

# Dados Semi-Estruturados

Ronaldo dos Santos Mello - ronaldo@inf.ufrgs.br<sup>1</sup>; ronaldo@inf.ufsc.br<sup>2</sup>

Carina Friedrich Dorneles - dorneles@inf.ufrgs.br<sup>1</sup>

Adrovane Kade - adrovane@upf.tche.br<sup>1,3</sup>

Vanessa de Paula Braganholo - vanessa@inf.ufrgs.br<sup>1</sup>

Carlos Alberto Heuser - heuser@inf.ufrgs.br<sup>1</sup>

<sup>1</sup> Instituto de Informática (II)  
Universidade Federal do Rio  
Grande do Sul (UFRGS)

<sup>2</sup> Departamento de Informática e de  
Estatística (INE)  
Centro Tecnológico (CTC)  
Universidade Federal de Santa  
Catarina (UFSC)

<sup>3</sup> Centro de Processamento de  
Dados (CPD)  
Universidade de Passo Fundo  
(UPF)

## Abstract

When semistructured data are considered, different representations of a same information may co-exist. This feature makes difficult the prescription of a database schema. Data available in the *Web* are an example, varying from texts in natural language to well-formatted records. This tutorial presents the state-of-art of research on semistructured data, covering the following topics: data modelling, query languages, data extraction and relationship with ontologies.

## 1 Introdução

### 1.1 Motivação

Dados mantidos em Sistemas Gerenciadores de Bancos de Dados (SGBDs) apresentam uma estrutura de representação ou esquema previamente definido. O acesso e a manipulação destas esquemas são tarefas específicas do SGBD. Usuários ou aplicações realizam operações sobre estes dados com base neste esquema.

Contudo, nota-se atualmente que boa parte dos dados disponíveis para acesso eletrônico não estão mantidos em BDs. Alguns exemplos são diretórios de arquivos de documentos (atas de reuniões, processos, etc) de uma organização ou informações acessíveis através da *World Wide Web* (WWW) (dados *Web*). A justificativa para tal fato decorre da própria natureza destes dados. Dados *Web*, por exemplo, apresentam uma organização bastante heterogênea, que pode variar de um texto sem nenhuma formatação até um conjunto de registros bem formatados [Abi97a, Nes97]. Além disso, o volume destes dados pode ser grande e com muitos relacionamentos. Considerando dados referentes a um *curriculum vitae*, por exemplo, pode-se ter um pequeno texto informal descrevendo dados pessoais e experiência profissional ou, no extremo oposto, um documento organizado em seções e subseções, com referências (*links*) para dados das empresas e instituições onde a pessoa trabalhou.

A alta heterogeneidade desses dados torna complexa as atividades de pesquisa de dados, uma vez que não existe uma esquema uniforme a partir do qual uma consulta possa ser formulada. Consultas são, em geral, realizadas através de navegação exaustiva ou busca por palavras-chaves (*full-text search*) [Ash97, Atz97b, Cat98]. No primeiro caso, uma necessidade de informação do usuário pode ser inviável de ser encontrada, devido ao grande volume de dados

a ser consultado. No segundo caso, são utilizadas técnicas de recuperação de informação (RI) através de ferramentas como *search engines*, cujas desvantagens são o custo de manutenção dos índices para termos presentes em documentos e o volume da resposta fornecida, ficando a análise de relevância da informação basicamente por conta do usuário.

Dados que se enquadram nessas características são denominados **dados semi-estruturados**. O gerenciamento destes dados traz novos problemas à comunidade científica de BD, uma vez que a tecnologia de BDs convencionais não pode ser diretamente aplicada. A necessidade imediata de soluções para estes problemas é motivada principalmente pelo crescente uso da *Web* como veículo para publicação e intercâmbio de dados.

## 1.2 Dados Semi-Estruturados

Dados semi-estruturados apresentam uma **representação estrutural heterogênea**, não sendo nem completamente não-estruturados nem estritamente tipados. Dados *Web* se enquadram nessa definição: em alguns casos os dados possuem uma descrição uniforme (um catálogo de produtos), em outros, algum padrão estrutural pode ser identificado (um conjunto de documentos no formato de artigo), ou então, praticamente não existem informações descritivas associadas (um arquivo de imagem) [Abi97a]. Afirma-se também que dados semi-estruturados são dados nos quais o esquema de representação está presente (de forma explícita ou implícita) juntamente com o dado, ou seja, o mesmo é **auto-descritivo**. Isto significa que uma análise do dado deve ser feita para que a sua estrutura possa ser identificada e extraída [Bun97].

As características principais de dados semi-estruturados são [Abi97a, Flo98]:

- **Definição à posteriori**: esquemas para dados semi-estruturados são usualmente definidos após a existência dos dados, com base em uma investigação de suas estruturas particulares e da análise de similaridades e diferenças. Isto não significa que sempre existe um esquema associado a um dado semi-estruturado;
- **Estrutura irregular**: coleções extensas de dados semanticamente similares estão organizados de maneiras diferentes, podendo algumas ocorrências terem informações incompletas ou adicionais em relação a outras. Em suma, não existe um esquema padrão para esses dados. O exemplo do *curriculum vitae* se enquadra nesta característica;
- **Estrutura implícita**: muitas vezes existe uma estrutura básica para os dados, porém, essa estrutura está implícita na forma como os dados são apresentados. É necessário realizar uma computação para obter essa estrutura;
- **Estrutura parcial**: apenas parte dos dados disponíveis pode ter alguma estrutura, seja implícita ou explícita. Por exemplo, componentes de objetos que são arquivos *bitmaps* são não-estruturados. Já dados pessoais podem ter uma estrutura básica implícita ou explícita. Como consequência, um esquema para estes dados nem sempre é completo do ponto de vista semântico e nem sempre todas as informações esperadas estão presentes;
- **Estrutura extensa**: a ordem de magnitude de uma estrutura para estes dados é grande, uma vez que os mesmos são muito heterogêneos. Supondo diferentes formatos para um *curriculum vitae*, uma união de atributos significativos em cada formato pode produzir um esquema extenso;

- Estrutura evolucionária: a estrutura dos dados modifica-se tão freqüentemente quanto os seus valores. Dados *Web* apresentam este comportamento, uma vez que existe o interesse em manter dados sempre atualizados;
- Estrutura descritiva e não prescritiva: dada a natureza irregular e evolucionária dos dados semi-estruturados, as estruturas de representação implícitas ou explícitas normalmente se restringem a descrever o estado corrente de poucas ocorrências de dados similares. Desta forma, não é possível prescrever esquemas fechados e muitas restrições de integridade com relação à semântica dos atributos. Um sinônimo para estrutura descritiva é estrutura indicativa;
- Distinção entre estrutura e dados não é clara: como a estrutura está embutida na descrição dos dados, muitas vezes a distinção lógica entre estrutura e valor não é clara. Pode-se ter, por exemplo, um endereço representado como um valor atômico em uma ocorrência de dado (*string*) ou como um tipo definido pelo usuário (com atributos rua, número e complemento) em outra ocorrência. Esta característica torna mais complicado o projeto de um BD para tais dados.

As características de dados semi-estruturados diferem bastante das características de dados mantidos em BDs tradicionais, como BDs relacionais. A tabela 1.1 apresenta estas diferenças.

**Tabela 1.1** - Diferenças entre dados tradicionais e dados semi-estruturados

Dados tradicionais	Dados semi-estruturados
Esquema predefinido	Nem sempre há um esquema predefinido
Estrutura regular	Estrutura irregular
Estrutura independente dos dados	Estrutura embutida no dado
Estrutura reduzida	Estrutura extensa
Estrutura fracamente evolutiva	Estrutura fortemente evolutiva
Estrutura prescritiva	Estrutura descritiva
Distinção entre estrutura e dado é clara	Distinção entre estrutura e dado não é clara

BDs tradicionais apresentam um esquema predefinido e uma estrutura homogênea a nível de atributos e tipos. Em se tratando de dados semi-estruturados, cada ocorrência de dado pode ser heterogênea nos dois aspectos mencionados anteriormente. Dada essa heterogeneidade, em geral a estrutura de um dado semi-estruturado está presente na própria descrição do dado, necessitando ser identificada e extraída. Estas tarefas são complexas, uma vez que a distinção entre esquema e dados nem sempre é clara, se compararmos ocorrências de dados semanticamente iguais.

Como cada dado pode ter uma organização própria, uma estrutura de representação para um conjunto de dados tende a ser extensa, de forma a refletir as particularidades de cada um deles. Já em um BD tradicional, todas as ocorrências de um mesmo tipo de dado estão descritas uma única vez em um estrutura independente, mais estável e mais reduzida. Como existe regularidade de representação, é mais fácil prescrever restrições de integridade semânticas aplicáveis a todas as ocorrências de dados.

### 1.3 Conteúdo

Este tutorial apresenta o estado da arte na área de dados semi-estruturados, detalhando os tópicos de pesquisa mais relevantes nos capítulos posteriores. O capítulo dois é dedicado à modelagem de dados semi-estruturados, apresentando dois modelos de representação: OEM e

XML. O capítulo três aborda linguagens de consulta, descrevendo requisitos para consulta a dados semi-estruturados e duas linguagens para XML: XML/QL e XQL. O capítulo quatro apresenta técnicas extração de dados semi-estruturados e algumas propostas inseridas nestas técnicas. O capítulo cinco introduz o conceito de ontologia e também algumas propostas de sua aplicação no gerenciamento de dados semi-estruturados. O capítulo seis é reservado às considerações finais.

## 2 Modelagem de Dados

Modelos de dados para BDs tradicionais não são adequados a dados semi-estruturados, uma vez que todas as ocorrências de dados devem apresentar a mesma estrutura. Assim, modelos de dados para dados semi-estruturados devem ser flexíveis no sentido de suportar representações heterogêneas de dados semanticamente iguais.

A forma mais usual de modelar dados semi-estruturados é através de **grafos direcionados rotulados**, onde os vértices representam objetos identificáveis e as arestas são arcos para outros objetos que fazem parte da sua estrutura, que é hierárquica. Um rótulo presente em um arco indica o nome do atributo do objeto de onde o arco parte (objeto origem), podendo ainda informar o tipo de dado do objeto alcançável através do arco (objeto destino). Como cada ocorrência de dado pode ter uma estrutura diferente, não deve haver restrição no número de arcos que partem de um objeto origem. Todo objeto semi-estruturado tem um vértice raiz que é o ponto de partida para a investigação da sua estrutura hierárquica [Bun97, Flo98].

A figura 2.1 ilustra um exemplo de um objeto semi-estruturado (a) e sua representação em um modelo de grafo (b). O objeto em questão é um documento que descreve um artigo. Tal artigo apresenta um título, autores, instituição e endereço eletrônico dos autores, resumo e um conjunto de seções (introdução, conclusão, etc). O grafo em (b) mostra uma das possíveis alternativas de representação da composição deste artigo. Pode-se imaginar uma raiz única para uma base de documentos semi-estruturados (indicado pelo objeto R) e, na seqüência, ligações para tipos de documentos, sendo um deles o artigo em questão. Os números presentes nos vértices podem ser imaginados como identificadores de objetos. Tipos de dados para os arcos, apesar de não mostrados no grafo em (b), poderiam ser definidos, como por exemplo, *string*, para o arco Título.

O modelo de grafo é apropriado para representar dados semi-estruturados pois a estrutura particular de cada dado já está embutida na próprio grafo. Neste sentido, os arcos têm uma importância fundamental, uma vez que nomeiam os atributos e tipos que fazem parte da estrutura do dado [Suc97].

### 2.1 OEM

**OEM** (*Object Exchange Model*) é um modelo de dados pioneiro para dados semi-estruturados proposto no projeto Tsimmis, do Departamento de Ciência da Computação da Universidade de Stanford [Pap95, Ham97]. OEM foi definido com o objetivo de ser um modelo simples e de propósito geral para representar dados de fontes semi-estruturadas.

OEM baseia-se na noção de **objeto com identidade**<sup>1</sup>, sendo os objetos os vértices do grafo. Objetos podem ser atômicos ou complexos. No primeiro caso, existem tipos predefinidos como *integer*, *string*, *gif* e *java*. No segundo caso, é definido um tipo conjunto de

---

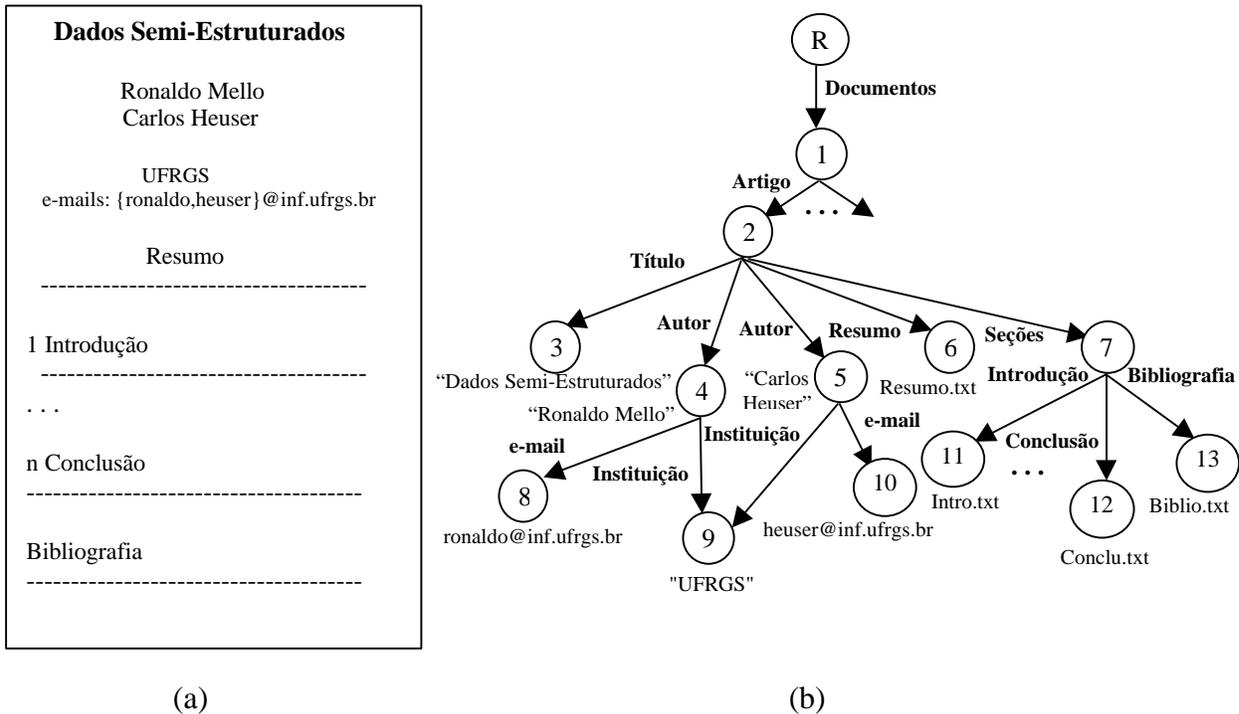
<sup>1</sup> Identidade de um objeto é conhecida como *Object IDentity* (OID).

referências a objetos, sendo os rótulos do tipo *string*. Objetos que são apenas destino de arcos possuem tipos atômicos. Objetos origem são sempre complexos [Abi97a]. A representação apresentada na figura 2.1 (b) está de acordo com este modelo.

A especificação de um objeto em OEM apresenta o seguinte esquema:

```
<rótulo, tipo, valor, OID>
```

Rótulo é uma *string* de tamanho variável que descreve o que o objeto representa. Tipo pode ser um tipo atômico ou conjunto de objetos. Valor é um campo de tamanho variável que mantém um valor compatível com o tipo definido e OID é a identificação única do objeto ou *null*. O OID pode ser omitido a nível de modelagem conceitual, uma vez que é controlado por SGBDs.



**Figura 2.1** - Exemplo de um objeto semi-estruturado (a) e sua representação em um modelo de dados baseado em grafo (b)

A figura 2.2, a seguir, especifica um objeto atômico (a) e um objeto complexo (b) em OEM, com base no objeto da figura 2.1 (b).

```
<Instituição, string, "UFRGS">
    (a)

<Seções, set, {intro, conclu, biblio}>
    intro: <Introdução, texto, "Intro.txt">
    conclu: <Conclusão, texto, "Conclu.txt">
    biblio: <Bibliografia, texto, "Biblio.txt">
    (b)
```

**Figura 2.2** – Exemplo de uma definição de objeto atômico (a) e complexo (b) em OEM

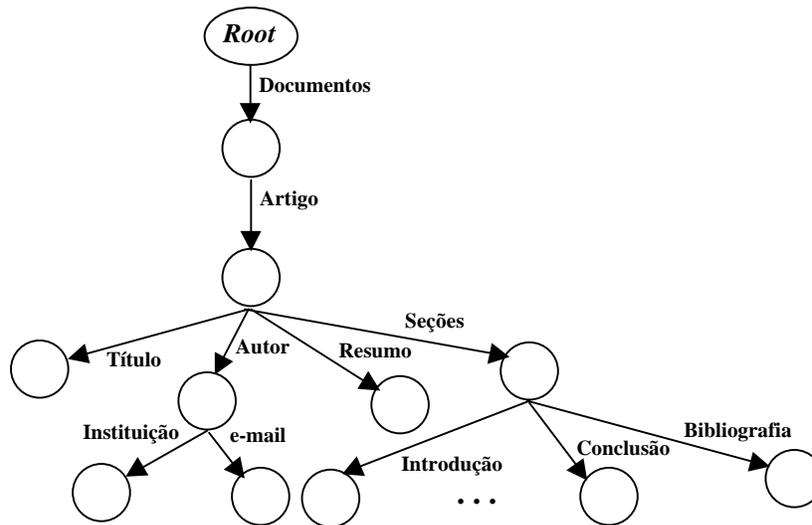
O objeto atômico *Instituição* representa a *string* “UFRGS”. Já o objeto *Seções* é composto pelos objetos *Introdução*, *Conclusão* e *Bibliografia*, cujas referências são respectivamente *intro*, *conclu* e *biblio*.

OEM é um modelo flexível pois não considera a existência de um esquema fixo para um conjunto de dados. Toda informação esquemática está incluída nos rótulos, que podem mudar dinamicamente. OEM é considerado um modelo para instâncias de dados semi-estruturados pois representa valores de dados e associa rótulos a cada valor para descrever o seu significado.

Uma extensão para representar apenas esquemas de dados semi-estruturados baseada em OEM são os chamados *data guides*. Um *data guide* é um sumário preciso e conciso da estrutura de um conjunto de instâncias semi-estruturadas, sendo representado como um objeto OEM [Hug97].

*Data guide* é um conceito similar à metadado em um BD tradicional, podendo ser consultado à nível de usuário ou aplicação para se descobrir a estrutura de um conjunto de objetos semi-estruturados. A idéia é que a estrutura de qualquer objeto do conjunto esteja representada no *data guide* e que o *data guide* não contenha estruturas que não existam no conjunto de objetos - atua como um esquema à *posteriori*.

A figura 2.3 mostra um *data guide* que representa a estrutura do objeto *Artigo* da figura 2.1 (b). Um *data guide* não coloca restrições no número de arcos de um dado tipo. Isto significa que um objeto do tipo *Artigo* pode ter diversos subobjetos do tipo *Autor*, por exemplo.



**Figura 2.3** - Um *data guide* para um objeto semi-estruturado do tipo *Artigo*

A vantagem dos *data guides* está na formulação e processamento de consultas, permitindo tarefas como a especificação de expressões de caminho a serem pesquisadas, análise sintática e determinação de planos de acesso. Esta última tarefa pode ser facilitada pela inclusão opcional de **anotações** a um tipo de objeto descrito em um *data guide*. A indicação de índices para certos tipos de objetos é um exemplo de anotação.

## 2.2 XML

**XML** (*eXtensible Markup Language*) é um padrão para publicação, combinação e intercâmbio de documentos multimídia, desenvolvido pelo consórcio W3C (*World Wide Web*

*Consortium*) [XML9?, Bra00]. Assim como outras linguagens de marcação, XML lida com instruções embutidas no corpo de documentos chamadas *tags*, que permitem a descrição de dados. XML tem como base linguagens mais antigas como SGML e HTML, sendo atualmente empregada na representação de estruturas de dados estruturados e semi-estruturados e seu intercâmbio na *Web*.

### 2.2.1 DTD

O principal conceito de XML é o **elemento**, que descreve uma unidade atômica ou não-atômica de dado. Um ou mais elementos podem estar definidos previamente através de uma DTD (*Document Type Definition*), que define um padrão para marcação de dados em documentos através da definição de uma hierarquia de elementos, onde um elemento é a raiz desta hierarquia. Para que um documento XML esteja de acordo com uma DTD, apenas os elementos e as estruturas de aninhamento entre elementos definidas na DTD são permitidos no corpo do documento - esta validação é feita por *parsers* XML. A figura 2.4 apresenta parte da descrição de uma DTD (a) e de um documento XML que segue a descrição desta DTD (b) para um domínio de artigos científicos do evento SBBD.

Em uma DTD, uma definição de elemento pode ser atômica ou complexa. No primeiro caso, o elemento possui apenas um conteúdo textual. No segundo caso, o elemento agrega elementos componentes. Elementos componentes podem ser definidos como obrigatórios ou opcionais e também podem se repetir. Elementos podem ainda ter atributos. Atributos devem pertencer a um tipo de dado<sup>2</sup> e podem ter um valor *default*.

A figura 2.4 (a) exemplifica parte de um arquivo de especificação de DTD chamado SBBD.dtd. O elemento não-atômico SBBD é a raiz da hierarquia, sendo composto por um conjunto de artigos. Cada elemento artigo, por sua vez, é composto por um título, um resumo e um ou mais (indicado pelo símbolo '+') autores e seções. Título e resumo são elementos atômicos (tipo #PCDATA). Um autor possui: um nome; uma ou mais instituições; e zero ou mais (indicado pelo símbolo '\*') e-mails. Uma instituição possui um nome e opcionalmente (indicado pelo símbolo '?') um endereço. O elemento SBBD possui dois atributos (cláusula ATTLIST) do tipo *string* (tipo #CDATA): ano (obrigatório - cláusula #REQUIRED) e local (com valor *default* igual a Rio).

### 2.2.2 XML Schema

**XML Schema** é uma proposta da W3C para descrever a estrutura de um documento XML. XML Schema é um padrão mais abrangente que uma DTD, permitindo expressar tipos de dados, herança, tipos abstratos, unicidade e chaves, entre outros [W3C 00a]. A figura 2.5 mostra um esquema exemplo descrito em XML Schema igualmente para o domínio de artigos científicos.

Na seqüência são apresentadas algumas características mais importantes deste padrão.

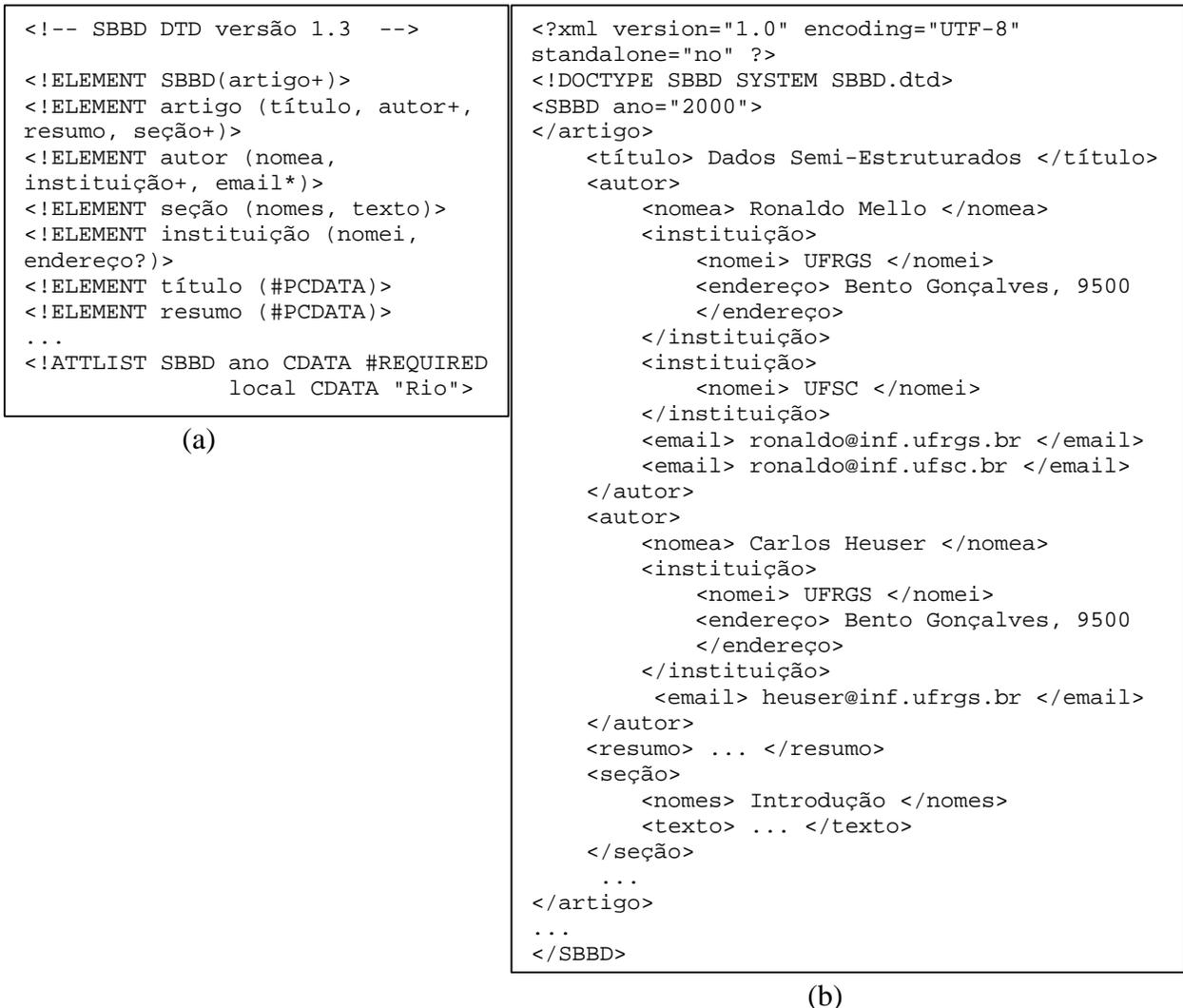
#### 2.2.2.1 Sintaxe Básica

Uma especificação em XML Schema sempre inicia com a *tag* <schema> e termina com a *tag* </schema>. Todas as declarações de elementos, atributos e tipos devem ser inseridas entre estas duas *tags*. Tipos, que representam a estrutura de uma classe de

---

<sup>2</sup> DTDs suportam alguns tipos de dados derivados de *strings*.

documentos e seus relacionamentos com outras classes, podem ser definidos. Um tipo pode ser simples (*simpleType*) ou complexo (*complexType*). Um *simpleType* é um tipo básico como *string*, *date*, *float*, *double*, *timeDurations*, etc. Um *complexType* define a estrutura de um elemento, ou seja, define características como subelementos, atributos, cardinalidades dos subelementos e obrigatoriedade dos atributos. A figura 2.5 mostra um exemplo de declaração de um *complexType* *tArtigo* e, mais adiante, uma declaração `<element name="artigo" type="tArtigo">`. Esta segunda declaração liga o nome *artigo* ao *complexType* *tArtigo*, indicando que, em uma instância de um documento XML que segue este esquema, deve-se ter um elemento *artigo* com subelementos *título*, *autor*, *resumo* e *seção*, como mostrado na figura 2.6. As cardinalidades mínima e máxima são indicadas pelos atributos *MinOccurs* e *MaxOccurs*, respectivamente.



**Figura 2.4** - Uma definição de DTD (a) e um documento XML que segue esta definição (b)

*ComplexTypes* podem ter atributos, que são declarados através da *tag* `<attribute>` e devem ser do tipo *simpleType*, como é o caso do atributo *data* de *tArtigo*, no esquema da figura 2.5. Um atributo pode ser declarado como obrigatório ou opcional através da cláusula *use*. Os valores permitidos para esta cláusula são *required* (obrigatório), *optional*

(opcional) ou *fixed* (fixo). No último caso, deve-se dizer o valor *default* do atributo utilizando a cláusula *value*.

Ainda, pode-se restringir o conteúdo de um elemento através do uso de um atributo chamado *content*, que pode assumir os seguintes valores: *textOnly* (apenas texto); *elementOnly* (apenas subelementos); *mixed* (texto e subelementos); ou *empty* (conteúdo vazio).

### 2.2.2.2 Derivação de Tipos

XML *Schema* possui um mecanismo de derivação de tipos, permitindo a criação de novos tipos a partir de outros já existentes. Isto pode ser feito de duas maneiras: por restrição ou por extensão. Tipos simples só podem ser derivados por restrição, aplicando-se “facetas” a um tipo básico ou utilizando uma linguagem de expressões regulares [W3C 00]. A figura 2.7 mostra a derivação de um tipo simples chamado *MeuInteiro* através da aplicação de facetas que restringem o valor mínimo e máximo de um número inteiro.

```
<?xml version="1.0"?>
<schema xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  xmlns:grp="http://meunamespace.com/Artigo"
  targetNamespace="http://meunamespace.com/Artigo">
  <complexType name="tArtigo">
    <group> <sequence>
      <element name="título" type="string" minOccurs='1' maxOccurs='1' />
      <element ref="autor" minOccurs='1' maxOccurs='*' />
      <element name="resumo" type="string" minOccurs='1' maxOccurs='1' />
      <element ref="seção" minOccurs='1' maxOccurs='*' />
    </sequence> </group>
    <attribute name="data" type="date" use="optional" />
  </complexType>
  <complexType name="tAutor">
    <element name="nomea" type="string" minOccurs='1' maxOccurs='1' />
    <element name="instituição" type="tInst" minOccurs='1' maxOccurs='*' />
    <element name="email" type="string" minOccurs='0' maxOccurs='*' />
  </complexType>
  <complexType name="tSeção">
    <element name="nomes" type="string" minOccurs='1' maxOccurs='1' />
    <element name="texto" type="string" minOccurs='0' maxOccurs='*' />
  </complexType>
  <complexType name="tInst">
    <element name="nomei" type="string" minOccurs='1' maxOccurs='1' />
    <element name="endereço" type="string" minOccurs='0' maxOccurs='1' />
  </complexType>
  <element name="artigo" type="tArtigo" />
  <element name="autor" type="tAutor" />
  <element name="seção" type="tSeção" />
</schema>
```

**Figura 2.5** – Especificação de um esquema em XML *Schema*

Tipos complexos podem ser derivados por restrição ou por extensão. A derivação por restrição permite, por exemplo, restringir a cardinalidade de um subelemento. A derivação por extensão adiciona características a um tipo, sendo semelhante ao conceito de herança. Um exemplo é mostrado na figura 2.8, onde o *complexType* *tAutorExtendido* acrescenta ao tipo *tAutor* um elemento *endereço*, com cardinalidades mínima igual a 0 e máxima igual a N.

### 2.2.2.3 Grupos

Grupos especificam restrições sobre um conjunto fixo de subelementos, podendo ser de três tipos: *sequence*, *choice* e *all*. Um grupo *sequence* estabelece que todos os elementos pertencentes a ele devem aparecer na ordem em que foram definidos e nenhum pode ser omitido. O grupo *choice* estabelece que apenas um dos elementos pertencentes ao grupo deve aparecer em uma instância XML. Já o grupo *all* diz que os elementos podem aparecer em qualquer ordem e podem ser repetidos ou omitidos.

Um exemplo de definição de um grupo do tipo *sequence* pode ser vista na figura 2.5 para o *complexType* *artigo*.

```
<?xml version="1.0"?>
<artigo xmlns="artigo.xsd">
  <título> Dados Semi-Estruturados </título>
  <autor>
    <nomea> Ronaldo Mello </nomea>
    <instituição> <nomei> UFRGS </nomei>
      <endereço> Bento Gonçalves, 9500 </endereço>
    </instituição>
    <instituição> <nomei> UFSC </nomei>
    </instituição>
    <email> ronaldo@inf.ufrgs.br </email>
  </autor>
  <autor>
    <nomea> Carlos A. Heuser </nomea>
    <instituição> <nomei> UFRGS </nomei>
      <endereço> Bento Gonçalves, 9500 </endereço>
    </instituição>
    <email> heuser@inf.ufrgs.br </email>
  </autor>
  <resumo> ... </resumo>
  <seção> <nomes> Introdução </nomes>
    <texto> ... </texto> </seção>
  ...
</artigo>
```

Figura 2.6 - Exemplo de um documento XML

```
<simpleType name="MeuInteiro" base="integer">
  <minInclusive value="1"/>
  <maxInclusive value="20"/>
</simpleType>
```

Figura 2.7 - Derivação de um tipo simples por restrição

```
<complexType name="tAutorExtendido" base="tAutor" derivedBy="extension">
  <element name="endereço" type="string" minOccurs='0' maxOccurs='*'/>
</complexType>
```

Figura 2.8 - Derivação de um tipo complexo por extensão

### 2.2.2.4 Referências

Uma declaração de atributo, elemento ou grupo pode ser referenciada, permitindo a reutilização de declarações, como a declaração `<element ref="autor" minOccurs='1' maxOccurs='*'/>` dentro do *complexType* *artigo*, na figura 2.5. A única restrição no uso

de referências é que o elemento referido seja global, ou seja, tenha sido declarado dentro de `<schema>`, porém, não dentro de um *complexType*.

### 2.2.2.5 Namespaces

Um esquema especificado em XML *Schema* pode ser visto como um conjunto de declarações de tipos e elementos cujos nomes pertencem a um **namespace** [W3C 99]. Todo esquema em XML *Schema* deve definir um único namespace, sendo usual o formato de URL (*Uniform Resource Locator*) para a sua identificação. As linhas de 2 a 4 na figura 2.5 definem um namespace chamado `http://meunamespace.com/Artigo` para o esquema apresentado.

O uso de namespaces aumenta a flexibilidade de XML *Schema*, permitindo a reutilização de definições feitas em outros esquemas. A utilização de um tipo definido em outro esquema é possível através da sua declaração e da associação de um prefixo a ele.

## 3 Linguagens de Consulta

A crescente utilização de documentos semi-estruturados para representação e intercâmbio de informações, principalmente por meio da *Web*, evidencia a necessidade de ferramentas de consulta mais abrangentes e mais eficientes do que aquelas baseadas em mecanismos de *full-text search*, que utilizam busca por palavras-chaves. Estes mecanismos de consulta são úteis para documentos não-estruturados, porém, são inadequados para documentos estruturados ou semi-estruturados, nos quais a estrutura subjacente contém informações referentes aos dados armazenados.

Neste contexto, o principal objetivo das linguagens para dados semi-estruturados é consultar conjuntos de documentos como se fossem um BD, permitindo consultas mais eficientes e eficazes. Para tanto, o primeiro passo na definição destas linguagens é estabelecer conjuntos de requisitos desejáveis a uma linguagem de consulta, que sirvam como guia no seu processo de desenvolvimento.

### 3.1 Requisitos para Linguagens de Consulta a Dados Semi-Estruturados

Os requisitos desejáveis para uma linguagem de consulta para dados semi-estruturados, em um contexto genérico, são apresentados por [Bun97] e [Abi97a], sendo que outros autores os descrevem dentro de contextos mais específicos, tais como XML [W3C 00b] e XML com BDs [Mai98]. Em um contexto genérico, os principais requisitos são:

- Habilidade de executar consultas sem conhecimento do esquema ou sobre o próprio esquema, se este existir;
- Capacidade de lidar com a heterogeneidade de tipos de atributos, que podem ocorrer em objetos semanticamente iguais;
- Utilização de operadores de consulta de linguagens tradicionais de BDs, tais como seleção e projeção (passíveis de otimização);
- Navegação no estilo hipertexto e busca por padrão;
- Consultas temporais;
- Uso de uma abordagem baseada em objetos, uma vez que dados semi-estruturados são complexos;
- Uso de coerção (critérios de conversão de valores), em função da heterogeneidade de tipo e de estrutura dos dados semi-estruturados;

- Suporte à especificação de expressões de caminho (*path expressions*), a fim de se navegar pela estrutura de dados semi-estruturados e documentos;
- Busca baseada em estrutura, que permite restringir o escopo da consulta àqueles elementos que possuem uma determinada estrutura hierárquica implícita ou explícita.

Os requisitos apresentados em [Mai98] serviram como inspiração para outros requisitos, entre os quais, os de [Abi00] e [W3C 00b]. Em [Abi00], sobre os requisitos apontados em [Mai98], afirma-se que muitos deles são completamente independentes do formato XML e podem também ser aplicados a qualquer linguagem de consulta para dados semi-estruturados, apesar de não se ter clareza quanto a forma de empregá-los no desenvolvimento de uma linguagem. Estes requisitos são:

- Poder de expressão: apesar de ser possível relacionar uma série de operações desejáveis a uma linguagem de consulta para dados semi-estruturados em enfoques específicos, ainda não está bastante claro o poder de expressão que uma linguagem de consulta para dados semi-estruturados deve possuir;
- Semântica: deve haver uma semântica precisa, sem a qual não se pode discutir transformação ou otimização de consultas. Uma questão interessante é que tipo de semântica (se é que há alguma) se pode extrair da sintaxe na qual os dados estão expressos;
- Composição: o resultado de uma consulta deve poder ser utilizado como entrada por outra consulta, o que é essencial para, por exemplo, construir visões;
- Esquema: dados semi-estruturados podem ou não possuir uma definição de esquema explícita. Caso não exista esta definição, a linguagem de consulta deve ser capaz de identificá-lo a partir da estrutura dos documentos. Assim, sendo a estrutura predefinida ou inferida, a linguagem de consulta deve ser capaz de explorá-la, para verificação de tipos, otimização, etc;
- Manipulação programática: deve-se sempre ter em mente que, freqüentemente, as expressões de consulta são geradas a partir de programas e não escritas por programadores. Neste caso, uma linguagem básica é mais apropriada do que uma linguagem de fácil entendimento para o usuário.

### 3.2 Linguagens Propostas

O crescente interesse pelo acesso a dados semi-estruturados tem gerado diversas propostas de linguagens, visando, principalmente, recuperar dados *Web*. Várias delas estão voltadas para a consulta a dados em XML, considerando a tendência de que XML venha a ser adotada como padrão para o intercâmbio de informações na *Web*. Duas gerações podem ser identificadas na evolução das linguagens de consulta para a *Web* [Flo98]:

- Primeira geração: combinam consultas baseadas em conteúdo (típico de *search engines*) e baseadas em padrões estruturais. Exemplos de linguagens desta geração são WebSQL [Men97], W3QL [Kon95], WebLog [Lak96] e WQL [Li99];
- Segunda geração: baseiam-se na estrutura interna de objetos semi-estruturados e documentos, permitindo a criação de novas estruturas como resultado da consulta. Dentre as linguagens desta geração, estão a WebOQL [Aro98] e a StruQL [Fer97].

Como evolução natural da segunda geração estão as linguagens de consulta para XML, tais como Lorel [Abi97b], XML-QL [Deu99] e XQL [Rob98]. Neste grupo de linguagens percebem-se dois paradigmas [Bar98]:

1. Paradigma de BD, que apresenta linguagens *SQL-like* ou *OQL-like*, dos quais podem-se citar XML-QL e Lorel;
2. Paradigma de programação funcional, baseado em XSL (*eXtensible Stylesheet Language*) [Mar00], cuja principal expoente é XQL<sup>3</sup>.

Com o objetivo de comparar estes paradigmas, duas linguagens são detalhadas a seguir: XML-QL e XQL. Estas linguagens estão em estudo pelo W3C para compor o padrão a ser proposto como linguagem de consulta para a *Web*.

### 3.2.1 XML-QL

Consultas em XML-QL podem extrair porções de dados de documentos XML e realizar transformações como mapeamento de dados XML entre DTDs e integração de dados XML de diferentes fontes [Deu99]. Além disso, XML-QL também suporta operações *data-intensive*, tais como *joins* e agregações, oferecendo um bom suporte para construir novos documentos XML.

O modelo de dados proposto para XML/QL é apresentado em duas versões: o não-ordenado e o ordenado. O modelo de dados não-ordenado representa um documento XML na forma de grafo, em que cada vértice é representado por um OID (*string* única). No grafo, os arcos são rotulados com as *tags* dos elementos XML, os vértices com conjuntos de pares *<atributo,valor>* e as folhas com uma *string* de valor. O grafo possui ainda um vértice *raiz*.

O modelo de dados ordenado é, em essência, semelhante ao não ordenado, com a distinção de que os seus elementos seguem uma ordenação total (grafo ordenado). A principal implicação desta ordenação é que pode haver mais de um arco com o mesmo nome entre um objeto origem e um objeto destino, o que não é permitido no grafo não-ordenado. Outra diferença se refere à complexidade das consultas, que é maior no grafo ordenado em função das restrições de ordenação.

Consultas XML-QL podem tanto **recuperar** dados de documentos XML quanto **construir** novos documentos XML. O formato geral da consulta é `WHERE seleção IN origem CONSTRUCT [resultado]`, onde *seleção* indica os dados a serem buscados nos documentos XML, *origem* é o endereço URL dos documentos, e *resultado* é o documento XML construído.

A base das consultas em XML-QL são **padrões de elementos**, como se pode observar no exemplo da figura 3.1, que produz como resultado uma lista composta por todos os títulos dos artigos cujo sobrenome do autor seja Heuser e que estão armazenados em um documento localizado no endereço `www.a.b.c/bib.xml`. A fim de facilitar a escrita da consulta, a linguagem simplifica as *tags* que delimitam o final do elemento, de forma que uma *tag* `</element>` seja escrita na forma `</>`.

No exemplo da figura 3.1, as variáveis *t* e *r* estão relacionadas respectivamente aos elementos *título* e *resumo*. Como resultado, é produzida uma lista de autores ligados a *t* e de resumos ligados a *r*. As variáveis *t* e *r* são precedidas pelo sinal `$`, a fim de distingui-las

---

<sup>3</sup> Outra alternativa para consultar XML com base em XSL é utilizando XSLT e Xpath [Mar00]. Entretanto, esta solução não é apresentada por não se constituir exatamente em uma linguagem de consulta.

dos valores de conteúdo do documento. O CONSTRUCT externo produz o documento XML resultado, que tem artigo como elemento raiz.

```

CONSTRUCT <result> {
  WHERE <artigo>
    <nomea>Heuser</>
    <título>$t</>
    <resumo>$r</>
  </> IN "www.a.b.c/bib.xml"
  CONSTRUCT <artigo>
    <título>$t</>
    <resumo>$r</>
  </> }

```

**Figura 3.1** – Consulta XML/QL básica

Consultas também podem ser aninhadas, a fim de agrupar os elementos dentro de um contexto específico. O exemplo da figura 3.2 produz como resultado uma lista contendo os títulos dos artigos, cada título, por sua vez, contendo todos os seus autores.

```

CONSTRUCT <result> {
  WHERE <artigo>
    <título>$t</>
    </> CONTENT_AS $art
  IN "www.a.b.c/bib.xml"
  CONSTRUCT <artigo>
    <título>$t</>
    {
      WHERE <autor>$aut</> IN $art
      CONSTRUCT <autor>$aut</>
    }
  </> }

```

**Figura 3.2** – Exemplo de consulta aninhada em XML-QL

A cláusula CONTENT\_AS armazena o conteúdo do elemento que o precede, atribuindo-o à variável \$art, que é, na seqüência, utilizada como origem para a consulta aninhada que produz a lista de autores de cada artigo.

XML-QL também pode expressar *joins*, combinando dois ou mais elementos que contêm o mesmo valor. Para exemplificar esta facilidade, acrescenta-se um elemento livro ao domínio de artigos científicos, cuja DTD que é o descreve é:

```

<!ELEMENT livro (título, autor+, ISBN, editora)>
<!ELEMENT editora (nomee, endereço)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT nomee (#PCDATA)>

```

Assim, a consulta da figura 3.3 retorna todos os artigos que possuam pelo menos um autor que já escreveu um livro. Neste exemplo, a variável naa é utilizada para manter nomes de autores presentes tanto em elementos <artigo> quanto em elementos <livro>, exigindo uma igualdade de valor entre ambos os elementos. O resultado da consulta é uma lista de artigos.

Consultas sobre modelos de dados XML ordenados podem definir condições que consideram a ordem existente entre os elementos. Supondo que autor tenha o atributo sobrenomea, a consulta mostrada na figura 3.4 retorna autores cujo sobrenome precede o primeiro nome.

```

CONSTRUCT <result> {
  WHERE <artigo>
    <autor><nomea>$naa</></>
    </>
    </> CONTENT_AS $a IN "www.a.b.c/bib.xml",
    <livro>
      <autor><nomea>$naa</>
      </>
      </> IN "www.a.b.c/bib.xml"
  CONSTRUCT <artigo>$a</> }

```

**Figura 3.3** – Exemplo de consulta com *join* em XML-QL

```

CONSTRUCT <result> {
  WHERE <autor> $a
    </> in "www.a.b.c/bib.xml",
    <nomea [$i]> </> in $a,
    <sobrenomea [$j]> </> in $a,
    $j < $i
  CONSTRUCT <autor> $a </> }

```

**Figura 3.4** – Exemplo de consulta XML-QL sobre um modelo de dados ordenado

Uma fraqueza a salientar em relação à XML/QL é que não há garantia na ordem do resultado. Um processador XML-QL não garante que a ordem dos elementos no documento resultante seja a mesma do documento de origem. Isto não representa um grande problema para documentos de dados, mas é sério para documentos de texto.

### 3.2.2 XQL

XML *Query Language* (XQL) é uma proposta para estender os padrões de XSL (*eXtensible Stylesheet Language*)<sup>4</sup>, a fim de expressar consultas a documentos XML [Rob98]. Embora pareça não haver relação entre uma linguagem de definição de estilos de apresentação e uma linguagem de consulta, uma "folha de estilos" XSL de fato transforma um documento ao acrescentar instruções de formatação, criando um novo documento cuja estrutura pode ser totalmente diferente da original. Estas semelhanças entre uma linguagem de consulta e XSL são resumidas na tabela 3.1 [Xwg98]. Apesar da linguagem utilizada como exemplo ser SQL, os recursos apresentados são comuns a quaisquer outras linguagens de consulta a dados.

**Tabela 3.1** – Semelhanças entre os recursos de SQL e XSL

	<b>Recursos de SQL</b>	<b>Recursos de XSL</b>
<b>Recuperação de dados</b>	<ul style="list-style-type: none"> <li>• SELECT identifica valores a recuperar;</li> <li>• WHERE identifica restrições de recuperação.</li> </ul>	<ul style="list-style-type: none"> <li>• padrões de seleção identificam vértices de um objeto semi-estruturados a considerar;</li> <li>• qualificadores de padrões especificam critérios para os vértices.</li> </ul>
<b>Construção</b>	<ul style="list-style-type: none"> <li>• consultas retornam tabelas virtuais;</li> <li>• ORDER BY ordena as linhas retornadas.</li> </ul>	<ul style="list-style-type: none"> <li>• folhas de estilo criam novos documentos XML;</li> <li>• Xsl:sort ordena os elementos criados.</li> </ul>

Consultas em XQL são escritas em uma sintaxe semelhante àquela utilizada para navegar em diretórios de um sistema operacional, porém, tomam como base a estrutura

<sup>4</sup> XSL é um padrão para apresentação de documentos XML.

hierárquica de um documento XML. Uma consulta que deseja obter todos os elementos que são artigos, por exemplo, é escrita simplesmente como `artigo`. Para obter o título de todos os artigos, a consulta é `artigo/título`.

XQL lida com o conceito de **contexto**. Por contexto entende-se o escopo de atuação da consulta dentro da hierarquia do documento XML. Assim, o resultado de uma consulta em que se procura o elemento autor é diferente quando o contexto é a hierarquia enraizada em um vértice artigo ou quando o contexto é a raiz de uma hierarquia de documentos, por exemplo. No último caso, a consulta é efetuada sobre todo documento XML, enquanto que no primeiro não.

A tabela 3.2 apresenta outros exemplos de consultas XQL sobre o domínio de artigos científicos, com os respectivos resultados [Rob98]. O símbolo “@” é utilizado para indicar atributos.

Além dos exemplos apresentados na tabela 3.2, XQL possui outros recursos, tais como expressões booleanas, equivalência, métodos para manipulação de coleções, indexação dentro de uma coleção e métodos de agregação.

**Tabela 3.2 – Exemplos de consultas XQL**

<b>Consulta</b>	<b>Resultado</b>
<code>Artigo/*</code>	Seleciona todos os sub-elementos de <code>artigo</code>
<code>Artigo//texto</code>	Seleciona todos os elementos <code>texto</code> , independente do nível de descendência de <code>artigo</code>
<code>Artigo/*/texto</code>	Seleciona todos os elementos <code>texto</code> que são “netos” de <code>artigo</code>
<code>Artigo[autor]</code>	Seleciona todos os elementos <code>artigo</code> que possuam um sub-elemento <code>autor</code>
<code>Artigo[@autor]</code>	Seleciona todos os elementos <code>artigo</code> que possuam um atributo <code>autor</code>
<code>//texto</code>	Seleciona todos os elementos <code>texto</code> , em qualquer nível de profundidade, dentro do documento XML

### 3.2.3 Comparação de XQL com XML/QL

Comparações prévias entre XQL e XML-QL afirmam que:

- Em essência, os recursos descritos por XML-QL são muito similares àqueles fornecidos pelas linguagens de transformação e de padrões XSL, na qual XQL é baseada. Ambas as abordagens são estruturadas em blocos e orientadas a modelos e ambas oferecem a capacidade de retornar estruturas na forma de árvores ou grafos, criar novos elementos na saída e consultar XML. As principais diferenças são a nível sintático [Sch98];
- Há dois pontos em que as funcionalidades de XML/QL e XQL se assemelham: ambas recuperam dados de uma fonte e constroem uma nova fonte de dados a partir dos dados recuperados [Xwg98].

Apesar destas similaridades, existe uma série de características de XML/QL que não são atualmente suportadas por XQL, como variáveis de consulta, OIDs, integração de dados de múltiplas fontes XML e formatação do documento resultado [Rob98]. Uma vez que XQL não suporta variáveis, consultas envolvendo *joins* não são possíveis. Este fato aliado à impossibilidade de se gerar uma estrutura de resultado para o documento de saída reduz o poder de expressão desta linguagem.

Em suma, pode-se considerar como principal diferença entre XML/QL e XQL o fato de que a primeira é voltada à consulta a dados XML e a segunda é voltada à consulta a documentos XML.

## 4 Extração de Dados Semi-Estruturados

O interesse em extrair dados semi-estruturados, como dados *Web*, e armazená-los em BDs, tem aumentado significativamente devido a ineficiência das ferramentas de consulta à *Web* disponíveis, como *browsers* e *search engines*. Mecanismos de extração de dados vêm sendo pesquisados e desenvolvidos para atender aos seguintes objetivos: i) extrair informações mais precisas e em número razoavelmente menor; ii) possibilitar que os resultados sejam armazenados de uma forma mais estruturada; iii) facilitar o processamento de consultas posterior ao processo de extração; e iv) fornecer resultados que estejam de acordo com a intenção do usuário [Ham98, Ade99, Lae99].

No gerenciamento de dados semi-estruturados, a extração de dados faz parte do processamento de uma consulta, além das etapas comumente executadas, como *parsing* e otimização [Bun96, Fer98]. A etapa de extração é responsável pela recuperação de informações contidas nas fontes de dados, porém, sem retornar os documentos completos que contenham estas informações. Apenas as informações solicitadas pelo usuário ou relevantes para o processamento da consulta devem ser extraídas, como ocorre em consultas a BDs convencionais, por exemplo.

Um processo de extração normalmente transforma dados semi-estruturados de uma fonte de dados para dados adequados a um modelo de dados, seja ele estruturado ou semi-estruturado. Este resultado é posteriormente processado de alguma forma: no caso de um dado estruturado, por exemplo, o resultado pode ser já materializado em um BD para fins de consulta; no caso de um conjunto de dados semi-estruturados, por exemplo, pode ser necessário um processo de *parsing* que verifica se cada ocorrência deste conjunto é uma resposta válida para a consulta.

Mediadores e *wrappers* são importantes neste contexto, uma vez que atuam como módulos intermediários entre aplicações e fontes de dados, sendo responsáveis pelas transformações de dados semi-estruturados. **Mediadores** são módulos que integram conhecimento presente em fontes heterogêneas de dados, produzindo informação útil para camadas superiores de um sistema de gerenciamento de dados [Wie92]. Um serviço de mediação apresenta uma regra para transformação de dados e contém estruturas de conhecimento que guiam estas transformações. Uma ampla gama de funcionalidades pode estar associada a um mediador, como reorganização lógica de dados, aplicação de métodos para obtenção de informações derivadas e aplicação de padrões de apresentação de dados. Um *wrapper* tem por função encapsular o acesso a uma fonte de dados, tornando-a utilizável de uma maneira mais conveniente que a sua representação original [Wra9?]. Um *wrapper* é diferente de um mediador. Um mediador integra dados de várias fontes de dados, enquanto um *wrapper* provê uma interface simplificada para uma única fonte de dados ou mais de uma fonte que tenham uma interface comum de acesso. Um *wrapper* pode ainda adicionar funcionalidade a uma fonte de dados, como por exemplo, prover uma camada de acesso para a um conjunto de documentos semi-estruturados, para facilitar a manipulação dos seus dados. Neste caso, o *wrapper* atua como um extrator de dados, identificando e adequando valores de dados com o esquema presente na aplicação.

Propostas de extratores na literatura podem ser classificadas em uma das seguintes técnicas: **extração semântica** e **extração sintática** [Dor00]. As características destas duas técnicas são discutidas a seguir.

## 4.1 Extração Sintática

O processo de **extração sintática** baseia-se na análise de padrões presentes em dados semi-estruturados, exigindo que documentos sejam percorridos por completo na busca da informação desejada. Esta análise pode ser realizada sobre *tags* de linguagens de marcação que compõem o documento ou estar baseada em **padrões** pré-determinados. No primeiro caso, as *tags* servem como delimitadores de regiões do texto, indicando o que se deve ser extraído. No segundo caso, padrões são definidos anteriormente à extração pelos usuários, indicando regiões de texto que devem ser extraídas.

A extração sintática baseada em *tags* se vale de programas simples que percorrem documentos em busca dos dados de interesse. Estes programas são escritos em linguagens de programação conhecidas, como a linguagem de *scripts* Perl [Ham98], ou em linguagens desenvolvidas especificamente para um extrator, com a finalidade de pesquisar e reestruturar dados de documentos [Atz97c, Cre98].

### 4.1.1 Extrator de TSIMMIS

O extrator do projeto TSIMMIS (*The Stanford IBM Manager of Multiple Information Sources*) segue uma abordagem sintática de extração de dados baseada em *tags*, recuperado informações de documentos HTML e gerando objetos OEM que são armazenados em um BD [Cha94, Ham98].

O extrator é um componente de um *wrapper* do sistema, funcionando como um *parser* que analisa um documento HTML com base em uma especificação definida manualmente pelo usuário. O programa de especificação consiste de comandos de extração. Um comando deve apresentar a seguinte forma:

[variáveis, fonte, padrão]

Fonte indica o texto de entrada a ser processado, padrão determina a regra de extração e variáveis define as variáveis que armazenam as informações extraídas. Variáveis definidas em um comando podem ser utilizadas em comandos seguintes. Cada comando é descrito entre colchetes.

Para exemplificar o processo de extração, considera-se uma aplicação que processa dados do domínio de artigos científicos, tendo como fonte de informação páginas HTML, como a mostrada na figura 4.1.

Clique no artigo para ler o resumo					
Autores	Artigo	Seção	Horário	Apresentador	Instituição
Mello, R.; Heuser, C	<a href="#">Integração de Fontes XML</a>	BD Heterogêneos	10:30	Mello, R.	UFRGS
Dorneles, C. ; Silva, A.	<a href="#">Caching XML Documents</a>	Armazenamento de dados	14:00	Dorneles, C.	UFRGS
Kade, A.; Becker, G.	<a href="#">Consulta a Documentos XML</a>	Linguagens de Consulta	16:30	Kade, A.	UPF

**Figura 4.1** - Página HTML alvo para o processo de extração

Com base na análise do código HTML deste documento, o usuário define um arquivo de especificação para ser posteriormente utilizado pelo programa extrator. Neste arquivo é especificada a URL do documento fonte, comandos para desconsiderar dados irrelevantes (como *tags* HTML e dados que não são relevantes) e comandos para associar informações a variáveis.

A figura 4.2 mostra o arquivo de especificação para a extração de dados do código HTML referente à página da figura 4.1. O processo de extração a ser realizado é o seguinte:

- **Comando 1:** busca o conteúdo do arquivo cuja URL está indicada no campo *fonte* e armazena na variável *root*. O caractere '#' significa extrair;
- **Comando 2:** solicita que o resultado da aplicação do conteúdo do campo *padrão* seja armazenado na variável *seção*. A regra de extração indica que se deve descartar tudo até a primeira ocorrência da *string* `</TR>` (o caractere '\*' significa descartar) na segunda definição de tabela e armazenar os dados entre `</TR>` e `</TABLE>`;
- **Comando 3:** divide o conteúdo da variável *seção* em porções de texto usando o *string* `<TR ALIGN=left>` como delimitador (separa, na verdade, as linhas da tabela). Os resultados são mantidos na variável *\_horaSeção* (o símbolo '\_' indica variável temporária – seu conteúdo não é considerado no resultado), que é uma variável do tipo lista. Futuras filtrações serão aplicadas a cada membro desta lista;
- **Comando 4:** extrai o conteúdo dos membros da lista para o *array* *hora\_Seção*, iniciando pelo segundo membro (posição 1 - elimina-se o cabeçalho da tabela). O valor 0 (segundo parâmetro) indica a última célula a ser incluída, contando-se do fim para o início da lista (extrai até o final da lista, neste caso);
- **Comando 5:** extrai valores individuais de cada posição do *array* (correspondentes às colunas da tabela), para variáveis que irão compor a resposta (*autores*, *artigo*, etc).

```
1  ["root",
2  "get('http://www.abcd.com.br/programa/'),
3  "#",
4  ],
5  ["seção",
6  "root",
7  "*<TABLE*<TABLE*</TR>#</TABLE>*"
8  ],
9  [_horaSeção",
10 "split(seção,'TR ALIGN=left')",
11 "#",
12 ],
13 [[hora_Seção ",
14 "[_horaSeção[1:0]",
15 "#",
16 ],
17 ["autores,artigo,seção,hora,apresentador,instituição",
18 "hora_Seção",
19 "*<TD>#</TD>*HREF=#>#</A>*<TD>*</TD>*<TD>#/#</TD>*<TD>*</TD>*<TD>#/#*"
20 ]]
```

**Figura 4.2** - Exemplo de arquivo de especificação para extração

O resultado do processo de extração é um objeto OEM (descrito em uma sintaxe particular) que contém todas as variáveis especificadas pelo usuário (exceto variáveis

temporárias), seu conteúdo e informações sobre a estrutura do objeto. A forma de estruturação hierárquica das informações segue a ordem de declaração das variáveis definida pelo usuário no arquivo de especificação. Parte do objeto OEM resultante da execução da especificação da figura 4.2 é mostrado na figura 4.3.

```
root complex {
  seção complex{
    hora_seção complex {
      autores string "Mello,R.; Heuser, C."
      artigo string "Integração de Fontes..."
      seção string "BD Heterogêneos"
      hora date 10:30
      apresentador string "Mello, R."
      Instituição string "UFRGS"
    }
    hora_seção complex {
      autores string "Dorneles, C.; Silva, A."
      ...
    }
    ...
  }
}
```

**Figura 4.3** - Objeto OEM gerado a partir do processo de extração

O módulo extrator de TSIMMIS apresenta a vantagem de permitir especificações reduzidas de extração e a possibilidade de integração de dados disponíveis na *Web*. Por outro lado, a especificação é inteiramente manual, exigindo que o usuário analise e descreva a estrutura da página HTML. Este processo é adequado a documentos que apresentam a mesma estrutura de *tags* e que não são alterados com frequência.

#### 4.1.2 Abordagens Semi-Automáticas

Além das informações contidas em documentos HTML, uma grande quantidade de dados semi-estruturados armazenados eletronicamente é considerada de interesse. Estas informações encontram-se na forma de arquivos texto, arquivos de *e-mail*, documentação de código, arquivos de configuração, etc. Neste caso, a extração dos dados é mais complexa pois, ao contrário de páginas HTML, estes arquivos não contêm indicadores de estrutura.

Neste sentido, abordagens semi-automáticas de extração de dados têm sido propostas através de ferramentas que auxiliam o usuário no processo de extração como o **Nodose** [Ade99]. Em Nodose, o usuário indica regiões do texto das quais ele deseja extrair determinados dados e a partir daí o processo automaticamente detecta, em documentos com comportamento descritivo similar, os mesmos dados relevantes. Evita-se assim, a codificação manual de um processo completo de extração, como é feito pelo extrator do TSIMMIS.

### 4.2 Extração Semântica

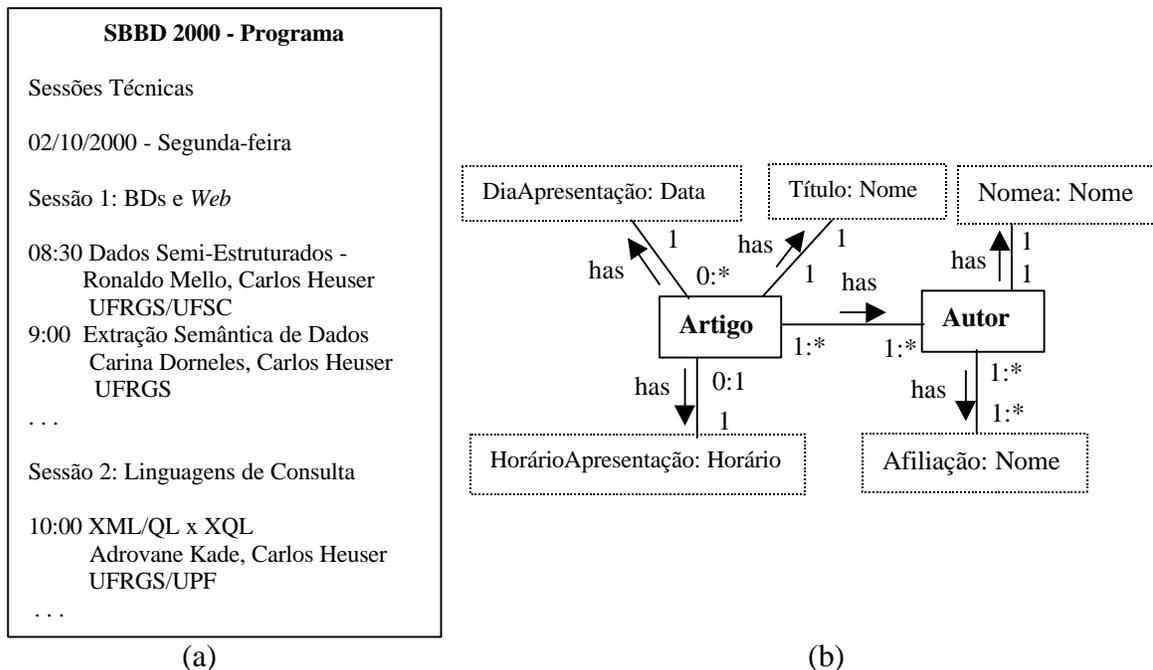
O principal problema associado aos processos de extração sintática é o fato de que, quando os documentos de uma fonte de dados são alterados, as regras de extração tornam-se inválidas. Além disso, informações semânticas não são consideradas, como por exemplo, as relações existentes entre os conceitos do domínio presentes nos documentos. Uma alternativa de extração que procura solucionar este problema é a chamada **extração semântica**.

Para fornecer informação semântica aos usuários, associam-se esquemas conceituais aos dados semi-estruturados que descrevam os conceitos de interesse presentes em um

domínio. A vantagem desta abordagem, em relação à extração sintática, é a possibilidade de se identificar conceitos e suas relações em fontes de dados, sendo que alterações nos processos de extração só ocorrem quando o esquema conceitual é alterado. Isto faz com que os resultados tornem-se mais precisos pois o usuário requisita uma informação baseada em conceitos de um domínio e não na sintaxe de documentos, que é mais suscetível a mudanças.

#### 4.2.1 Extrator de *Embley*

A técnica de extração de dados utilizada por *Embley* associa uma ontologia<sup>5</sup> a documentos semi-estruturados presentes na *Web*, visando a recuperação e estruturação de dados destes documentos [Emb98]. A figura 4.4 exemplifica esta associação, considerando um documento pertencente ao domínio do SBBD, relativo ao programa do evento (a), e parte de uma ontologia que representa os seus conceitos (b).



**Figura 4.4** - Um documento do evento SBBD (a) e uma ontologia associada (b)

Esta técnica focaliza em documentos ricos em dados e com abrangência semântica limitada. Um documento é considerado rico em dados se ele tem um número de constantes identificáveis, tais como nomes de autores, datas e horários, considerando o domínio do SBBD. Um documento tem abrangência semântica limitada se os conceitos do seu domínio podem ser descritos em uma ontologia relativamente pequena.

Especificamente, uma ontologia descreve um esquema conceitual e *data frames*. O esquema conceitual é formado por conceitos léxicos (atômicos) e não-léxicos (não-atômicos), relacionamentos e cardinalidades associadas. *Data frames* descrevem, para as propriedades de conceitos e relacionamentos, restrições de integridade e comandos de extração de dados que devem ser executados sobre documentos do domínio. A figura 4.5 mostra uma especificação de *data frame* para o atributo Nomea do conceito Autor descrito na ontologia da figura 4.4 (b). Conforme esta especificação, todo valor de Nomea é composto de um primeiro nome (First)

<sup>5</sup> O conceito de ontologia é descrito em detalhes no capítulo 5. Do ponto de vista de BD, uma ontologia é similar a um esquema conceitual.

e um sobrenome (*Last*), separados por um ou mais espaços em branco ("*\s+*"). Um Nomea também pode ser um padrão que inicia com uma letra maiúscula seguido de letras minúsculas, podendo ocorrer uma inicial de nome intermediário (indicado pelo padrão opcional "*[A-Z]\.\s+)?*") antes do sobrenome. Na seção *lexicon* (descrição de constantes léxicas) estão definidas extensas listas de nomes e sobrenomes, respectivamente nos arquivos *first.dict* e *last.dict*.

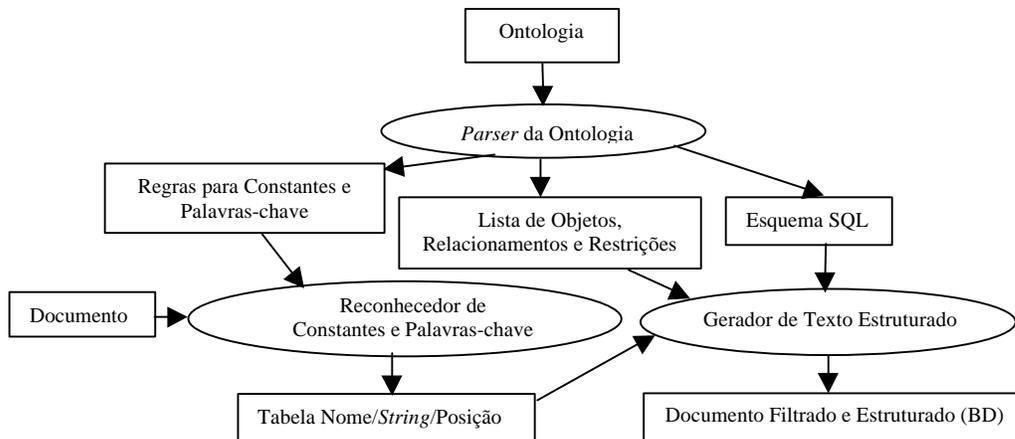
```

Nomea matching[80] case sensitive
constants
{extract First, "\s+", Last;},
...
{extract "[A-Z][a-zA-Z]*\s+([A-Z]\.\s+)?", Last;},
...
lexicon{
    First case insensitive;
    Filename "first.dict";
}, {
    Last case insensitive;
    Filename "last.dict";
};

```

**Figura 4.5** - Exemplo de uma especificação de *data frame*

O processo de extração de dados é apresentado na figura 4.6. O *parser* (analisador sintático) da ontologia gera três arquivos: regras, lista de objetos e esquema SQL. O primeiro arquivo descreve as regras para identificação de elementos da ontologia (conceitos, relacionamentos e atributos) presentes no documento, como valores constantes válidos, expressões regulares que descrevem a forma dos atributos e palavras-chaves associadas. O esquema SQL é uma seqüência de comandos de criação de tabelas relacionais resultante do mapeamento do esquema conceitual. A lista de objetos é uma associação entre os objetos da ontologia e as declarações das tabelas do esquema SQL, mais as restrições de cardinalidade.



**Figura 4.6** - Procedimento de extração proposto por *Embley*

Para cada documento de entrada, inicialmente um processo de pré-filtragem é realizado para remoção de *tags*. Na seqüência, dois procedimentos são executados, utilizando os três arquivos gerados: (i) reconhecimento de constantes e palavras-chaves e (ii) geração de texto estruturado. O reconhecedor produz uma tabela nome/string/posição com base na análise do documento e do arquivo de regras. O gerador utiliza esta tabela para povoar o BD relacional, utilizando a lista de objetos e o esquema SQL. O reconhecedor tem por incumbência identificar

*strings* no texto do documento conforme as expressões regulares, gravando o seu nome (atributo ou palavra-chave), valor (*string*) e posição no texto. O gerador se vale de heurísticas, baseadas na análise de restrições da lista de objetos e dos elementos da tabela, para gerar tuplas do esquema SQL e inseri-las no BD.

Conforme apresentado na figura 4.6, a entrada para o processo de extração é uma página *Web* e a saída é um BD povoado. A ontologia é uma entrada independente, sendo apenas ela alterada quando os requisitos do domínio são modificados. Os demais módulos (*parser*, reconhecedor e gerador) podem ser reexecutados para estes novos requisitos do domínio sem necessitar de alterações.

#### 4.2.2 Extrator de *Dorneles*

O extrator proposto por **Dorneles** atua no contexto de documentos XML [Dor00].

Esta proposta tem como objetivo extrair instâncias XML que estejam de acordo com DTDs derivadas a partir de uma ontologia que representa um domínio sobre o qual pertencem documentos XML. Estas DTDs geradas servem, na verdade, para determinar quais instâncias XML farão parte do resultado de uma consulta.

O processo de extração recebe como entrada um subesquema conceitual (subontologia) referente aos conceitos da ontologia que estão presentes em uma consulta de usuário. Um gerador de DTDs é responsável pela derivação das DTDs a partir desta subontologia. Um gerenciador de documentos fornece classes de documentos XML em que estes conceitos devem ser procurados. Instâncias XML destes documentos são analisadas e, caso estejam de acordo com alguma DTD, são selecionadas para compor a resposta. Este conjunto de instâncias XML, correspondente ao domínio representado pela ontologia, é posteriormente materializado em um BD relacional.

### 4.3 Comparação das Técnicas de Extração

Ambas as técnicas de extração são mais precisas, em termos de resultados de consultas, do que as ferramentas do tipo *search engine* disponíveis atualmente para o acesso a dados *Web*. No caso da extração sintática, quanto mais homogêneos forem os documentos em termos de marcação ou padrão de discurso, mais precisos serão os resultados. No caso da extração semântica, a precisão está diretamente relacionada à fidelidade do documento aos conceitos do domínio representados no esquema conceitual definido.

Uma das vantagens da técnica de extração semântica sobre a sintática é o fato de haver uma maior relação entre o que se deseja extrair e a resposta a esta extração, ou seja, a resposta condiz semanticamente com a consulta formulada. No entanto, o desenvolvimento de uma abordagem semântica de extração é mais complexa pois requer um projeto conceitual de dados bem definido para que todas as informações desejadas em consultas sejam extraídas.

Outro ponto a ressaltar é o conhecimento base para o processo de extração, que no caso da técnica semântica é o domínio do problema, enquanto que na técnica sintática é a organização de cada instância de dado semi-estruturado.

Ainda, a manutenção do processo de extração, no caso da técnica semântica, está intimamente relacionado com alterações realizadas sobre o esquema conceitual do domínio. Já no caso da técnica sintática, alterações nas instâncias de dados semi-estruturados podem invalidar o processo de extração.

A tabela 4.1 mostra um comparativo entre as duas técnicas.

**Tabela 4.1** - Comparação entre as técnicas de extração semântica e sintática

	<b>Extração Semântica</b>	<b>Extração Sintática</b>
<b>Desenvolvimento</b>	+ complexo	- complexo
<b>Resultados</b>	precisos	precisos
<b>Base da Extração</b>	domínio do problema	instâncias de dados
<b>Alteração no Processo</b>	quando altera o esquema	quando altera a instância

## 5 Ontologias e Dados Semi-Estruturados

### 5.1 Ontologia

#### 5.1.1 Definição

O termo “*Ontologia*” já é conhecido e aplicado a bastante tempo na área da Filosofia, significando um “*sujeito de existência*” ou uma “contabilização sistemática da Existência”. Na década de 90, este conceito passou a ser utilizado na Ciência da Computação. Neste contexto, uma ontologia pode ser entendida como “*uma especificação parcial de uma conceitualização consensual*” [Gru93, Gua97, Béz9?, Fik9?]. Por *conceitualização* entende-se um vocabulário de termos, um conjunto de fatos e regras, ou um conjunto de entidades e relacionamentos considerados em um universo de discurso que se deseja representar. Por *consensual* entende-se que esta especificação é senso comum para um conjunto de especialistas do universo de discurso.

Ontologias foram aplicadas primeiramente na área de Inteligência Artificial (IA) como uma **teoria lógica** que restringe os modelos de uma linguagem lógica [Gru93, Gua97]. Neste sentido, dado um conjunto de predicados e funções de uma linguagem lógica, uma ontologia provê axiomas que restringem o sentido dos predicados ou permitem a realização de inferências, como por exemplo,  $\neg\text{casado}(X,X)$ , indicando que uma pessoa não pode estar casada consigo mesma. Essa noção de teoria lógica é aplicada em diversas subáreas da IA, como sistemas de aquisição e representação de conhecimento e sistemas de processamento de linguagem natural.

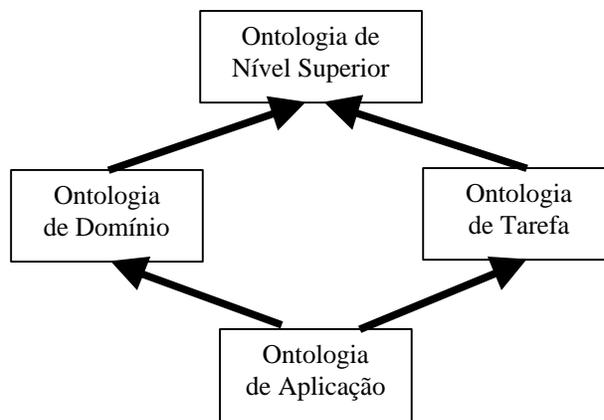
Recentemente, ontologias vêm sendo utilizadas nas áreas de BD e RI como suporte à interoperabilidade de fontes de dados distribuídas e heterogêneas. Do ponto de vista de BD, pode ser definida como “*uma especificação parcial de um domínio ou meta-domínio, descrevendo entidades, relações entre elas e regras de integridade.*” Apesar da definição sugerir uma similaridade com um esquema conceitual de dados, uma ontologia não é precisamente um esquema conceitual: um esquema conceitual descreve a estrutura pretendida dos dados de um BD em um alto nível de abstração. Já uma ontologia propõe um esquema de consenso para um grupo de especialistas que não tem a intenção de refletir estritamente as estruturas de representação das fontes de dados associadas a ela, exigindo que mecanismos de tradução e integração de dados sejam definidos, quando possíveis [Mel00a].

#### 5.1.2 Características

Algumas características desejadas de uma ontologia são [Hwa99]:

- Aberta e dinâmica: deve ser capaz de suportar ajustes decorrentes de mudanças na estrutura ou comportamento do domínio;
- Escalável e interoperável: deve ser facilmente escalável, considerando um domínio amplo, e adaptável a novos requisitos. Deve também ser possível integrar várias ontologias em uma nova ontologia quando o tratamento de diferentes vocabulários conceituais é requerido;
- Fácil manutenção: sua manutenção não deve ser complexa. Portanto, deve ser de fácil compreensão;
- Semanticamente consistente com o domínio;
- Coerente com o contexto: não deve ter termos muito específicos para não tornar complexa a associação com as fontes de dados e futuras integrações com outras ontologias.

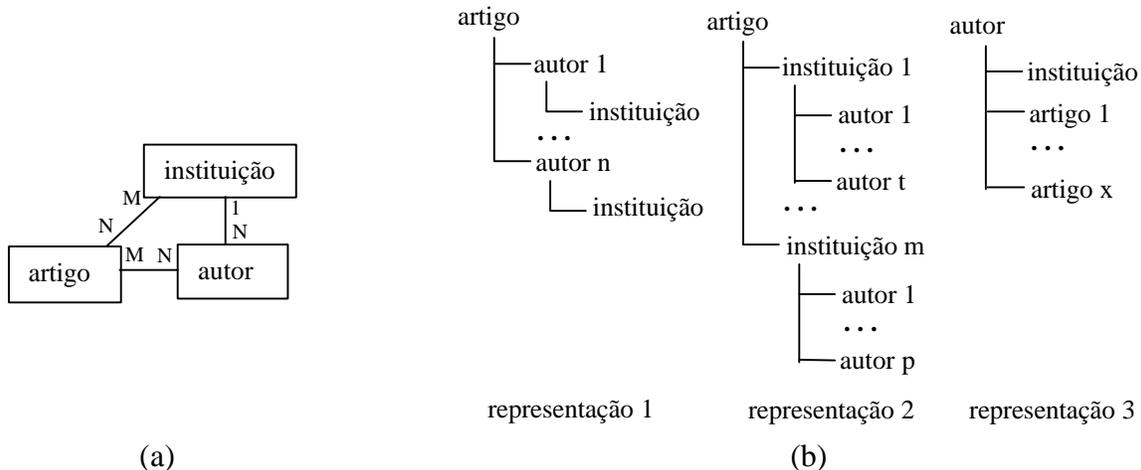
Ontologias são usualmente classificadas em quatro tipos, de acordo com o relacionamento de herança apresentado na figura 5.1 [Gua9?]. Uma **ontologia de nível superior** descreve conceitos genéricos, como espaço, tempo, objeto, ação, etc, sendo utilizada na definição de meta-domínios. Uma **ontologia de domínio** e uma **ontologia de tarefa** descrevem uma conceitualização para, respectivamente, um domínio genérico (conferências, por exemplo) ou uma tarefa genérica (leitura de artigos, por exemplo), especializando conceitos da ontologia de nível superior. Uma **ontologia de aplicação**, específica para uma certa atividade dentro de um domínio, define regras a serem seguidas por conceitos do domínio quando uma certa tarefa é realizada (avaliação de um artigo para uma conferência, por exemplo).



**Figura 5.1** - Tipos de ontologias e seus relacionamentos

## 5.2 Aplicação de Ontologias a Dados Semi-Estruturados

Ontologias vêm sendo aplicadas no gerenciamento de dados semi-estruturados como um suporte semântico para o acesso a determinadas informações de interesse presentes em um conjunto de fontes semi-estruturadas. Uma importante vantagem neste contexto é que a ontologia provê uma **interpretação semântica unificada** para diferentes representações de dados semi-estruturados referentes a um mesmo domínio. Considerando, por exemplo, um pequeno esquema ontológico para o domínio de artigos científicos, composto pelos conceitos artigo, autor e instituição (figura 5.2 (a)), diversas representações semi-estruturadas (DTDs, por exemplo) podem ser derivadas (figura 5.2 (b)). Assim, diversos objetos semi-estruturados neste domínio, com diferentes estruturas, podem estar associados a esta mesma ontologia.



**Figura 5.2** - Uma ontologia (a) e algumas representações semi-estruturadas derivadas dela (b)

Uma vez definido um nível conceitual ontológico, outra vantagem a salientar é que consultas que levam em conta a semântica do domínio podem ser formuladas. Esta vantagem é significativa pois grande parte das linguagens de consulta para dados semi-estruturados baseiam-se na especificação de padrões a serem percorridos em uma estrutura de grafo, considerando apenas a organização hierárquica dos objetos e não a sua semântica. A ontologia abstrai esta organização, permitindo que a intenção de consulta do usuário esteja concentrada nos conceitos do domínio e seus relacionamentos e não nas estruturas lógicas de representação de dados semi-estruturados.

Além das facilidades oferecidas a nível de modelagem e consulta, ontologias podem também guiar processos de extração de dados através da manutenção de informações que auxiliem na identificação de atributos de conceitos e relações em representações semi-estruturadas. A seguir são descritas sucintamente algumas propostas de aplicação de ontologias a dados semi-estruturados que provêm estas facilidades.

### 5.2.1 Observer

**Observer** é um sistema de informação global que utiliza ontologias para o processamento de consultas a fontes de dados heterogêneos estruturados e semi-estruturados [Nie98]. Ontologias de domínio independentes descrevem conceitos e regras organizados em uma hierarquia de especialização que captura a semântica do conteúdo de uma ou mais fontes.

Relacionamentos entre ontologias são definidos através de um gerenciador de relacionamentos (GR) que estabelece, para cada conceito de uma ontologia, informações de mapeamento para conceitos da mesma ou de outras ontologias. Os tipos de relacionamento suportados são: sinônimos, mais geral que, menos geral que, interseção, disjunção e cobertura. Além disso, relacionamentos entre conceitos de uma ontologia e as fontes de dados associadas também são definidos através de uma álgebra relacional estendida. Dado um conceito ontológico artigo, por exemplo, um possível mapeamento é apresentado na figura 5.3.

```

CONCEPT artigo:
<[Union [Selection pub-BDI.publicação[= pub-BDI.publicação.formato "Paper"]]
  [Selection bib-BDI.referência [= bib-BDI.referência.tipo "Artigo"]]],
  pub-BDI.publicação.código, string>

```

**Figura 5.3** - Mapeamento de um conceito ontológico para fontes de dados associadas

Neste exemplo da figura 5.3 tem-se uma expressão de álgebra relacional estendida como primeiro argumento, o(s) atributo(s) que identifica(m) seus objetos e o(s) tipo(s) deste(s) atributo(s). O conceito *artigo* está presente em duas fontes de dados: *pub-BDI* e *bib-BDI*. A álgebra atua como uma linguagem de consulta intermediária entre a especificação textual da consulta do usuário e as linguagens de consulta ou métodos de acesso utilizados por *wrappers*. Expressões de álgebra são otimizadas e divididas em expressões individuais para cada fonte de dados antes de serem traduzidas para expressões de consulta executáveis por cada *wrapper*.

O processador de consultas é o núcleo da arquitetura do sistema, realizando os seguintes passos na execução de uma consulta: construção, acesso e expansão. Os dois últimos passos são cíclicos, terminando quando o usuário estiver satisfeito com o resultado. No primeiro passo, o usuário seleciona uma ontologia alvo e edita a consulta. Para o acesso a dados, o processador de consultas invoca o servidor da ontologia, que recupera e correlaciona os dados, com o auxílio de informações de mapeamento e dos *wrappers*, passando o resultado de volta ao processador. Caso o usuário queira enriquecer a consulta, ocorre o que se chama de expansão: uma nova ontologia alvo é selecionada e a consulta é reescrita na linguagem (vocabulário de conceitos) desta nova ontologia alvo, com o auxílio do GR, preservando-se o máximo possível a semântica da mesma. Caso esta reescrita não seja completa (considerando o número de conceitos da consulta que foram mapeados com sucesso), o percentual de perda de informação não pode ser superior a um limite máximo de perda estipulado pelo usuário. Não ocorrendo este problema, o acesso a essa nova ontologia é feito e o ciclo se repete.

### 5.2.2 SHOE

**SHOE** (*Simple HTML Ontology Extensions*) é uma linguagem que estende a linguagem HTML com *tags* orientadas a conhecimento [Hef99]. Conceitos e regras de inferência disponíveis em ontologias são referidos em SHOE e habilitam a descoberta de conhecimento implícito em documentos da *Web*. Esta linguagem faz parte de um ambiente cuja arquitetura é composta por ontologias, uma ferramenta de marcação que produz novos documentos, um *browser* chamado *Exposé* e uma base de conhecimento chamada *Parka*. Conhecimento SHOE é descoberto em documentos através de *Exposé* e adicionado à *Parka*.

Ontologias são criadas também através da sintaxe SHOE, que especifica hierarquias de conceitos, relacionamentos (*links* entre documentos) e regras de inferência. Atributos de conceitos e relacionamentos são definidos através de regras que associam valores a instâncias de conceitos. Na figura 5.4, uma especificação de ontologia em SHOE (a) e uma instância gerada a partir de uma marcação SHOE baseada nesta ontologia (b) são mostradas.

A ferramenta de marcação cria um documento HTML (página SHOE) e adiciona a semântica da ontologia ao mesmo. Uma ou mais instâncias de conceitos podem ser descritos em um mesmo documento, assim como conceitos de diversas ontologias. A descrição de uma instância consiste da ontologia que esta faz referência, o conceito que a classifica e suas propriedades. No caso de páginas HTML criadas externamente que possuam dados relevantes, definem-se páginas de sumário com *tags* SHOE e *links* para estas páginas.

### 5.2.3 Ontobroker

**Ontobroker** é uma ferramenta baseada em ontologias que processa documentos criados através de linguagens de marcação como HTML e XML, provendo recuperação inteligente de informação [Erd99, Fen99]. A ferramenta possui uma arquitetura formada pelos seguintes componentes: (i) uma máquina de consulta, que recebe consultas e as responde,

verificando o conteúdo de uma base de conhecimento; (ii) um agente de informação, responsável pela coleta de conhecimento factual da *Web* e armazenamento na base de conhecimento; e (iii) uma máquina de inferência, que usa fatos, a terminologia e os axiomas das ontologias para derivar conhecimento factual adicional que também é armazenado na base de conhecimento. As ontologias são o princípio geral de estruturação de dados: o agente de informação as utiliza para extrair fatos, a máquina de inferência para inferir fatos, o gerenciador da base de conhecimento para estruturar os dados e a máquina de consulta para formular consultas.

<pre> &lt;HTML&gt; ... &lt;BODY&gt; &lt;ONTOLOGY ID="Documentos" VERSION="1.0"&gt; &lt;USE-ONTOLOGY ID="Base Ontology" VERSION="1.0" PREFIX="base"&gt; ... &lt;DEF-CATEGORY NAME="Documento" ISA="base.SHOEntity"&gt; &lt;DEF-CATEGORY NAME="Artigo" ISA="Documento"&gt; &lt;DEF-CATEGORY NAME="Conferência" ISA="base.SHOEntity"&gt; ... &lt;RELATION NAME="Apresentação"&gt;   &lt;ARG POS="1" TYPE="Artigo"&gt;     &lt;ARG POS="2" TYPE="Conferência"&gt; &lt;/RELATION&gt; ... &lt;/ONTOLOGY&gt; &lt;/BODY&gt; ... &lt;/HTML&gt; </pre>	<pre> &lt;HTML&gt; ... &lt;BODY&gt; &lt;INSTANCE KEY="http://www.exemplo/shoe/exemplo.html"&gt; &lt;USE-ONTOLOGY ID="Documentos" VERSION="1.0" PREFIX="Documentos" URL="http://www.exemplo/shoe/documentos.html "&gt; ... &lt;CATEGORY NAME="Documentos.Artigo"&gt; &lt;RELATION NAME="Documentos.Nome"&gt;   &lt;ARG POS="TO" VALUE="Ontologias e BDs"&gt; &lt;/RELATION&gt; &lt;RELATION NAME="Documentos.Apresentação"&gt;   &lt;ARG POS="TO" VALUE="http://www.exemplo/shoe/ exemploSBBD2000.html"&gt; &lt;/RELATION&gt; &lt;/INSTANCE&gt; ... &lt;/BODY&gt; ... &lt;/HTML&gt; </pre>
--	---

(a)

(b)

**Figura 5.4** - Especificação de uma ontologia em SHOE (a) e um documento HTML definido com base nela (b).

No caso de uma marcação XML, o agente de informação traduz conceitos, relacionamentos e atributos de ontologias para elementos uma DTD. As ontologias funcionam como um padrão para a sintaxe de documentos XML, ao mesmo tempo em que permitem consultas com caráter semântico. Recebe-se especificações ontológicas e gera-se arquivos de DTDs. As DTDs geradas definem classes de conceitos que auxiliarão na marcação futura de documentos XML. Uma especificação parcial de uma ontologia (em um formalismo orientado a objetos) (a) e sua correspondente DTD (b) são mostradas na figura 5.5.

A especificação da DTD não impõe nenhuma restrição sobre qual dos elementos definidos deve ser o elemento raiz no documento XML, assim como a obrigatoriedade de existência de todos os subelementos.

### 5.2.4 Proposta de *Embley*

A proposta de *Embley* é uma técnica de extração de dados semi-estruturados baseada em ontologias de domínio [Emb98]. Esta técnica é particularmente interessante para conjuntos de documentos que seguem um padrão de discurso uniforme.

A ontologia é um esquema conceitual que, juntamente com um documento semi-estruturado, serve de entrada para um procedimento de extração que produz um documento estruturado que é inserido em um BD relacional.

O processo de extração proposto por *Embley* encontra-se descrito na seção 4.2.1.

```
Object[ ].
Pessoa:: Object.
    Empregado:: Pessoa.
        EquipeAcadêmica:: Empregado.
    Estudante:: Pessoa.
        EstudanteDoutorado:: Estudante.
Publicação:: Object.
    Livro:: Publicação.
    Artigo:: Publicação.
...
Pessoa[nome =>> string; email =>> string; telefone =>> string; publicação =>> Publicação,
    editor =>> Livro].
Publicação[autor =>> Pessoa; título =>> string; ano =>> number; abstract =>> string].
Livro[editor =>> Pessoa].
FORALL Pes1, Publ    Publ: Livro[editor ->> Pes1] <-> Pes1: Pessoa[editor ->> Publ]
```

(a)

```
<!ENTITY % Pessoa "Pessoa | Empregado | EquipeAcadêmica | Estudante | EstudanteDoutorado" >
<!ENTITY % Publicação "Publicação | Livro | Artigo" >
<!ELEMENT Pessoa (#PCDATA | nome | email | telefone | publicação | editor)*>
<!ELEMENT Publicação (#PCDATA | autor | título | ano | abstract)*>
<!ELEMENT Livro (#PCDATA | autor | título | ano | abstract | editor)*>
<!ELEMENT autor (#PCDATA | %Pessoa;)*>
<!ELEMENT título (#PCDATA)*>
<!ELEMENT ano (#PCDATA)*>
<!ELEMENT editor (#PCDATA | %Book; | %Pessoa;)*>
```

(b)

**Figura 5.5** - Especificação de uma ontologia em Ontobroker (a) e uma DTD derivada a partir dela (b).

### 5.2.5 MOMIS

**MOMIS** (*Mediator envirOnment for Multiple Information Sources*) é um ambiente para integração de fontes de dados estruturados e semi-estruturados [Ber99]. A arquitetura do ambiente apresenta os seguintes componentes: (i) um esquema conceitual ou ontológico orientado a objetos especificado em uma linguagem chamada ODL<sub>I</sub><sup>3</sup> (linguagem de integração semântica); (ii) *wrappers* para tradução de esquemas em representações ODL<sub>I</sub><sup>3</sup>; e (iii) componentes mediador e processador de consultas, baseados em duas ferramentas: ARTEMIS e ODB-Tools.

A integração de dados exige que dados semi-estruturados sejam organizados em classes ODL<sub>I</sub><sup>3</sup>. Cada classe é o resultado do mapeamento de um objeto não-atômico de um grafo OEM. Um exemplo de especificação de classe para um objeto Artigo é mostrado na figura 5.7. O símbolo “\*” nesta especificação indica que o atributo pode ser opcional em algumas instâncias.

```
Interface Artigo (source semistructured SBBDD)
{
    attribute string título;
    attribute set <Autor> Autor;
    attribute integer nroPáginas*;
    attribute set <Seções> Seções*;
};
```

**Figura 5.7** - Especificação de uma de classe de objetos semi-estruturados

Para a definição da ontologia, deve-se descrever três tipos de relacionamentos terminológicos para classes e atributos de objetos: sinônimo-de (t1 SYN t2), termo-mais-geral-que (t1 BT t2) e termo-relacionado-a (t1 RT t2). O relacionamento simétrico à termo-mais-geral-que é termo-mais-específico-que (t1 NT t2). A determinação destes relacionamentos é uma tarefa semi-automática feita com o auxílio da ferramenta ODB-Tools.

Na seqüência, o módulo mediador converte a ontologia em um esquema global. Para tanto, são associados coeficientes de afinidade a relacionamentos entre classes ontológicas de fontes de dados distintas, com base nos seus nomes e seus atributos. Um procedimento semi-automático de clusterização hierárquica classifica as classes de acordo com estes coeficientes, gerando uma árvore de afinidade que tem classes como folhas e vértices intermediários com um valor de afinidade associado às classes do seu *cluster* (subárvore). A determinação dos coeficientes e da árvore de afinidade são tarefas da ferramenta ARTEMIS. Após, um procedimento semi-automático de criação de esquemas globais é executado em dois passos, nesta ordem: (i) para cada *cluster*  $C_i$  gera-se uma classe global  $C_i$ . No caso de atributos com relacionamento SYN, apenas um deles é selecionado como atributo global. No caso de atributos com relacionamento BT/NT, o atributo mais geral em  $C_i$  é selecionado como atributo global; (ii) definem-se regras de mapeamento dos atributos globais para os atributos locais das fontes de dados. Regras de integridade (*rules*) podem ser especificadas, expressando relacionamentos semânticos entre fontes de dados distintas.

Um exemplo parcial de especificação da classe Artigo na ontologia global é mostrado na figura 5.8, considerando SBBDD uma fonte semi-estruturada que mantém dados deste evento e BDbib um BD relacional de documentos científicos.

```

Interface Global_Artigo {
  attribute título
    mapping_rule (BDbib.Artigos.título_abrangência and
                  BDbib.Artigos.título_profundidade),
    SBBDD.Artigo.título;
  attribute número_páginas
    mapping_rule (BDbib.Artigos.página_final -
                  BDbib.Artigos.página_inicial + 1),
    mapping_rule SBBDD.Artigo.nroPáginas;
  attribute conferência
    mapping_rule SBBDD.Artigo = 'Simpósio Brasileiro de Banco de Dados',
                  BDbib.Artigos.evento;
  ... }
rule R1 forall X in Global_Artigo:
  (X.conferência = 'Simpósio Brasileiro de Banco de Dados')
  then X.subárea = 'Banco de Dados';

```

**Figura 5.8** - Especificação de um conceito na ontologia global de MOMIS

### 5.2.6 WebKB

**WebKB** é uma ferramenta para indexação, anotação e exploração de conhecimento em documentos da *Web* [Mar99]. Uma linguagem de representação de conhecimento define associações e especializações entre termos predefinidos em uma ampla ontologia de nível superior que serve de base para a criação de ontologias de aplicação. A ferramenta possui operadores para diversos formatos de documentos, como HTML e inglês formalizado.

A linguagem de representação de conhecimento apresenta *tags* especiais que embutem comandos de especificação, indexação ou consulta em documentos. Os termos da ontologia são utilizados nestes comandos para definir ocorrências de relações ou conceitos, como na linguagem SHOE.

Indexação pode ser aplicada a qualquer trecho de um documento textual ou HTML, seja ele uma sentença, seção, referência, imagem ou o documento completo. O código descrito na figura 5.9 define um índice para a segunda ocorrência da sentença “*ontologias em BDs*”, presente em um documento HTML, referente aos conceitos Artigo e Assunto de uma ontologia.

```
$ (Indexation
  (Context: Language: GC; Ontology: http://www.bar.com/topLevelOntology.html;
    Repr_author: phmartin; Creation_date: Mon sep 14 02:32:21 PDT 1998;
    Indexed_doc: http://www.bar.com/exemplo.html;)
  (DE: (2nd occurrence) ontologias em BDs)
  (Repr: [Assunto: ontologias] <- (Assunto) <- [Artigo]) )$
```

**Figura 5.9** - Indexação de conceitos ontológicos em um documento HTML

Consultas são formuladas através da seleção de termos da ontologia e especificação de predicados de seleção. Resultados podem ser estruturas de dados ou documentos completos. Graças ao embutimento de comandos de consulta em documentos, pode-se associar uma consulta a um *link*. A consulta é processada quando este *link* é ativado, gerando um documento HTML (documento virtual) com a estrutura do resultado da consulta. Documentos virtuais adaptam o conteúdo dos documentos da *Web* à semântica da ontologia.

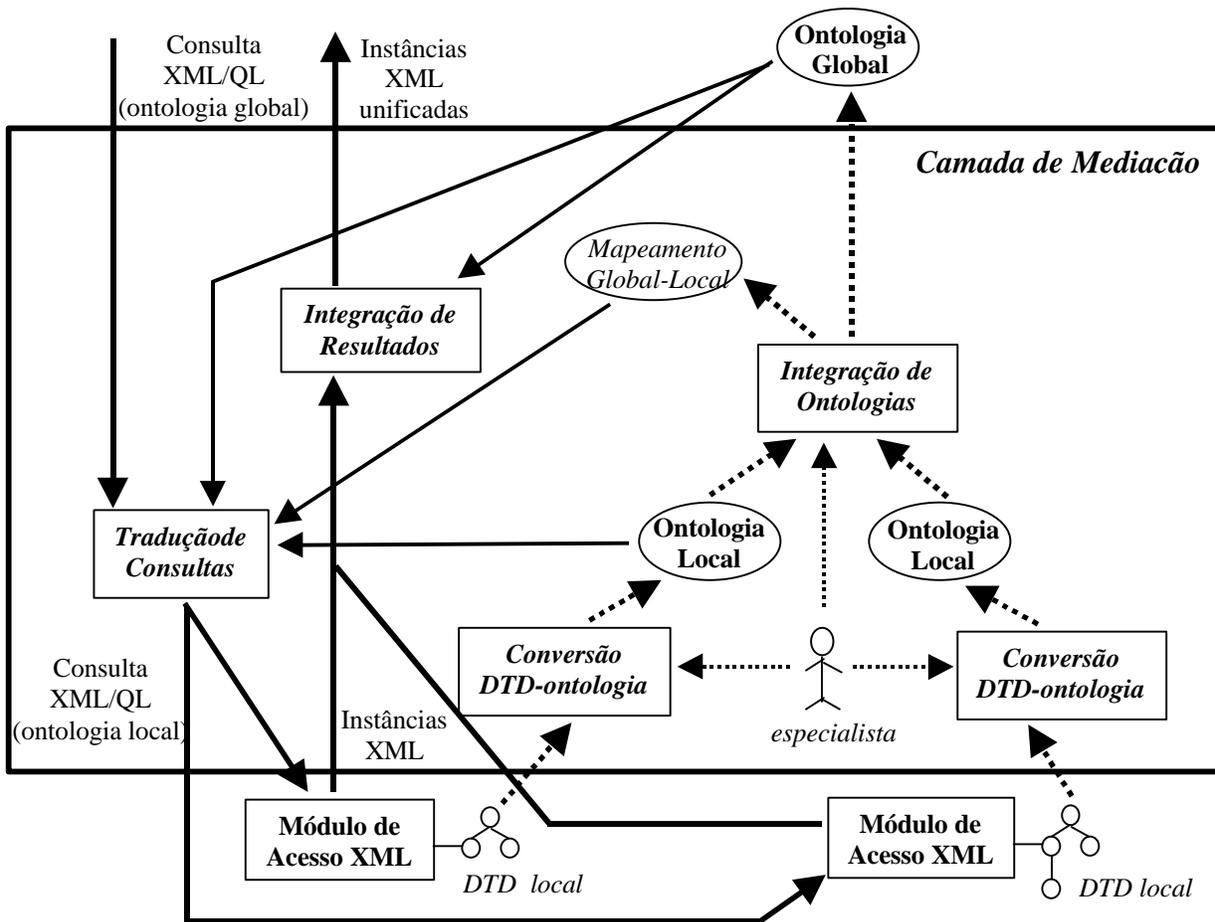
### 5.2.7 Camada de Mediação XML

A **Camada de Mediação XML** é um serviço que realiza a integração de esquemas XML e a consulta a instâncias XML de diversas fontes integradas, sendo um componente da arquitetura de um sistema de acesso a fontes XML [Mei00b]. Tal camada é composta por quatro módulos e dois níveis de abstração conceitual de dados formados por ontologias de domínio, como mostra a figura 5.10. As ontologias são esquemas conceituais gerados a partir de DTDs, servindo de base para a formulação de consultas que são enviadas a módulos de acesso a fontes XML. Cada módulo de acesso, que pode ser um *wrapper* ou uma base de documentos, é responsável por executar consultas XML/QL sobre documentos que estão definidos de acordo com uma DTD (DTD local) e retornar um conjunto de instâncias XML como resultado.

O processo de integração de esquemas XML é *bottom-up* e semi-automático, partindo de várias DTDs locais e terminando em uma ontologia global que é um esquema global que integra elementos de diversas DTDs locais. Um especialista humano é necessário para validar a semântica das ontologias geradas. Este processo é mostrado na figura 5.10 através de setas tracejadas, sendo composto por dois módulos: Conversão DTD-Ontologia e Integração de Ontologias. O primeiro módulo gera uma ontologia local a partir de uma DTD local. Para tanto, realiza a análise da descrição da DTD e das instâncias XML associadas para determinar os conceitos e seus relacionamentos, cardinalidades e atributos da ontologia. Por conceitos entende-se os elementos da DTD local. O segundo módulo realiza a integração semântica de ontologias locais, baseado na análise de equivalências e resolução de conflitos entre conceitos destas ontologias, gerando uma ontologia global e informações de mapeamento entre conceitos. A ontologia global é disponibilizada para camadas superiores da arquitetura do sistema para servir de base conceitual na formulação de consultas XML/QL.

O processamento de consultas na camada de mediação apresenta igualmente dois módulos: Tradução de Consultas e Integração de Resultados, sendo representado na figura 5.10 através de setas contínuas. O primeiro módulo é responsável por transformar as estruturas hierárquicas de condição e resultado (DTDs derivadas da ontologia global) presentes em uma

consulta XML/QL em estruturas compatíveis com o vocabulário e a organização hierárquica dos conceitos de cada DTD local. Assim, para cada DTD local são executadas operações de transformação das DTDs da consulta, gerando consultas XML/QL interpretáveis por cada módulo de acesso XML. Estas operações de transformação necessitam das descrições das ontologias global e local e das informações de mapeamento entre elas. O segundo módulo recebe conjuntos de instâncias XML provenientes dos módulos de acesso e realiza uma integração de ocorrências de elementos idênticos, produzindo instâncias XML unificadas que são enviadas às camadas superiores da arquitetura do sistema. Este módulo se vale de informações de identificação de elementos não-atômicos presentes na ontologia global para realizar tarefas como eliminação de dados redundantes e complementação de dados de elementos idênticos.



**Figura 5.10** - Camada de mediação para o acesso a fontes XML baseada em ontologias

### 5.3 Comparação das Propostas

A tabela 5.1 compara alguns aspectos do uso de ontologias pelas propostas.

As ontologias diferem principalmente em termos de **tipos** suportados. Ontologias de domínio estão presentes em todas as propostas, pois refletem diretamente os requisitos de um universo de discurso. *WebKB* apresenta uma ontologia de nível superior predefinida como base para novas ontologias e permite a definição de todos os tipos de ontologias. Algumas propostas definem regras associadas a conceitos e relações. SHOE e Ontobroker, por exemplo, descrevem regras para inferência de conhecimento. Em MOMIS, pode-se especificar restrições de

integridade associadas a cada conceito ontológico. A Proposta de Embley associa regras de extração (padrões para reconhecimento de dados em documentos) a conceitos, relacionamentos e atributos.

**Tabela 5.1** – Comparação entre as propostas de aplicação de ontologias a dados semi-estruturados

	Observer	SHOE	Ontobroker	Proposta de Embley	MOMIS	WebKB	Camada de Mediação XML
<b>Tipos de ontologia</b>	nível superior; domínio	domínio	domínio	domínio	domínio	nível superior; domínio; tarefa; aplicação	domínio
<b>Elementos da ontologia</b>	conceitos; relações; regras de mapeamento para fontes de dados	conceitos; relações; regras de inferência	conceitos; relações; regras de inferência	conceitos; relações; regras de extração	conceitos; relações; regras de mapeamento para fontes de dados; restrições de integridade	conceitos; relações	conceitos; relações
<b>Suporte dado pela ontologia</b>	processamento de consultas; esquema conceitual	anotação de conhecimento; inferência de conhecimento; esquema conceitual	anotação de conhecimento; inferência de conhecimento; esquema conceitual	extração de conhecimento; esquema conceitual	processamento de consultas; esquema conceitual	anotação de conhecimento; indexação de conhecimento; esquema conceitual	processamento de consultas; esquema conceitual
<b>Base de definição da ontologia</b>	requisitos do domínio	requisitos do domínio; ontologias existentes	requisitos do domínio; ontologias existentes	requisitos do domínio; ontologias existentes	fontes de dados	requisitos do domínio; ontologia única	fontes de dados
<b>Relação ontologia – fonte de dados</b>	um-para-muitos	muitos-para-muitos	um-para-muitos	muitos-para-muitos	um-para-muitos	um-para-muitos	um-para-muitos

Quanto à **relação ontologia-fonte de dados**, a alternativa mais flexível é aquela na qual o conhecimento de uma fonte de dados pode estar associado a diversas interpretações semânticas, ou seja, diversas ontologias. Várias propostas, porém, restringem esta relação para o caso um-para-muitos, considerando apenas documentos específicos de um certo domínio, como MOMIS e Ontobroker. Observer mapeia o conhecimento de uma fonte de dados para apenas uma ontologia, porém, este conhecimento pode ser traduzido para outras ontologias relacionadas.

A **definição** de ontologias segue, em geral, requisitos do domínio alvo e o reuso de ontologias pré-existentes. Observer não leva em conta integração de ontologias no processo de definição. MOMIS e a Camada de Mediação XML constróem ontologias a partir de esquemas de fontes de dados. A maioria das propostas lida com várias pequenas ontologias e com a possibilidade de reuso de conhecimento ao invés de uma única e ampla ontologia.

Quanto à **aplicação**, é consenso a utilização de ontologias como **esquemas conceituais** a partir do qual consultas com caráter semântico podem ser formuladas. Observer, MOMIS e a Camada de Mediação XML empregam ontologias no **processamento de consultas** para, respectivamente, expandir uma consulta a várias conceitualizações semanticamente relacionadas, restringir fontes de dados a serem pesquisadas através da aplicação de restrições de integridade e mapear consultas XML/QL baseadas em uma ontologia global para DTDs de fontes locais XML. A Proposta de Embley tem por finalidade a **extração de dados** semi-estruturados de documentos cujo domínio está representado em uma ontologia que mantém as regras para reconhecimento de seus conceitos. SHOE, Ontobroker e WebKB utilizam ontologias para a **anotação de conhecimento** em documentos e posterior armazenamento em um BD.

SHOE e Ontobroker realizam inferência de conhecimento armazenado através da investigação de relacionamentos de herança. WebKB permite ainda a especificação de **índices** de elementos ontológicos sobre trechos de documentos. A principal fraqueza destas abordagens é que apenas documentos novos ou com permissão de escrita são passíveis de anotação, o que não se aplica a maioria dos documentos já existentes. SHOE tenta contornar este problema através da criação de páginas de sumário com anotações semânticas para páginas já existentes.

## 6 Considerações Finais

Dados semi-estruturados é um tema atual de pesquisa na área de BDs, motivado principalmente pela grande disponibilidade de dados na *Web* e a conseqüente necessidade de acesso a estes dados de forma simples pelos usuários, com resultados de consultas precisos e concisos. As ferramentas disponíveis atualmente para pesquisa na *Web* não atendem a estas necessidades. Por isto, o objetivo final das pesquisas apresentadas é manipular dados semi-estruturados de maneira eficiente, da mesma forma que se manipulam dados em BDs convencionais.

Este tutorial apresenta um panorama dos tópicos em maior evidência dentro deste tema, abordados nos capítulos anteriores: modelagem de dados, linguagens de consulta, extração de dados e aplicação de ontologias.

**Modelos de dados** são orientados a grafos, para melhor representar a heterogeneidade de dados semi-estruturados. Nestes modelos, vértices representam objetos e arestas direcionadas representam subobjetos ou valores atômicos. Esta estrutura é hierarquizada, devendo existir pontos de entrada que indiquem conjuntos de objetos complexos semi-estruturados. Em alguns casos, estes modelos podem ser abstraídos para páginas (vértices) e *links* (arestas), como o modelo ADM [Atz97a, Atz97b], que representa *sites* na *Web*. Nestes contextos, objetos modelados podem apresentar uma maior estruturação, de forma a manter atributos específicos do domínio, como URL, data da última alteração, lista de itens da página, etc.

Quanto a **linguagens de consulta**, várias propostas existem, permitindo buscas em estruturas de grafo. Para tanto, pesquisam-se sintaxes para declaração de expressões regulares de caminho, necessários para a recuperação da estrutura de objetos, assim como algoritmos de otimização ou álgebras para grafos, de forma a tornar mais eficiente o processamento de consultas. Mecanismos para reestruturação do resultado de consultas, a fim de gerar novos objetos ou simplesmente facilitar a navegação também estão sendo levados em consideração.

Um ponto a salientar é a ausência de alguns requisitos importantes nas diversas linguagens existentes, como a busca baseada em estrutura e o uso de coerção, fundamentais no tratamento de dados semi-estruturados. As próprias linguagens de consulta para XML ainda estão em um processo inicial de desenvolvimento. Não há, até o presente momento, uma linguagem que atenda todos os requisitos apontados na seção 3.1. Em função disto, o W3C criou o XML *Query Working Group*, com o objetivo de fornecer recursos de consulta flexíveis para extrair dados de documentos *Web* reais e virtuais [Mar00]. Espera-se definir, como resultado deste processo, uma linguagem capaz de consultar eficientemente a *Web* tanto do ponto de vista da comunidade de documentos quanto da comunidade de BD. Um pioneiro esforço de pesquisa nesta direção é a linguagem Quilt [Rob00], ainda em desenvolvimento.

A pesquisa em **extração de dados** tem revelado várias técnicas semi-automáticas, baseadas principalmente em padrões sintáticos ou esquemas conceituais a partir dos quais são aplicadas heurísticas para recuperação de informações. Estas técnicas consideram padrões presentes na descrição de documentos ou em *tags* de linguagens de marcação que organizam

o texto de documentos. Estas abordagens limitam a reusabilidade destas técnicas para fontes de dados de domínios quaisquer, porém, os resultados de consultas são mais precisos e concisos, se comparados com ferramentas de RI.

A arquitetura de consenso para extração parece ser a que utiliza *wrappers* para recuperar dados e retorná-los de forma adequada ao modelo de dados do sistema e mediadores para combinar estes resultados em uma resposta integrada. Quando não existe um módulo mediador explícito, o processador de consultas normalmente realiza a tarefa de integração de dados. Porém, uma camada de mediação é interessante de ser implementada quando o modelo de dados tem pouca semântica, procurando resolver problemas como nomenclatura de atributos e atributos inexistentes ou com tipos distintos.

No que tange à **ontologias**, o uso pioneiro em IA, principalmente em representação e inferência de conhecimento, motivou a aplicação das mesmas na área de dados semi-estruturados. A definição de um domínio de representação de dados facilita a legibilidade e a precisão de acesso a dados semi-estruturados. A finalidade básica de uma ontologia é servir como um esquema conceitual que dá suporte semântico a consultas. A conceitualização definida em uma ontologia ou é anotada diretamente nos dados das fontes ou é extraída das fontes através de *wrappers* e mediadores. Algumas propostas também a empregam no processamento de consultas e extração de dados através de regras de mapeamento de conceitualizações, regras de integridade, regras que especificam padrões de busca e heurísticas voltadas a domínios específicos. Ainda, em alguns casos, ontologias descrevem metadados em um alto nível de abstração para facilitar o desenvolvimento de ontologias de domínio, de tarefas e de aplicações. Porém, não é consenso a vinculação de novas ontologias a uma única e ampla ontologia universal, devido a problemas como o uso imposto de um vocabulário global e manutenção de consistência. A maioria das propostas prefere lidar com várias pequenas ontologias e com a possibilidade de reuso de conhecimento.

Ontologias vem se popularizando como meio de busca mais eficiente de dados na *Web* através de sistemas gerenciadores de dados semi-estruturados. Esta constatação sugere que soluções efetivas em termos de manipulação e metodologias de construção de ontologias sejam encontradas.

Diversos pontos ainda continuam em aberto na área de dados semi-estruturados, dada a carência de literatura a respeito, como por exemplo, atualização de dados e propagação de modificações, armazenamento e indexação, gerência de transações, restrições de integridade, autorização, interoperabilidade com SGBDs tradicionais, etc. Não se sabe ainda se as técnicas tradicionais de BDs para estas questões devem ser modificadas ou não. Particularmente, os problemas de atualização e interoperabilidade são complexos pois dados semi-estruturados evoluem rapidamente e são heterogêneos.

## Bibliografia

- [Abi97a] ABITEBOUL, S. Querying Semistructured Data. In: **INTERNATIONAL CONFERENCE ON DATABASE THEORY**, 1997, Delphi, Greece. p.1-18.
- [Abi97b] ABITEBOUL, S. et al. The Lorel Query Language for Semistructured Data. **International Journal on Digital Libraries**, v.1, n.1, p.68-88, Apr. 1997.
- [Abi00] ABITEBOUL, S.; BUNEMAN, P.; SUCIU, D. **Data on the Web: from Relations to Semistructured Data and XML**. San Francisco: Morgan Kaufmann, 2000.
- [Ade99] ADELBERG, B.; DENNY, M. **Building Robust Wrappers for Text Sources**. Chicago: Department of Computer Science, Northwestern University, 1999. Technical Report.

- [Aro98] AROCENA, G.; MENDELZON, A. WebOQL: Restructuring documents, databases and webs. In: **INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE)**, Orlando, Florida, 1998.
- [Ash97] ASHISH, N.; KNOBLOCK, C. Wrapper Generation for Semi-structured Internet Sources. **SIGMOD Record**, v.26, n.4, p.8-15, Dec. 1997.
- [Atz97a] ATZENI, P.; MECCA, G.; MERIALDO, P. To Weave the Web. In: **INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES (VLDB'97)**, 23., 1997, Athens. **Proceedings...** [S.l.]: Morgan Kaufmann, 1997, p.206-215.
- [Atz97b] ATZENI, P.; MECCA, G.; MERIALDO, P. Semistructured and Structured Data in the Web: Going Back and Forth. **SIGMOD Record**, v.26, n.4, p.16-23, Dec. 1997.
- [Atz97c] ATZENI, P.; MECCA, G.; MERIALDO, P. Cut and Paste. In: **SIGMOD INTERNATIONAL SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS (PODS'97)**, 16., Tucson, Arizona, USA, 1997. p.144-153.
- [Bar98] BARU, C. et al. Features and Requirements for an XML View Definition Language: Lessons from XML Information Mediation. In: **W3C WORKSHOP ON QUERY LANGUAGES (QL'98)**, 1998, Boston. Disponível por WWW em <http://www.w3.org/TandS/QL/QL98/pp/xmas.html> (julho de 2000)
- [Ber99] BERGAMASCHI, S.; CAETANO, S.; VINCINI, M. Semantic Integration of Semistructured and Structured Data Sources. **SIGMOD Record**, v.28, n.1, p.54-59, Mar. 1999.
- [Béz9?] BÉZIVIN, J. **Who's Afraid of Ontologies?** Disponível por WWW em <http://www.metamodel.com/oopsla98-cdn-workshop/bezivin>. (agosto de 1999)
- [Bun96] BUNEMAN, P. et al. A Query Language and Optimization Techniques for Unstructured Data. In: **ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA (SIGMOD'96)**, Montreal, Canadá. **Proceedings...** New York: ACM Press, 1996, p.505-516.
- [Bun97] BUNEMAN, P. Semistructured Data. In: **SIGMOD INTERNATIONAL SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS (PODS'97)**, 16., Tucson, Arizona, USA, 1997. p.117-121.
- [Bra00] BRADLEY, N. **The XML Companion**. Harlow: Addison Wesley Longman Limited, 2ª Ed., 2000.??p.
- [Cat98] CATARCI, T. et al. Accessing the Web: Exploiting the Data Base Paradigm. In: **INTERDISCIPLINARY WORKSHOP ON BUILDING, MANTAINING AND USING ORGANIZATIONAL MEMORIES (OM'98)**, 1., 1998, Brighton, UK.
- [Cha94] CHAWATE, S. et al. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In: **Tenth Anniversary Meeting of the Information Processing Society of Japan**, 1994.
- [Cre98] CRESCENZI, V.; MECCA, G. **Grammars Have Exceptions**. 1998. Disponível por WWW em <http://www.difa.unibas.it/users/gmecca> (março de 2000)
- [Deu99] DEUTSCH, A. et al. A Query Language for XML. In: **WORLD WIDE WEB CONFERENCE (WWW8)**, 1999, Toronto, Canada. Disponível por WWW em <http://www.research.att.com/~mff/xmlql/doc/files/final.html> (agosto 2000)
- [Dor00] DORNELES, C. F. **Extração de Dados de Semi-Estruturados Baseados em uma Ontologia**. Porto Alegre: PGCC da UFRGS, 2000. (Dissertação de Mestrado)
- [Emb98] EMBLEY, D.W. et al. A Conceptual-Modelling Approach to Extracting Data from the Web. In: **INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING (ER'98)**, 17., 1998, Singapore. **Proceedings...** Berlin: Springer-Verlag, 1998. (Lecture Notes in Computer Science, v.1507)

- [Erd99] ERDMANN, M.; STUDER, R. Ontologies as Conceptual Models for XML Documents. In: **KNOWLEDGE ACQUISITION FOR KNOWLEDGE-BASED SYSTEMS WORKSHOP**, 12., 1999, Banff, Canada.
- [Fen99] FENSEL, D. et al. On2broker: Semantic-Based Access to Information Sources at the WWW. In: **WORLD CONFERENCE ON THE WWW AND INTERNET (WEBNET 99)**, 1999, Honolulu, Hawaii, USA.
- [Fer97] FERNANDEZ, M. et al. A Query Language for a Web-Site Management System. **SIGMOD Record**, v.26, n.3, p.4-11, Sept. 1997.
- [Fer98] FERNANDEZ, M.; SUCIU, D. **Query Optimizations for Semistructured Data Using Graph Schema**, 1998. Disponível por WWW em <http://www.research.att.com/info/~mff> (fevereiro de 2000)
- [Fik9?] FIKES, R.E. **Ontologies: What are They, and Where's the Research?** Disponível por WWW em <http://www-ksl.stanford.edu/KR96/FikesPositionStatement.html> (setembro de 1999)
- [Flo98] FLORESCU, D.; LEVY, A.; MENDELZON, A. Database Techniques for the World Wide Web: A Survey. **SIGMOD Record**, v.27, n.3, p.59-74, Mar. 1997.
- [Gru93] GRUBER, T.R. A Translation Approach to Portable Ontologies. **Knowledge Acquisition**, v.5, n.2, p.199-200, 1993.
- [Gua9?] GUARINO, N. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In: **Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology**. Berlin: Springer Verlag, p.139-170.
- [Gua97] GUARINO, N. Understanding, Building and Using Ontologies: A Commentary to "Using Explicit Ontologies in KBS Development", by van Heijst, Schreiber, and Wielinga. **International Journal of Human and Computer Studies**, v.46, n.2/3, p. 293-310, 1997.
- [Ham97] HAMMER, J. et al. Extracting Semistructured Information from the Web. In: **WORKSHOP ON MANAGEMENT OF SEMISTRUCTURED DATA**, 1997, Tucson, Arizona, USA. **Proceedings...** New York: ACM Press, 1997, p.18-25.
- [Ham98] HAMMER, J.; MCHUGH, J.; GARCIA-MOLINA, H. **Semistructured Data: The TSIMMIS Experience**. Disponível por WWW em <http://www-db.stanford.edu/> (abril 2000)
- [Hef99] HEFLIN, J.; HENDLER, J.; LUKE, S. Applying Ontologies to the Web: A Case Study. In: **INTERNATIONAL WORK-CONFERENCE ON ARTIFICIAL AND NATURAL NEURAL NETWORKS (IWANN'99)**, 1999. **Proceedings...** Berlin: Springer-Verlag, v.II, 1999, p.715-724.
- [Hug97] MCHUGH, J. et al. Lore: A Database Management System for Semistructured Data. **SIGMOD Record**, v.26, n.3, p.54-66, Sep. 1997.
- [Hwa99] HWANG, C.H. **Incompletely and Imprecisely Speaking: Using Dynamic Ontologies for Representing and Retrieving Information**. Austin: Microelectronics and Computer Technology Corporation, Texas, 1999. (Technical Report MCC-INSL-043-99)
- [Kon95] KONOPNICKI, D.; SHMUELI, O. W3QS: A Query System for the World Wide Web. In: **INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES (VLDB)**, 1995, Zurich, Switzerland. p.54-65.
- [Lae99] LAENDER, A.; SILVA, E.; SILVA, A. DEByE - Uma Ferramenta para Extração de Dados Semi-Estruturados. In: **SIMPÓSIO BRASILEIRO DE BANCO DE DADOS (SBBD'99)**, 1999, Florianópolis, SC. p.155-170.

- [Lak96] LAKSHMANN, L.V.S.; SADRI, F.; SUBRAMANIAN, I.N. A Declarative Language for Querying and Restructuring the Web. In: **IEEE WORKSHOP ON RESEARCH ISSUES IN DATA ENGINEERING (RIDE-NDS'96)**, 1996, New Orleans, Louisiana, USA, 1996.
- [Li99] LI, WEN-SYAN et al. WebDB Hypermedia Database System. **IEICE Transactions On Information Systems**, v.E00 A, n.1, Jan 1999.
- [Mar99] MARTIN, P.; EKLUND, P. Embedding Knowledge in Web Documents. In: **WORLD WIDE WEB CONFERENCE (WWW8)**, 1999, Toronto, Canada.
- [Mar00] MARTIN, D. et al. **Professional XML**. Birmingham: Wrox Press, 2000.
- [Mel00a] MELLO, R. S. **Aplicação de Ontologias a Bancos de Dados Semi-Estruturados**. Porto Alegre: PPGC/UFRGS, Fev. 2000. 150p. (Exame de Qualificação nº 45)
- [Mel00b] MELLO, R. S. **Uma Camada de Mediação para a Integração de Fontes XML com o Suporte de Ontologias**. Porto Alegre: PPGC/UFRGS, 2000. (Tese de Doutorado em andamento)
- [Men97] MENDELZON, A. O.; MIHAILA, G. A.; MILO, T. Querying the World Wide Web. **International Journal on Digital Libraries**, v.1, n.1, p. 54-67, Apr. 1997.
- [Nes97] NESTOROV, S.; ABITEBOUL, S.; MOTWANI, R. Inferring Structure in Semistructured Data. **SIGMOD Record**, v.26, n.4, p.39-43, Dec. 1997.
- [Nie98] NIETO, E.M. **OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies**. Zaragoza: Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, 1998. 200p. (Tesis Doctoral)
- [Pap95] PAPAKONSTANTINOU, Y.; GARCIA-MOLINA, H.; WIDOM, J. Object Exchange Across Heterogeneous Information Sources. In: **INTERNATIONAL CONFERENCE ON DATA ENGINEERING**, Taipei, Taiwan. p.251-260.
- [Rob98] ROBIE, J. LAPP, J. SCHACH, D. XML Query Language (XQL). In: **W3C WORKSHOP ON QUERY LANGUAGES (QL'98)**, 1998, Boston, USA. Disponível por WWW em <http://www.w3.org/TandS/QL/QL98/pp/xql.html>.
- [Rob00] ROBIE, J.; CHAMBERLAIN, D.; FLORESCU, D. Quilt: an XML Query Language. **XML Europe**, Paris, Jun. 2000. Disponível por WWW em [http://www.almaden.ibm.com/cs/people/chamberlin/quilt\\_euro.html](http://www.almaden.ibm.com/cs/people/chamberlin/quilt_euro.html).
- [Sch98] SCHACH, D.; LAPP, J.; ROBIE, J. XML Querying and Transforming XML. In: **W3C WORKSHOP ON QUERY LANGUAGES (QL'98)**, 1998, Boston. Disponível por WWW em <http://www.w3.org/TandS/QL/QL98/pp/query-transform.html>.
- [Suc97] SUCIU, D. Management of Semistructured Data. **SIGMOD Record**, v.26, n.4, p.4-7, Dec. 1997.
- [W3C 99] **W3C Namespaces in XML**. Disponível por WWW em <http://www.w3.org/TR/1999/REC-xml-names-19990114/> (agosto, 2000)
- [W3C 00a] **W3C XML Schema Part 0: Primer**. Disponível por WWW em <http://www.w3.org/TR/2000/wd-xmlschema-0-20000407/> (agosto, 2000)
- [W3C 00b] **W3C Working Draft - XML Query Requirements**. Disponível por WWW em <http://www.w3.org/TR/2000/WD-xmlquery-req-20000815>.
- [Wie92] WIEDERHOLD, G. Mediators in the Architecture of Future Information Systems. **Computer**, v. 25, n.3, March, 1992, p.38-49.
- [Wra9?] **WRAPPERS**. Disponível por WWW em <http://www.objs.com/survey/wrap.htm>. (dezembro de 1999)

[XML9?] **W3C Extensible Markup Language (XML) Activity**. Disponível por WWW em <http://www.w3.org/XML> (agosto, 2000).

[Xwg98] The Query Language Position Paper of the XSL Working Group. In: **W3C WORKSHOP ON QUERY LANGUAGES (QL'98)**, 1998, Boston. Disponível por WWW em <http://www.w3.org/TandS/QL/QL98/pp/xsl-wg-position.html> (agosto, 2000)