label>
  <domain resource="#Document" />
  <range resource="#Document" />
</Property>

<Property ID="recipient">
  <label>is received by</label>
  <domain resource="#Communication" />
  <range
resource="http://www.cs.umd.edu/projects/plus/DAML/onts/general1.0.d
aml#Agent" />
</Property>

<Property ID="title">
  <label>is titled</label>
  <domain resource="#Document" />
</Property>

<Property ID="volume">
  <label>is volume</label>
  <domain resource="#Periodical" />
</Property>

<Property ID="softwareVersion">
  <label>is version</label>
  <domain resource="#Software" />
</Property>

</rdf:RDF>
```

Como visto na seção 4.2 do presente capítulo, a interface é criada de acordo com a ontologia escolhida, representando tal ontologia em forma de árvore hierárquica. Portanto, suponha que o usuário navegue na ontologia escolhida e realize os seguintes passos ilustrados na figura 4.3:

1 - Acessa o elemento **Publication**;

2 - Escolhe o campo **author** e o preenche com "*J.K. Rowling"* para realizar sua pesquisa.

3 – Escolhe o campo **title** e o preenche com "*Harry Potter and the Goblet Fire"*;

4 – Submete a consulta.

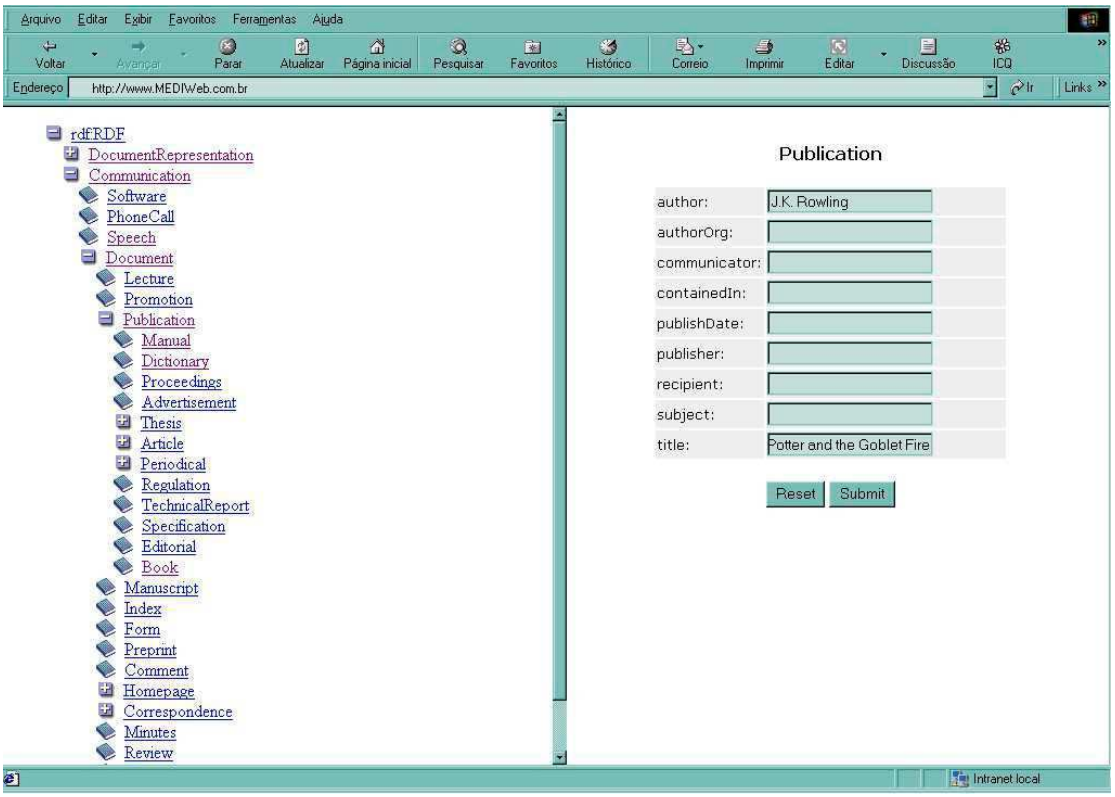

Figura 4.3: Exemplo de consulta no elemento *Publication.*

Como mencionado anteriormente, o carregador de ontologias junto com o *framework* de publicação faz um acoplamento da interface com as ontologias existentes no sistema. Deste modo, cada item da estrutura da árvore montada em HTML e os seus campos de entrada de dados correspondem respectivamente a uma **Classe** da ontologia DAML + OIL e suas **propriedades**. Portanto, neste primeiro exemplo, tem-se a **classe Publication** e suas propriedades escolhida para consulta, as **propriedades author e title**.

A partir das informações passadas pelo usuário através da interface, uma consulta genérica é automaticamente criada e a partir dela, são criadas consultas específicas para cada tipo de fonte de dados:

Consulta XMLQL:

```
function query() {
construct $a
where <Root.Publication><author>"J.K. Rowling"</><title>"Harry Potter and the Goblet Fire"</></>
 ELEMENT_AS $a IN{“http://www.dsc.ufpb.br/~ladjane/Publication.xml”}
}
```

Consulta SQL:

```
SELECT * from Publication WHERE Author = ‘J.K. Rowling’ AND title = ‘Harry Potter and the Goblet Fire’;
```

O componente de reescrita registra o elemento escolhido (*Publication)*, instancia a classe ***CarregadorOntologia,*** que por sua vez, instancia a classe ***DomTreeOntologia*** para realizar o *parsing*, caminhar na árvore XML gerada e resgatar “os filhos” do elemento escolhido. O número limite de filhos a ser resgatado deverá ser estipulado para que não haja sobrecarga de consultas e conseqüentemente a perda de performance no sistema. A configuração deste número é feita também pelo método ***reescreveConsulta()*** e pode variar para cada ontologia cadastrada no sistema.

Para simplificar o processo de enriquecimento semântico no nosso estudo de caso, consideramos apenas o elemento ***Book*** como filho do elemento ***Publication***. Na figura 4.4 circulamos os elementos filhos de *Publication***,** destacando o elemento *Book*. A partir do que foi considerado, a consulta original é reescrita, adicionando-se o elemento Book e então, são criadas novas consultas específicas:

Consulta XMLQL:

```
function query(){
construct $a
where <Root.Book><author>"J.K. Rowling"</><title>"Harry Potter and
the Goblet Fire"</></>
ELEMENT_AS $a IN{"http://www.dsc.ufpb.br/~ladjane/Book.xml”}
}
```

Consulta SQL:

```
SELECT * from Book WHERE Author = ‘J.K. Rowling’ AND title = ‘Harry Potter and the Goblet Fire’;
```

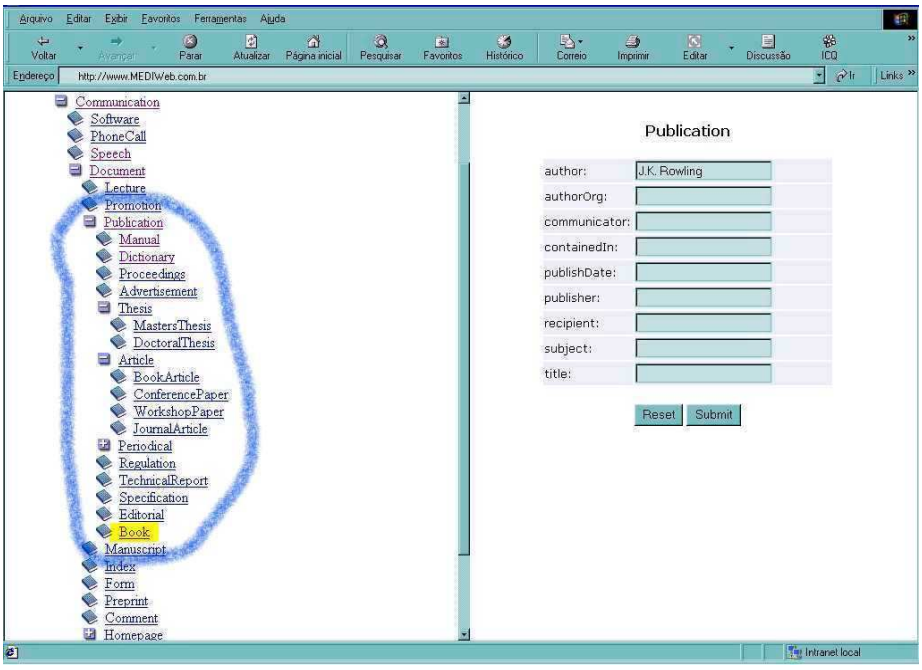

Figura 4.4: Destaque do elemento *Book*

A classe ***MaquinaBusca*** direciona as consultas XMLQL e SQL para a ***MaquinaXML*** e para os ***Wrappers***, respectivamente.

A ***MaquinaXML*** chama o processador XMLQL e este realiza a busca aos dados semi-estruturados localizados nas fontes descritas na consulta XMLQL. Ressalta-se que tais fontes são previamente cadastradas no sistema e são representadas pela classe ***FonteSemiEstruturada.*** Neste exemplo em particular, as fontes nas quais serão realizadas as buscas são:

- *http://www.dsc.ufpb.br/~ladjane/Publication.xml*.
- *http://www.dsc.ufpb.br/~ladjane/Book.xml*.

É importante ressaltar também que estas fontes XML estão descritas sintaticamente de acordo com o esquema da ontologia *Document*, garantindo a consistência semântica da consulta. Como resultado da consulta, os dados requeridos são formatados em XML e assim, são retornados para a ***MaquinaBusca***. A figura 4.5 mostra este resultado.

```
<Resultado>
<Book>
  <Title>Harry Potter and the Goblet Fire</Title>
  <Author>J.K. Rowling</Author>
  <Date>May, 2001</Date>
  <ISBN>85-325-1252-6</ISBN>
  <Publisher>Rocco</Publisher>
</Book>
</Resultado>
```

Figura 4.5: Resultado da busca aos dados semi-estruturados

Para a realização da consulta aos dados estruturados, a ***MaquinaBusca*** envia a consulta SQL para cada ***wrapper*** das fontes que pertencem à ontologia *Document.* A fim de deixar mais claro nosso exemplo, suponha que existe apenas uma fonte pertencente à citada ontologia. Esta fonte será denominada de ***Library*** e seu respectivo *wrapper* denominado de ***WrapperLibrary,*** que é por herança um ***WrapperRelacional,*** pois a fonte Library é um banco de dados do modelo relacional, contendo informações de um sistema de biblioteca.

O ***WrapperLibrary*** é responsável pelo processo de mapeamento dos esquemas, realizando por exemplo, tarefas como:

- Converter a consulta SQL recebida pela máquina de busca, que está sintaticamente de acordo com o esquema da ontologia, em uma consulta válida para a fonte local;

- Realizar a consulta na sua fonte Library;
- Retornar os resultados da consulta em formato XML para a máquina de busca.

Levando em consideração as regras para o processo de correspondência de esquemas vistas na seção anterior, mostramos como cada regra é utilizada para o preenchimento da tabela de mapeamento, que é utilizada pelo *WrapperLibrary***.** Tal preenchimento é feito através da análise dos elementos envolvidos nas consultas do nosso estudo de caso.

Temos o seguinte cenário:

SQL Query: SELECT * from Book WHERE Author = 'J.K. Rowling' AND title = 'Harry Potter and the Goblet Fire';

Pelo esquema da ontologia:

- **Book e Publication são classes da ontologia Document; (figura.. 4.6);**
- **author e title** são as propriedades de ambas classes. (figura. 4.6);

No esquema da fonte *Library*:

- **Publication** corresponde à tabela *Book, Thesis, Proceedings, Journal, Article e Magazine*. Porém, para facilitar o entendimento dos processos em questão, consideramos que Publication corresponde à tabela Book (figura. 4.7). Além do mais, a lógica da aplicação das regras para os demais elementos é a mesma.
- **Book** corresponde à tabela Book (figura. 4.7);
- A propriedade **author** corresponde à tabela Author (figura. 4.7);
- **title** corresponde a um atributo de book, o atributo title (figura. 4.7);
- A relação (book, author) tem cardinalidade (*,*);
- Há uma tabela de junção, chamada **Author_Book,** das tabelas book e author.

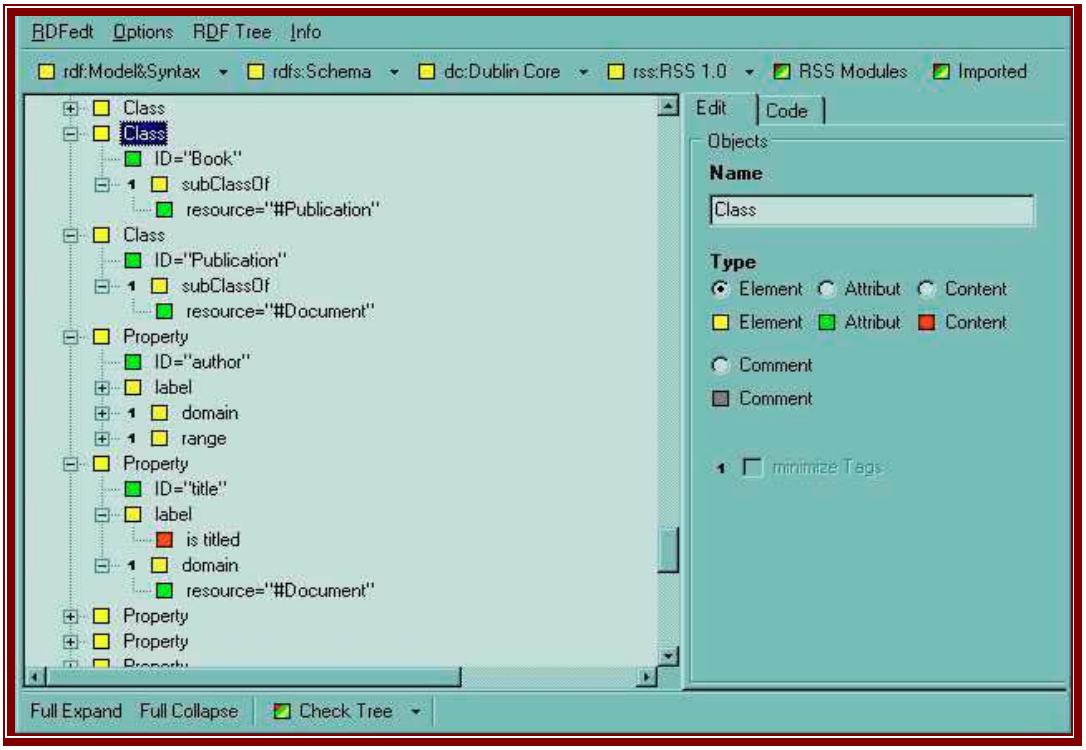

Figura 4.6: A ontologia *Document*

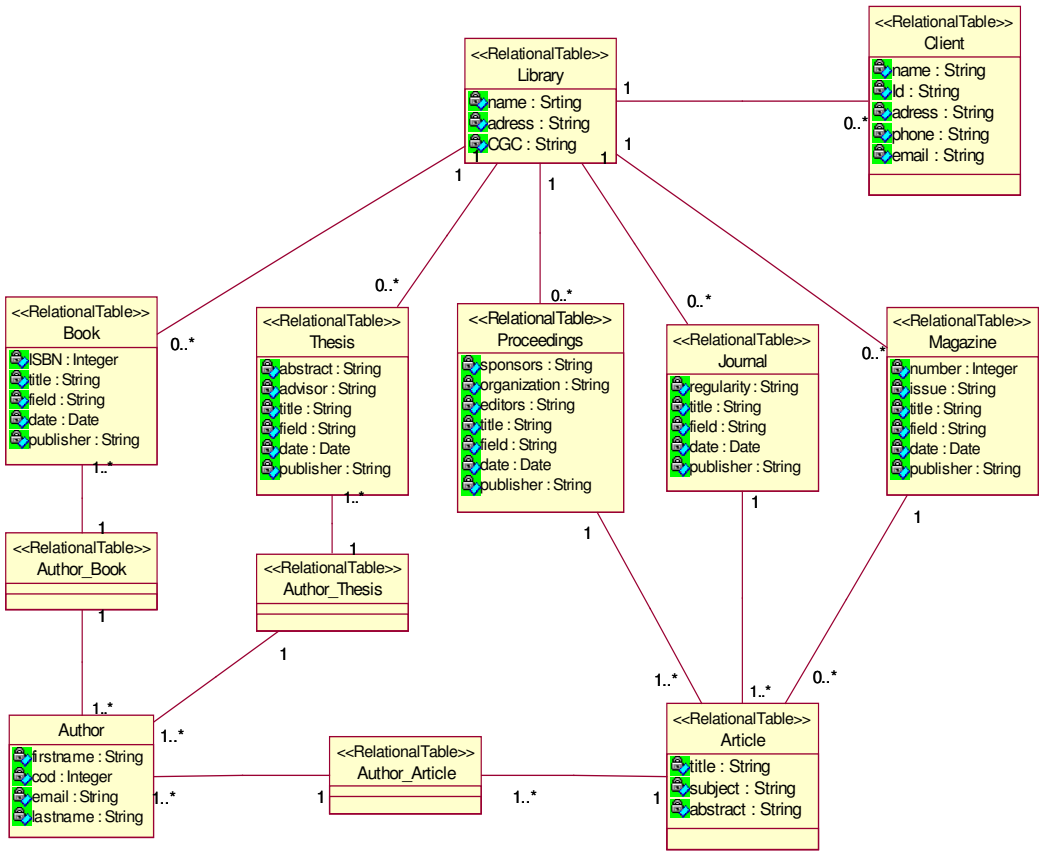

Figura 4.7: Esquema da fonte *Library*

Com estas informações sobre os dois esquemas, podemos identificar quais regras foram usadas para preencher a tabela de mapeamento da fonte *Library.* Analisemos primeiramente o relacionamento das tabelas Book e Author na figura 4.7. Percebe-se que sua cardinalidade é n:n, existindo então, uma tabela de junção, denominada Author_Book. Portanto, a regra adequada para tal situação é a regra #4. A tabela 4.8 mostra o preenchimento da tabela de mapeamento seguindo esta regra.

Classe	Atributo	Tabela	Atributo	Junção
C1	p	Eq_C1 x, Eq_p y, tabjuncao ab	y.atributo	y.PK = ab.FK_p AND ab.FK_x = x.PK
Book	Author	Book x, Author a, Author_Book y	a.firstname	a.cod = y.author AND y.book = x.ISBN

Tabela 4.8: Preenchimento da tabela de mapeamento da regra #4

Analisemos agora a propriedade *title* no esquema da fonte Library. Percebe-se que *title* corresponde diretamente a um atributo da tabela Book Portanto, podemos aplicar a regra #1. A tabela 4.9 mostra o preenchimento da tabela de mapeamento para esta regra.

Classe	Propriedade	Tabela	Atributo	Junção
C1	P	Eq_C1	Eq_p	*null*
Book	Title	Book	title	*null*

Tabela 4.9: Preenchimento da tabela de mapeamento da regra #1

É através destas tabelas que a instância do ***WrapperLibrary*** faz o seu processo de mapeamento, criando conseqüentemente, uma consulta válida para a fonte *Library.*

Logo, a consulta SQL para a fonte *Library* é:

*Select * from Book x, Author a, Author_Book y WHERE a.firstname = 'J.K. Rowling' AND a.id = y.id_author AND y.isbn_book = x.ISBN AND x.title =* "Harry Potter and the Goblet Fire"*;*

Esta consulta é portanto processada na fonte *Library* e um conjunto de dados que a satisfaz é retornado para o *WrapperLibrary*. O passo seguinte do *WrapperLibrary* é transformar o conjunto de resultados recebido em XML e retorná-lo para a máquina de busca. A figura 4.8 mostra o resultado da consulta da fonte *Library*, em XML.

```
<Library>
<Book>
  <Title> Harry Potter and the Goblet Fire</Title>
  <Date> May, 2001</Date>
  <ISBN> 85-325-1252-6</ISBN>
  <Field> infantile </Field>
  <Publisher> Rocco</Publisher>
  <Firstname> J.K </Firstname>
  <Lastname> Rowling </Lastname>
  <Email> Rowling@rocco.com </Email>
</Book>
</Library>
```

Figura 4.8: Resultado da fonte *Library*

Com os resultados de ambos tipos de fonte (Fig. 4.5 e Fig. 4.8), o **Mediador** faz a integração destes em um único resultado. A figura 4.9 mostra o resultado final, em XML, da consulta feita pelo usuário.

```
<Resultado>
<Book>
  <Title>Harry Potter and the Goblet Fire</Title>
  <Author>J.K. Rowling</Author>
  <Date> May, 2001</Date>
  <ISBN>85-325-1252-6</ISBN>
  <Publisher> Rocco</Publisher>
</Book>
<Book>
  <Title>Harry Potter and the Goblet Fire</Title>
  <Date> May, 2001</Date>
  <ISBN>85-325-1252-6</ISBN>
  <Field> infantile </Field>
  <Publisher> Rocco</Publisher>
  <Firstname> J.K </Firstname>
  <Lastname> Rowling </Lastname>
  <Email> Rowling@rocco.com </Email>
</Book>
</Resultado>
```

Figura 4.9: Resultado Final

Depois da integração dos resultados, o mediador chama os serviços da interface e os envia o resultado final. O *framework* de publicação usa o processador da linguagem *XSL stylesheet* para transformar XML em HTML e logo após, publica o resultado da consulta no dispositivo do usuário – o *web browser*.

Capítulo 5

Conclusões

A Web Semântica representa atualmente um dos grandes desafios para a comunidade científica, pois consiste em uma integração de dados que pode ser utilizada por máquinas e humanos. Além disto, deve permitir automação e a reutilização de dados por várias aplicações.

Neste trabalho, apresentamos uma proposta baseada em mediadores para a integração semântica de dados na *web.* A nossa proposta procurou ser flexível o bastante para atender aos novos requisitos provenientes da visão da Web Semântica. Para isto, utilizamos a abordagem de ontologias com o intuito de prover principalmente o suporte à interoperabilidade e ao consenso semântico na integração dos dados. Também adotamos a linguagem XML para a representação dos dados semi-estruturados e estruturados no processo de integração, adquirindo vantagens do uso deste padrão, como a interoperabilidade sintática e um alto ganho de flexibilidade.

Na camada de interface da arquitetura proposta, a representação da informação integrada em XML, aliada ao uso do *framework* de publicação, proporciona o gerenciamento, transformação e apresentação dos dados de forma automatizada em diferentes dispositivos *web.* Propomos também, uma interface acoplada com as ontologias cadastradas no sistema. Desta forma, criou-se um "espelho" das ontologias, possibilitando uma navegação sempre coerente com as eventuais atualizações das mesmas.

Para representar as ontologias do sistema utilizamos a linguagem DAML + OIL, que se baseia em RDF e XML e supre limitações das mesmas, provendo infra-estrutura para o desenvolvimento de agentes inteligentes e outras aplicações. Desta forma, permitimos uma arquitetura extensível à implementação dos requisitos da Web Semântica.

No decorrer do trabalho, apresentamos também o projeto e implementação de um protótipo para validar a arquitetura proposta. Procuramos separar as consultas em dois tipos: consultas para fontes estruturadas e para fontes semi-estruturadas.

Para as fontes semi-estruturadas, projetamos o sistema para a utilização de processadores de quaisquer linguagens de consulta para XML. No entanto, utilizamos no nosso protótipo o processador da linguagem XML-QL. Esta ferramenta supriu nossas necessidades, principalmente por possibilitar a realização de uma consulta em diferentes URLs, por ter uma sintaxe relativamente fácil de usar (SQL-like) e por estruturar as respostas em documentos XML. Porém, identificamos através de alguns testes, que a performance da ferramenta é comprometida com um pequeno aumento no número das fontes a consultar. Isto se torna um fato grave face à necessidade de consultas a um grande número de fontes no ambiente *web.*

No caso das fontes estruturadas, consideramos apenas fontes do modelo relacional. Desenvolvemos *wrappers* para cada fonte do sistema com a responsabilidade de realizar o mapeamento das consultas. Adicionalmente, adotamos uma tabela padrão para armazenar os dados das correspondências entre os esquemas envolvidos, como também, criamos algumas regras de correspondências para nortear o especialista humano no preenchimento da tabela de mapeamento, a qual é utilizada pelos *wrappers.*

Estudos de caso foram desenvolvidos a fim de ajudar o leitor a entender melhor o modo como o sistema funciona. Ademais, os exemplos apresentados ajudaram a esclarecer alguns tópicos mais específicos, como todo o processo de mapeamento e a integração dos dados no mediador.

O componente de reescrita, responsável pelo enriquecimento semântico das consultas, foi desenvolvido considerando a relação hierárquica dos elementos da ontologia. Um número representando o limite de profundidade da árvore hierárquica foi pré-estabelecido para que não ocasionasse alta carga na consulta, o que geraria conseqüentemente a perda de performance.

Levantamos alguns requisitos na fase de projeto, com base neles, concluímos que os objetivos foram alcançados. Enfim, quando propomos uma arquitetura e implementação de um integrador semântico de dados na *web*, primamos por flexibilidade para que a visão da Web Semântica fosse atendida.

Trabalhos Futuros

A partir das conclusões expostas, pode-se identificar algumas questões importantes para o aperfeiçoamento e extensão futura deste trabalho. Vale salientar que abordamos alguns tópicos de diferentes áreas, o que dá margem ao desenvolvimento de diversas outras pesquisas e extensões mais específicas. No entanto, consideramos aqui alguns tópicos mais relevantes no âmbito do nosso projeto de integração de dados na *web*.

Com relação à interface, é preciso realizar a implementação, seguida obviamente de testes, para outros dispositivos *web* como PDAs e telefones celulares. Com o auxílio do framework de publicação, esta tarefa torna-se mais simples devido ao conjunto de serviços apropriados para a automação deste processo.

O projeto desenvolvido foi genérico suficiente para dar suporte ao uso de processadores para linguagens de consulta aos documentos XML. Por conseguinte, vislumbra-se a necessidade de realizar extensão para outros tipos de linguagens, tais como XQuery, a qual foi adotada como padrão.

Na integração dos dados estruturados, levantamos algumas regras básicas para ajudar no processo de correspondência entre o esquema ontológico do sistema e o esquema local da fonte de dados. No entanto, estas regras necessitam de um estudo minucioso, tangendo a problemática de mapeamento entre esquemas. Identificamos também, a necessidade de estender o nosso protótipo para acessar fontes dos modelos objeto-relacional e de orientação a objetos, pois em sua implementação consideramos apenas fontes do modelo relacional.

Por fim, o componente de reescrita, responsável pelo processo de enriquecimento semântico, deverá incorporar mais artifícios para enriquecer as consultas. Como este processo está correlacionado com as ontologias em DAML + OIL, deve-se principalmente investir na utilização e/ou na construção de regras de inferência, na construção de agentes inteligentes, na integração e na interoperabilidade entre as ontologias, pois assim, poderemos prover mais serviços para a Web Semântica.

Referências Bibliográficas

[Abiteboul, 1997] Abiteboul, S. *Querying Semi-Structured Data*. In Proceedings of the International Conference on Database Theory, pp. 1-18. Jan .1997.

[Abiteboul *et al*, 1997] Abiteboul, S. et al. *The Lorel query language for semistructured data.* International Journal on Digital Libraries, 1(1), pp. 68-88, 1997.

[Abiteboul *et al*, 2000] Abiteboul, S.; Buneman, P.; Suciu, D. *Data on the Web: from Relations to Semistructured Data and XML*. San Francisco: Morgan Kaufmann, 2000.

[Andersen, 2001] Andersen, B. *What is an ontology?.* Ontology Works, Inc. Janeiro, 2001.

[Arruda *et al*, 2002] Arruda, L. S.; Baptista, C. S.; Lima, C. A. A. *MEDIWeb: a Mediator-based Environment for Data Integration on the Web.* 4th International Conference on Enterprise Information Systems, Ciudad Real – Spain, Abril de 2002.

[Baru *et al,* 1998] Baru, C. et al. *Features and Requirements for an XML View Definition Language: Lessons from XML Information Mediation.* In: W3C WORKSHOP ON QUERY LANGUAGE (QL´98), 1998, Boston. Disponível em http://www.w3.org/Tands/QL/QL98/pp/xmas.html. Acesso em 14 de dezembro de 2001.

[Bergamaschi *et al*, 1999] Bergamaschi, S.; Caetano, S.; Vincini, M. *Semantic Integration of Semistructured and Structured Data Sources*. SIGMOD Record, v.28, n.1, p.54-59, março de 1999.

[Berners-Lee et al, 2001] Berners-Lee, T; Hendler, J.; Lassila, O. *The Semantic Web.* Scientific American, vol. 284, nº. 5, maio 2001, p. 34-43.

[Bonifati, 2000] Bonifati, A. *Comparative Analysis of the Most Representative XML Query languages.* Politecnico di Milano. Disponível em: http://www.dbai.tuwien.ac.at/education/agenten/xml/xmlquery.pdf. Acessado em 13 de junho de 2001.

[Bonifati, 2000a] Bonifati, A., Ceri, S. *Comparative analysis of five XML query languages*. SIG-MOD Record, 29(1):68-79, maio de 2000.

[Booch *et al*., 1999] Booch, G. *et al*. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[Bosworth, 1999] Bosworth, A.; Brown Jr, A: *Microsoft's vision for XML***.** Disponível em: http://msdn.microsoft.com/xml/. Acessado em 23 de novembro de 2001.

[Bourret, 2000] Bourret, R. *XML and Databases*. Disponível em http://www.rpbourret.com/xml/XMLAndDatabases.htm. Acessado em novembro de 2000.

[Bradley, 1998] Bradley, N. *The XML Companion***.** Addison-Wesley, Reading (Mass) 1998.

[Brickley, 1999] Brickley, D., Guha, R.V. *Resource Description Framework (RDF) Schema Specification.* W3C Proposed Recommendation. March, 1999. http://www.w3.org/TR/PR-rdf-schema. Acessado em julho de 2001.

[Brownell, 2002] Brownell, D. *SAX2*. O´Reilly. Janeiro de 2002. ISBN:0-596-00237-8.

[Buneman, 1997] Buneman, P. Tutorial**:** *Semistructured data.* In Proceedings of ACM Symposium on Principles of Database Systems, 1997.

[Cattell *et al*, 1996] Cattell, R. et al. *The Object Database Standard: ODMG-93,* Release 1.2. Morgan Kaufmann Publishers, San Francisco, 1996.

[Chamberlin et al, 2000] Chamberlin, D., Robie, J. and Florescu, D. Quilt: An XML Query Language for Heterogeneous Data Sources. In. Proceedings of WebDB 2000 Conference, in Lecture Notes in Computer Science, Springer-Verlag, 2000 pp. 53--62, Dallas, US., Maio de 2000.

[Chaudhri *et al*, 1997] Chaudhri, V. *et al*. *The Generic Frame Protocol 2.0*. July, 1997.

[Chawathe, 1994] Chawathe, S.; Garcia-Molina, H.; Hammer, J. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In Proc. of 10th Meeting of the Information Processing Society of Japan (IPSJ). 1994.

[Cluet *et al*, 1999] Cluet, S., Jacqmin, S.; Simeon, J. *The New YATL: Design and Specifications.* Technical Report, INRIA, 1999.

[Cocoon, 2000] *Cocoon: A publishing framework.* Disponível em: http://xml.apache.org/cocoon/index.html. Acessado em março de 2001.

[Corcho, 2000] Corcho, O.; Gómez-Pérez, A. *A Roadmap to Ontology Specification Languages.* EKAW 2000: Juan-les-Pins, France.

[Cover, 2002a] The XML Cover Pages. *XML and Query Languages*. Disponível em: http://xml.coverpages.org/xmlQuery.html. Acessado em Março de 2002.

[Cover, 2002b] The XML Cover Pages. *Extensible Stylesheet Language (XSL).* Disponível em: http://xml.coverpages.org/xsl.html. Acessado em abril de 2002.

[Cui et al, 2001] Cui, Z., Jones, D., O´Brien, P. *Issues in Ontology-based Information Integration*. Workshop on Ontologies & Information Sharing. Seattle, August 5 2001.

[Deutsch et al, 1999a] Deutsch, A. *et al*. *A Query Language for XML*. In: World Wide Web Conference (WWW8), 1999, Toronto, Canada. Disponível em: http://www.research.att.com/~mff/xmlql/doc/files/final.html. Acessado em março de 2001.

[Deutsch *et al*, 1999b] Deutsch, A. *et al*. *Querying XML Data***.** 1999. IEEE Data Engineering Bulletin, Vol 22 No. 3.

[Dodge, 2000] Dodge, G.; Gorman, T. *Essential Oracle8i Data Warehousing.* John Wiley & Sons, 2000.

[Elmasri, 2000] Elmasri, E.; Navathe, S. B. *Fundamentals of Database Systems***.** Third Edition, Addison-Wesley, 2000.

[Erdmann, 1999] Erdmann, M.; Studer, R.: *Ontologies as Conceptual Models for XML Documents.* In: Proceedings of the 12th Workshop on KnowledgeAcquisition, Modelling and Management (KAW'99), Banff, Canada, October 1999.

[Erdmann, 2000] Erdmann, M.; Decker, S. *Ontology-aware XML-Queries.* Submission for WebDB 2000.

[Farquhar *et al*, 1996] Farquhar, A., Fikes, R., Rice, J. *The Ontolingua Server: A Tool for Collaborative Ontology Construction.* Proceedings of KAW96. Banff, Canada, 1996.

[Fernandez et al, 1999] Fernandez M. *et al*. *XML Query Languages*: *Experiences and Exemplars*. Disponível em: http://www.w3.org/1999/ql/docs/xquery.html. Acessado em novembro de 2001.

[Flanagan, 1997] Flanagan, D. *Java in a Nutshell*. 2nd edition. CA: O'Reilly, 1997.

[Florescu et al, 1998] Florescu, D.; Levy A. Y; Mendelzon. *Database Techniques for the World Wide Web: A Survey*. SIGMOD Record, v.27, n.3, p.59-74, Mar. 1997.

[Fowler, 1999] Fowler, Martin. *Refactoring: improving the design of existing code*. ISBN: 0-201-48567-2, Addison-Wesley, 1999.

[Garcia-Molina *et al* , 2000] Garcia-Molina, H.; Ullman, J. D.; Widom, J. *Database System Implementation.* Prentice Hall; ISBN: 0130402648; Maio, 1999.

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

[Gómez-Pérez et al, 2000] Gómez-Pérez, A.; Fernández-López, M.; Corcho, O. *Methodologies, Tools and Languages. Where is the Meeting Point?* Artificial Intelligence Laboratory Technical University of Madrid (UPM) – Spain. Disponível em: http://www.schin.ncl.ac.uk/protege2001/presentations/GOMEZ-~1.PDF. Acessado em março de 2002.

[Gómez-Pérez, 2002] Gómez-Pérez, A.; Corcho, O. *Ontology Languages for the Semantic Web.* IEEE Intelligent Systems, January/February, vol. 17, nº. 1.

[Gruber, 1993] Gruber, T. *What is an ontology?*, 1993. Disponível em http://www-ksl.stanford.edu/kst/what-is-an-ontology.html. Acessado em: abril de 2001.

[Guarino, 1996] Guarino, N. *Understanding, Building, And Using Ontologies.* Disponível em: http://ksi.cpsc.ucalgary.ca/KAW/KAW96/guarino/guarino.html. Acessado em abril de 2002.

[Guarino, 1998] Guarino, N. *Formal Ontology and Information Systems*. In: (Ed.) *Formal Ontology in Information Systems*. pp. 3-15, IOS Press, Amsterdam, Netherlands.

[Hall, 2000] Hall R. *Exchanging Data via XML: How Oracle Portal-to-Go Delivers the Internet to Mobile Devices.* Oracle Corporation. February 2000. Disponível em : http://otn.oracle.com/tech/xml/info/htdocs/portaltogo/ptghome.htm. Acessado em

[Hammer *et al*, 1998] Hammer, J. *et al*. *Template-based Wrappers in the Tsimmis System.* In proc. of ACM SIGMOD Conf. On Management of Data, Tucson, Arizona.1998.

[Hendler, 2001] Hendler, J. *Agents and the Semantic Web*. IEEE Intelligent Systems, vol. 16, no. 2, Mar./Apr. 2001, pp. 30--37.

[Horrocks *et al*, 2000] Horrocks, I. et al. *OIL in a Nutshell*, Proc. ECAI '00 Workshop on Application of Ontologies and PSMs, Berlin, Germany, 2000, pp. 4.1–4.12.

[Horrocks, 2001] Horrocks, I; Harmelen, F. V. *Reference Description of the DAML+OIL Ontology Markup Language.* Draft report, 2001. Disponível em: www.daml.org/2000/12/reference.html. Acessado em Junho de 2001.

[Hwang, 2000] Hwang, C. *Incompletely and Imprecisely Speaking: Using Dynamic Ontologies for Representing and Retrieval Information.* Technical Report of Microelectronics and Computer Technology Corp, MCC, 2000.

[Ives, 2000] Ives, Z. G.; Lu, Y. *XML Query Languages in Practice: An Evaluation.* Web Age Information Management 2000, Shanghai, China.

[Jones *et al*, 1998] Jones, D. M.; Bench-Capon T. J. M; Visser, P. R. S. *Methodologies for Ontology Development.* Proceedings IT&KNOWS Conference of the 15th IFIP. World Computer Congress, Budapest, Chapman-Hall, 1998.

[Karp *et al*,1999] Karp, R., Chaudhri, V., Thomere, J. *XOL: An XML-Based Ontology Exchange Language* (version 0.4). Aug., 1999. Disponível em: www.ai.sri.com/~pkarp/xol. Acessado outubro de 2001.

[Kent, 1998] Kent, R. *Conceptual Knowledge Markup Language (version 0.2).* 1998. Disponível em: www.ontologos.org/CKML/CKML%200.2.html. Acessado em maio 2001.

[Kifer *et al*, 1995] Kifer, M.; Lausen, G.; Wu, J. *Logical Foundations of Object-Oriented and Frame-Based Languages*. Journal of the ACM. 1995.

[Kimball, 1996] Kimball, R. *The Data Warehouse Toolkit.* John Wiley & Sons, 1996.

[Kimball et al, 1998] Kimball, R. et al. *The Data Warehouse Lifecycle Toolkit*. John Wiley & Sons, 1998.

[Larman, 1998] Larman, C. *Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design*. ISBN: 0-13-748880-7, Prentice-Hall, 1998.

[Lassila, 1999] Lassila, O.; Swick, R. *Resource Description Framework (RDF) Model and Syntax Specification.* W3C Recommendation. January, 99. Disponível em: http://www.w3.org/TR/PR-rdf-syntax.. Acessado em maio de 2001.

[Levy, 1999] Levy, A. Y. *Combining artificial intelligence and databases for data integration.* In: M. Wooldridge, M.M. Veloso (Eds.), Artificial Intelligence Today: Recent Trends and Developments, Lecture Notes in Comput. Sci., Vol. 1600, Springer, Berlin, 1999, pp. 249—268.

[Lee, 2000] Lee, D.; Chu, W. W. *Comparative Analysis of Six Schema Languages*. University of California, Los Angeles 2000.

[Lahiri et al, 1999] Lahiri, T, Abiteboul, S and Widom, J. *Ozone: Integrating Structured and Semistructured Data*. Disponível em: http://wwwdb. stanford.edu/pub/papers/ozone.ps

[Luke, 2000] Luke S.; Heflin J. *SHOE 1.01. Proposed Specification. SHOE Project. February, 2000.* Disponível em: http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm. Acessado em novembro de 2001.

[MacGregor, 1991] Macgregor, R. *Inside the LOOM classifier*. SIGART Bulletin. #2(3):70-76. Junho, 1991.

[McLaughlihn, 2000] Mclaughlihn, B. *Java and XML.* O`reilly, June 2000.

[Maier, 1998] Maier, D. *Database desiderata for an XML query language.* In: W3C Workshop On Query Languages (QL'98), 1998, Boston. Disponível em: http://www.w3.org/TandS/QL/QL98/pp/maier.html. Acessado em junho de 201.

[Marchiori, 1999] Marchiori, M. *Query Languages*. In: The Eighth International World Wide Web Conference (WWW8), 1999, Toronto. Disponível em: http://www.w3.org/Talks/1999/0512-QL-WWW8/

[Mello *et al*, 2000] Mello, R. et al. Tutorial: *Dados Semi-Estruturados.* No SBBD 2000, João Pessoa, Paraíba, Brasil.

[Mena *et al*, 2000] Mena, E. *et al*. *OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies*. Distributed and Parallel Databases, 2000.

[Momis, 2002] *The MOMIS Project*. DataBase Group. Facoltà di Ingegneria - sede di Modena. Università di Modena e Reggio Emilia. Disponível em: http://sparc20.ing.unimo.it/Momis/. Acessado em novembro de 2001.

[Motta, 1999] Motta, E. *Reusable Components for Knowledge Modelling.* IOS Press. Amsterdam. 1999.

[Noy, 2001] Noy, N. F.; Mcguinness, D. L. *Ontology Development 101: A Guide to Creating Your First Ontology.* Stanford University, Stanford, CA. Disponível em: http://www.ksl.stanford.edu/dlm/papers/ontology-tutorial-noy-mcguinness.htm. Acessado em agosto de 2001.

[O´Neil, 2000] O´ NEIL, P.; O´NEIL, E. *Database: Principles, Programming, and Performance,* Morgan Kaufmann, 2000.

[Pulvermacher et al, 2000] Pulvermacher, M. K., Brandsma, D. L. Wilson, J. R. *A Space Surveillance Ontology Captured in an XML Schema.* Center for Air Force C2 Systems. Bedford, Massachusetts, October 2000.

[Robie, 1998] Robie, J.; Lapp, J.; Schach, D. *XML Query language (XQL).* Proposta para o XSL Working Group. Disponível em: http://www.w3.org/Style/XSL/Group/1998/09/XQL-proposal.html

[Rosenthal, 1999] Rosenthal A.; Seligman L.; Costello R. *XML, Databases, and Interoperability.*

[Rumbaugh *et al*, 1999a] Rumbaugh, J.; Booch, G.; Jacobson, I. *The Unified Modeling Language Reference Manual.* ISBN: 0-201-30998-X, Addison-Wesley, 1999.

[Salgado, 2001] Salgado, A. C.; Lóscio, B. F. *Integração de Dados na Web.* Anais da XX Jornada de Atualização em Informática, p. 131-173. 2001.

[SAX, 2002] *The official website for SAX.* Disponível em: http://www.saxproject.org/. Acessado em janeiro de 2002.

[SemanticWeb, 2002] *The Semantic Web Community Portal.* Última atualização em 04 de junho de 2002. Disponível em: http://www.semanticweb.org/. Acessado em julho de 2002

[Schach *et al*, 1998] Schach, D.; Lapp; J.; Robie, J. *Querying and Transforming XML.* In Proc. of the Query Languages workshop, Cambridge, Mass., Dec.1998. Disponível em: http://www.w3.org/TandS/QL/QL98/pp/query-transform.html. Acessado em maio de 2001.

[Subrahmanian *et al*, 1995] Subrahmanian, V. S. *et al*. *HERMES: a Heterogeneous reasoning and mediator system.* Technical Report, University of Maryland. 1995.

[Suciu, 1998] Suciu D. *Semistructured Data and XML*. In Proceedings of International Conference on Foundations of Data Organization, Kobe, Japan, November 1998. Disponível em: http://citeseer.nj.nec.com/suciu98semistructured.html. Acessado em abril de 2001.

[Suciu, 1998a] Suciu D. *An Overview of Semistructured Data***.** SIGACT News, vol. 29, no. 4, p. 28-38. Dezembro, 1998.

[Uschold, 1993] Uschold, M.; Gruninger M. *Ontologies: Principles, Methods and Applications.* AIAI-TR-191. To Appear in Knowledge Engineering Review. Volume 11, number 2, June 1996.

[Ullman, 1997] Ullman J.; Widon, J. *A First Course in Database Systems.* Prentice Hall, New Jersey, 1997.

[W3C, 1999] *W3C Namespaces in XML*. Disponível em: http://www.w3.org/TR/1999/REC-xml-names-19990114. Acessado em agosto de 2001.

[W3C, 2000a] *W3C Working Draft – XML Query Requirements*. Disponível em: http://www.w3.org/TR/2000/WD-xmlquery-req-20000815. Acessado em agosto de 2001.

[W3C, 2000b] *W3C XML Schema Part 0: Primer.* Disponível em: http://www.w3.org/TR/2000/WD-xmlschema-0-20000407. Acessado em setembro de 2001.

[W3C, 2000c] *W3C Extensible Markup Language (XML) Activity***.** Disponível em http://www.w3.org/XML. Acessado em agosto de 2001.

[W3C, 2001a] *W3C XML Schema Part 2: Datatypes.* Disponível em : http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/. Acessado em setembro de 2001

[W3C, 2001b] *Extensible Stylesheet Language (XSL).* Version 1.0, Outubro 2001. Disponível em: http://www.w3.org/TR/xsl/. Acessado em dezembro de 2001

[W3C, 2001c] *XQuery 1.0: An XML Query Language*. W3C Working Draft 20 December 2001. Disponível em: http://www.w3.org/TR/xquery. Acessado em dezembro de 2001

[W3C, 2001d] *XML Path Language (XPath) 2.0*. W3C Working Draft 20 December 2001. Disponível em: http://www.w3.org/TR/xpath20/. Acessado em janeiro de 2002

[W3C, 2002a] *XQuery 1.0 Formal Semantics*. W3C Working Draft 26 March 2002. Disponível em: http://www.w3.org/TR/query-semantics/. Acessado em abril de 2002.

[W3C, 2002b] *Document Object Model (DOM) Level 3 Core Specification.* W3C Working Draft 09 April 2002. Disponível em: http://www.w3.org/TR/DOM-Level-3-Core/. Acessado em abril de 2002.

[W3C, 2002c] *Namespaces in XML 1.1.* W3C Working Draft 03 April 2002. Disponível em: http://www.w3.org/TR/xml-names11/. Acessado em abril de 2002.

[Wait, 1999] Wait, B. *Using XML in Oracle Database Applications: Part 3: Customizing Data Presentation.* Oracle Corporation. November 1999. Disponível em http://technet.oracle.com/tech/xml/info/htdocs/otnwp/xml_custom_presentation.htm Acessado em abril de 2001.

[Walsh, 1999] Walsh, N. *Schemas for XML***.** Disponível em: http://www.xml.com/xml/pub/1999/07/schemas/index.html. Acessado em março de 2001

[Wiederhold, 1994] Wiederhold, G. *Interoperation, Medition, and Ontologies*. In the Proceedings International Symposium on Fifth Generation Computer Systems (FGCS94), Workshop on Heteogeneous Cooperative Knowledge-Bases, Vol.W3, pages 33-48, ICOT, Tokyo, Japan, Dec. 1994.

[Wells, 1999] WELLS, D. *Wrappers*. Disponível em: http://www.objs.com/survey/wrap.htm. Acessado em abril de 2001.

[Zachary, 2000] Zachary G. Ives, Y. L: *XML Query Languages in Practice: An Evaluation.* Web-Age Information Management 2000. Disponível em: http://www.informatik.uni-trier.de/~ley/db/conf/waim/waim2000.html#IvesL00. p. 29-40. Acessado em agosto de 2001.