UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Extração Semântica de Dados Semi-Estruturados Através de Exemplos e Ferramentas Visuais

por

IRACI CRISTINA DA SILVEIRA

Dissertação submetida à avaliação como requisito
parcial para a obtenção do grau de Mestre em
Ciência da Computação

Prof. Dr. Carlos Alberto Heuser
Orientador

Porto Alegre, setembro de 2001.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Silveira, Iraci Cristina da

Extração Semântica de Dados Semi-Estruturados Através de Exemplos e Ferramentas Visuais / por Iraci Cristina da Silveira. Porto Alegre: PPGC da UFRGS, 2001.

99 p.il.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2001. Orientador: Heuser, Carlos Alberto.

1. Dados Semi-Estruturados. 2. Extração de Dados Semi-Estruturados. 3.Extração Semântica. 4. XML. 5. Ontologia. 6. Gramática. I. HEUSER, Carlos Alberto II. Título

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profª. Wrana Panizzi
Pró-Reitor de Ensino: Prof. José Carlos Ferraz Hennemann
Pró-Reitor Adjunto de Pós-Graduação: Prof. Philippe Olivier Alexandre Navaux
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PPGC : Prof. Carlos Alberto Heuser
Bibliotecária Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Agradecimentos

Em primeiro lugar, agradeço à pessoa mais envolvida neste trabalho, o meu orientador, Prof. Dr. Carlos Alberto Heuser. Mais uma vez demonstrou o excelente profissional que é, sabendo compreender e corrigir os meus desvios de caminho. Pela forma como orienta os alunos, torna esta experiência mais agradável e tranqüila. Sua participação foi imprescindível à conclusão desta dissertação. Obrigado por ter acreditado em mim.

Muitas são as pessoas que precisariam ser citadas neste agradecimento. Todos familiares, amigos e colegas que convivem comigo e que sempre demonstraram interesse em acompanhar o andamento deste trabalho. A simples demonstração de interesse sempre trouxe motivação para que eu me empenhasse ainda mais, apesar dos obstáculos encontrados. Meus maiores agradecimentos devo às pessoas que sentiram minha falta e compreenderam esta ausência em muitas ocasiões.

Ao meu marido, Mauro De Carli, o melhor companheiro que alguém poderia ter, cumpriu a promessa de "sempre juntos em todas as horas", dando apoio e incentivo em todas as atividades em que me envolvo. Com certeza, nossa escolha e nossa convivência me fizeram crescer e progredir como pessoa.

Agradeço aos meus pais, Newton Borges da Silveira e Luci Pocai da Silveira, que me acolheram nesta jornada e que foram responsáveis por me ensinar os princípios e os valores que hoje norteiam minha personalidade e minha vida. Aos meus irmãos, Rodrigo e Luana, que proporcionaram a convivência em família e que estão sempre juntos em todos os momentos.

Aos meus colegas de mestrado, com os quais compartilhamos nosso tempo no desenvolvimento dos exaustivos trabalhos do curso. Foram muitas noites e muitos finais de semana juntos, mas valeu a pena.

Ao professor Ricardo Dorneles, que demonstrou o maior interesse em ajudar a resgatar os conceitos de compiladores e gramáticas, disponibilizando material para pesquisa. E ao professor Marcos Casa que se empenhou em tornar este Curso uma realidade.

Ao grupo de orientandos do Prof. Dr. Carlos Alberto Heuser sempre prontos para troca de referências bibliográficas e idéias.

Enfim, a todos que de uma forma ou de outra cruzaram por meu caminho e deixaram seus rastros nesta estrada.

Sumário

Lista de Abreviaturas

AVP	*Attribute-Value Pair*
BNF	*Backus-Naur Form*
DEByE	*Data Extraction By Example*
DMBS	*Data Base Management System*
DTD	*Document Type Definition*
EBNF	*Extend Backus-Naur Form*
E-R	Entidade-Relacionamento
GUI	*Graphical User Interface*
HC	*Highest Count Heuristic*
HTML	*Hypertext Markup Language*
IA	Inteligência Artificial
IS	*Identifiable Separator Heuristic*
JDBC	*Java Database Connectivity*
JEDI	*Java based Extraction and Dissemination of Information*
LORE	*Lightweight Object Repository*
LOREL	*Lightweight Object Repository Language*
NIAM	*An Information Analysis Method*
NoDoSe	*Northwestern Document Structure Extractor*
OCL	*Object Constraint Language*
OE	*Object Extraction*
OEM	*Object Exchange Model*
OSM	*Object-oriented Systems Model*
RP	*Repeating Pattern Heuristic*
SD	*Standard Deviation Heuristic*
SGML	*Standard Generalized Markup Language*
SQL	*Structured Query Language*
TSIMMIS	*The Stanford-IBM Manager of Multiple Information Sources*
UML	Unified Modeling Language
URL	*Unified Resource Locator*
W3C	*World Wide Web Consortium*
W4F	*World Wide Web Wrapper Factory*
WWW	*World Wide Web*
XML	*Extensible Markup Language*

Lista de Figuras

Lista de Tabelas

Resumo

Existe uma necessidade latente de pesquisar, filtrar e manipular informações disponíveis em diversos formatos irregulares, entre elas as informações distribuídas na WWW (*World Wide Web*). Esses tipos de dados são semi-estruturados, pois não possuem uma estrutura explícita e regular, o que dificulta sua manipulação.

Este trabalho apresenta como proposta o projeto de uma ferramenta para realizar a extração semântica e semi-automática de dados semi-estruturados. O usuário especifica, através de uma interface visual, um exemplo da estrutura hierárquica do documento e de seu relacionamento com os conceitos da ontologia, gerando uma gramática descritiva da estrutura implícita do mesmo. A partir dessa gramática, a ferramenta realiza a extração dos próximos documentos de forma automática, reestruturando o resultado em um formato regular de dados, neste caso, *XML (eXtensible Markup Language*).

Além da conceituação do método de extração, são apresentados os experimentos realizados com o protótipo da ferramenta, bem como, os resultados obtidos nestes experimentos. Para a construção desta ferramenta, são analisadas características de outros métodos que constituem o estado da arte em extração de dados semi-estruturados.

Palavras-chave: dados semi-estruturados, extração de dados semi-estruturados, extração semântica, XML, ontologia, gramática.

TITLE: “SEMANTIC EXTRACTION OF SEMISTRUCTURED DATA BY EXAMPLES E VISUAL INTERFACE”

Abstract

There is a necessity of researching, filtering and manipulating information available on WWW (World Wide Web). These informations are called semistructured data because it doesn´t have an explicit and regular structure.

This work proposes an environment for to do a semantic and semiautomatic extraction of semistructured data. The user specifies in a visual interface, an example of hierarchic structure of the document and its relationship with the objects of ontology. After that, the environment can extract the informations of further documents automatically. The results are stored in a XML (eXtensible Markup Language), which it´s a data regular format.

It describes the method of extraction, the experiments performed by prototype tool, and the results of these experiments. This work analised too the features of other extraction methods.

Keywords: semistructured data, data extraction, extraction of semi-structured data, semantic extraction, XML, ontology, grammar.

1 Introdução

Existe uma necessidade latente de pesquisar, filtrar e manipular informações disponíveis em diversos formatos irregulares armazenados em meios eletrônicos. Nesse contexto, a *WWW (World Wide Web)* aparece como o principal veículo para difusão de tal tipo de informação.

Segundo Abiteoul [ABI97, ABI98], as informações ou dados eletrônicos podem ser classificados em três tipos: dados não estruturados ou "crus", como arquivos binários; dados estruturados, como bancos de dados; e dados semi-estruturados que formam uma classe intermediária entre esses dois tipos. Dados semi-estruturados são autodescritivos, pois não existe um esquema previamente definido para eles [ABI2000, BAE99, BUN97, BUN97a]. As características básicas dos dados semi-estruturados são:

- *Estrutura Irregular*: As instâncias de um mesmo tipo de dados podem possuir estruturas diferentes.
- *Estrutura Implícita*: A estrutura não está explícita como ocorre com as definições de bancos de dados. A estrutura é implícita e exige um processo de reconhecimento.
- *Estrutura Parcial*: Somente parte da estrutura de dados da instância precisa ser identificada. Não é necessário ou não é possível identificar a estrutura de outra parte da instância.
- *Estrutura Indicativa*: Existem vários tipos de dados para o mesmo objeto relacionado nas várias instâncias.
- *Estrutura Flexível*: A própria estrutura da instância é flexível, uma vez que varia de uma instância para outra. Um objeto, em uma instância, pode ser um tipo de dados atômico (simples que não possui decomposição), e, em outra instância, pode ser um tipo de dados complexo (composto por outros objetos).

As primeiras ferramentas que surgiram para pesquisa de dados semi-estruturados foram os programas de busca de dados na WWW, como o Altavista[1] e o Yahoo[2]. Essas ferramentas utilizam técnicas chamadas de *full text search,* que comparam o texto completo com um determinado conjunto de caracteres informado. Os resultados de buscas feitas com essas ferramentas são imprecisos, pois não consideram a semântica dos textos.

É emergente a necessidade de ferramentas que possam pesquisar dados semi-estruturados e devolver respostas mais objetivas e precisas às consultas do usuário, da mesma forma como ocorre com os bancos de dados tradicionais. Diferente dos dados armazenados em bancos de dados relacionais ou orientados a objetos, os dados semi-estruturados necessitam de uma etapa prévia ao processo de busca da informação, que consiste na recuperação de sua estrutura implícita.

Tal necessidade tem estimulado a realização de várias pesquisas com o intuito de extrair informações de dados semi-estruturados. De uma forma genérica, os trabalhos desenvolvidos recuperam a estrutura implícita do documento, extraem as informações e as convertem em algum formato estruturado de dados. A partir da informação

[1] Disponível em http://www.altavista.com

[2] Disponível em http://www.yahoo.com

estruturada, o usuário pode realizar consultas e obter os resultados desejados. Podem ser citadas as ferramentas DEByE [LAE99], TSIMMIS [HAM98], NoDoSe [ADE98], JEDI [HUC98], W4F [AZA99], Editor [ATZ98] e os estudos de Embley [EMB98a], que são explicados nas seções que seguem. Os métodos aqui estudados estão relacionados a técnicas de bancos de dados. Entretanto, a área de Inteligência Artificial também tem estudado os métodos de extração para dados semi-estruturados [BER99, LOH99].

O objetivo do presente trabalho é apresentar o projeto de uma ferramenta de extração semântica e semi-automática de dados semi-estruturados. Ela abrange os processos de definição da estrutura dos dados, de extração propriamente dita e da organização dos dados extraídos em um formato estruturado[SIL2001]. Com os dados reorganizados, o usuário pode se valer de outras ferramentas para manipular e consultar as informações desejadas.

O processo de extração utilizado na ferramenta pode ser classificado em extração semântica, porque está baseado em uma ontologia (modelo conceitual) que possui conhecimento prévio sobre o domínio do problema, e que incrementa o conhecimento com a evolução dos processos de extração. Cada documento processado pertence a uma determinada classe de documentos, e cada classe está relacionada a um determinado domínio do mundo real através de uma ontologia (modelo conceitual). Essa ontologia oferece à ferramenta informações sobre como os objetos estão relacionados entre si, quais são as relações de aninhamento entre eles, e que tipos de dados podem ser aplicados aos objetos.

A ferramenta é dita semi-automática porque o usuário deve exemplificar uma primeira definição da estrutura do documento. Em uma primeira etapa, através de uma interface visual, o usuário decompõe o documento em partes cada vez menores e associa cada uma dessas porções aos objetos representados na ontologia. Para cada um desses objetos, o usuário identifica o tipo de dado que melhor o representa, respeitando as regras impostas pela ontologia, e informa os delimitadores de texto para encontrar a informação desejada. Essa característica é muito importante porque a montagem manual de regras de combinação é um trabalho exaustivo que consome muito tempo, está sujeito a erros e exige muito conhecimento do usuário.

O resultado do processo é uma gramática representativa do documento que serve como guia para as demais extrações que passam a ser automáticas para aquela classe. As informações extraídas são organizadas em uma nova forma de dados, agora estruturada – *XML (eXtensible Markup Language).*

Este trabalho está estruturado em vários capítulos. O capítulo 2 apresenta uma avaliação sobre os métodos que constituem o estado da arte em extração de dados semi-estruturados, comparando algumas características entre eles. No capítulo 3 está descrita a proposta principal do trabalho – o método de extração de dados semi-estruturados. Esse capítulo explica o conceito de ontologias e sua aplicação no método, uma vez que é o fundamento de todo processo. Nesse mesmo capítulo é descrito o processo de extração completo, fundamentando seus conceitos básicos. O capítulo 4 demonstra os experimentos que foram realizados e os resultados alcançados. O capítulo 5 apresenta um esboço do processo de interação com o usuário através de uma ferramenta visual. O capítulo 6 apresenta as conclusões e os trabalhos futuros.

2 Métodos de Extração de Dados Semi-Estruturados

Esta Seção tem como objetivo apresentar o estado da arte em métodos de extração de dados semi-estruturados.

Conforme mencionado, as ferramentas mais populares, através da técnica *full text search*, tornam o trabalho do usuário exaustivo, não retornam a informação completa e precisa e não satisfazem a necessidade de consulta.

Não existe um método já concebido e aceito como um padrão eficiente de extração. Muitas pesquisas estão sendo desenvolvidas e cada qual trabalha com uma abordagem diferente e específica. Os caminhos são divergentes, mas o objetivo é único: definir a estrutura implícita dos dados da forma mais automática possível, extrair as informações e disponibilizá-las em um formato estruturado, a fim de facilitar a consulta.

Nesta Seção, estão descritas as principais pesquisas desenvolvidas em extração de dados semi-estruturados voltadas ao aspecto de banco de dados. As pesquisas normalmente fazem parte de um projeto mais amplo que englobam linguagens de consultas, formação de documentos, otimização de extrações, materialização de visões e outros componentes. O foco de estudo é somente o processo de extração, analisando-o sobre alguns critérios definidos por este trabalho. Os critérios de análise, citados a seguir, são utilizados no final desta Seção para comparar os métodos estudados.

a) *Tipos de Documentos Manipulados:* Este critério identifica o tipo de documento alvo tratado pela ferramenta.

b) *Recuperação da Estrutura:* Relaciona a forma de recuperação da estrutura implícita dos documentos, a qual pode ocorrer antes ou depois da existência das instâncias.

c) *Dependência de Contexto:* Os extratores são baseados em regras que podem utilizar os termos predecessores e sucessores para chegar à informação desejada. O nível de dependência destes termos identifica a dependência de contexto.

d) *Nível de Automaticidade do Processo:* Este critério é identificado pelo grau de necessidade da interferência do usuário no processo de extração.

e) *Tipos de Dados:* É importante reconhecer se o método trabalha com dados atômicos e/ou complexos.

f) *Regras de Extração:* Identifica a forma de representação das regras de extração utilizadas pelos extratores.

g) *Processo de Extração:* Classifica a estratégia adotada para extrair as informações desejadas a partir das regras de extração.

h) *Resultado da Extração:* Identifica o formato em que as informações extraídas são reorganizadas.

Os métodos analisados são: Editor [ATZ98, ATZ98a, ATZ98b], Nodose [ADE99, ADE98], o projeto TSIMMIS [PAP95, HAM98], DEByE [LAE99, LAE2000, LAE99a], W4F [AZA99], JEDI [HUC98] e o método de Embley [EMB99, EMB99a, EMB99b, EMB98, EMB98a].

2.1 Editor

O projeto *Araneus Web-Base Management System*, desenvolvido pela *Universitá di Roma Tre*, é formado por vários componentes: Penelope (criação de páginas HTML); Minerva (criação do extrator); Editor (linguagem utilizada pelo extrator); Gerenciador de Objetos (gerencia objetos em um banco de dados); Ulixes (criação de consultas sobre os objetos do banco) [ATZ98a].

O componente Editor é uma linguagem de pesquisa e reestruturação de documentos que utiliza as operações comuns de um editor de texto. Ele opera basicamente sobre documentos HTML, uma vez que o projeto está voltado para a WWW. O componente Editor trata os documentos como uma cadeia de símbolos de um determinado alfabeto. Os documentos são divididos pelo usuário em regiões que são reorganizadas em uma nova estrutura de documento. O programa montado é executado: extrai as informações desejadas e as organiza em uma TABELA de bancos de dados [ATZ98].

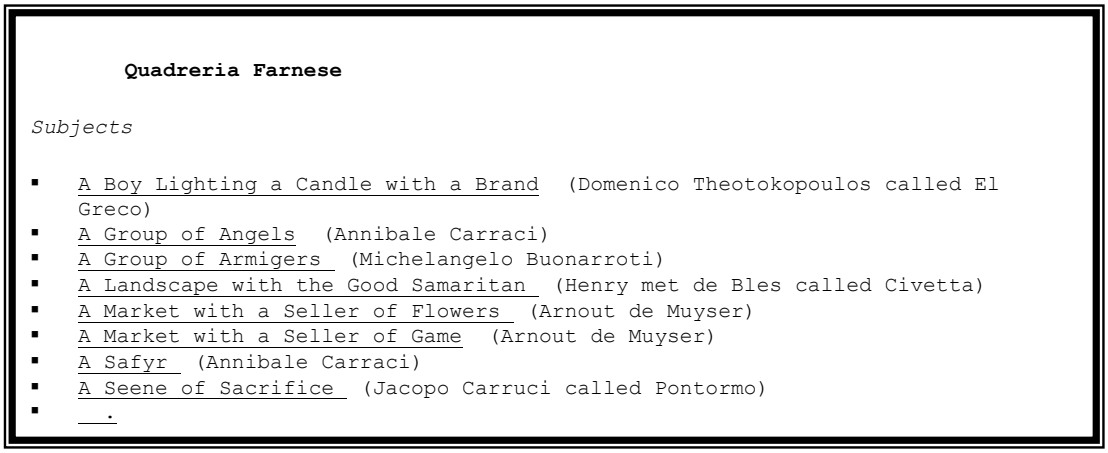
As instruções básicas do Editor são:

- *Pesquisa*: Para demarcar as regiões de interesse, o usuário seleciona esta porção do texto através da instrução "*search"*. A instrução "*search"* escolhe a porção do texto que combina com a cadeia de caracteres informada. Cada caracter da cadeia deve ser comparado, com exceção do símbolo coringa ("*"), que é validado contra qualquer caracter. Ao validar a pesquisa, a porção do texto correspondente passa a ser a região corrente. Para realizar tais operações, o Editor precisa de uma área de armazenamento na memória, chamada de "*clipboard"*.
- *Modificação*: O usuário pode modificar o texto correspondente à região corrente através da instrução *"replace"*.
- *Cortar*: A instrução "*cut"* remove a cadeia de símbolos do documento e a copia para o *clipboard.*
- *Copiar*: A instrução *"copy"* copia a região corrente para o *clipboard* não modificando o documento original.
- *Iterações*: Para casos em que se faz necessária a execução de várias repetições do mesmo conjunto de comandos, está disponível a instrução "*loop"*.

A FIGURA 2.1 exibe uma página escrita em HTML que contém dados sobre pintores famosos. Cada linha da página relaciona o título de uma obra de pintura e seu pintor, que está citado entre parênteses. Cada citação de pintor e obra no fonte HTML possui o seguinte formato: <LI> <A HREF="referencia-pagina"> titulo-da-obra </A> (nome-pintor).

A FIGURA 2.2 exibe um programa em Editor que tem como objetivo extrair os títulos das obras e os nomes dos pintores que as criaram. Para efetivar a extração, o programa precisa realizar o mesmo conjunto de instruções repetidas vezes, o que leva à utilização da instrução "*loop*". Para cada iteração, o programa pesquisa a região do texto

que contém a cadeia de caracteres “<LI*>)”. Como a informação desejada está contida em uma única linha, é transferida para a área de *clipboard*. Nessa etapa, são eliminadas as informações desnecessárias e dividida a informação em duas porções menores - título da obra e pintor. As informações extraídas são reorganizadas em uma TABELA, onde cada tupla é formada por duas colunas. A primeira coluna contém o nome da obra e a segunda o nome do pintor. Ambos atributos são delimitados por um par de colchetes.



Quadreria Farnese

Subjects

- A Boy Lighting a Candle with a Brand (Domenico Theotokopoulos called El Greco)
- A Group of Angels (Annibale Carraci)
- A Group of Armigers (Michelangelo Buonarroti)
- A Landscape with the Good Samaritan (Henry met de Bles called Civetta)
- A Market with a Seller of Flowers (Arnout de Muyser)
- A Market with a Seller of Game (Arnout de Muyser)
- A Safyr (Annibale Carraci)
- A Seene of Sacrifice (Jacopo Carruci called Pontormo)
- .

FIGURA 2.1 – Página HTML sobre Pintores – Exemplo do Editor

```
Replace (TABELA, ∈, “Titulo \t Autor \n”);      Monta o cabeçalho da TABELA.
Replace (TABELA, ∈, “------- \t ------- \n”);   Monta o cabeçalho da TABELA.
Loop search (HTMLPintores, “<LI>*)”)            Procura a lista de itens de forma iterativa.
        Copy (HTMLPintores);                    Copia o item corrente para o clipboard.
        Paste (Temporário);                     Cola este item no arquivo temporário.
        Replace (TABELA, ∈, “ [”);              Escreve um colchetes na TABELA gerada.
        Search (Temporario,”<LI>*>”);           Elimina da TABELA temporária os caracteres
        Cut (temporario);                       que antecedem ao título.
        Search (Temporario,”*<\A>”);            Seleciona o título,
        Cut (temporario);                       Elimina do temporário,
        Paste (TABELA);                         Gera na TABELA nova.
        Replace (TABELA,”</A>”,”\t”);           Substitui a marca HTML por um tabulador.
        Search (temporario,”*)”);               Procura o restante do item que contem o autor.
        Cut (temporario);                       Remove do temporário.
        Paste (TABELA);                         Cola o autor na TABELA nova.
        Replace (TABELA, “ (“, ∈);              Remove o parênteses da esquerda e
```

FIGURA 2.2 – Programa Extração Dados de Pintores – Exemplo do Editor

Em [CRE98], as instruções básicas do Editor são utilizadas juntamente com o mecanismo de gramáticas, definidas na notação EBNF (*Extend Backus-Naur Form)*. Para cada documento, é definido um conjunto de produções. Cada produção é composta por um símbolo terminal ou não-terminal. O objetivo principal do trabalho é oferecer um mecanismo de manuseio de exceções. Para cada produção é possível adicionar uma cláusula de exceção que é executada sempre que o extrator falhar para aquela produção.

O componente Editor do Araneus define a estrutura dos documentos a partir de suas instâncias, o que o torna totalmente dependente do contexto da informação. O

programa Editor precisa ser escrito manualmente pelo usuário. Não trabalha especificamente com tipos de dados, pois todas as informações são tratadas como uma seqüência de caracteres. O processo de extração segue a lógica utilizada pelo usuário que acessa o documento do início ao fim. O resultado da extração é reorganizado em TABELAs de bancos de dados. Uma característica importante do projeto é o de possuir uma arquitetura composta por componentes, o que permite trabalhar em focos isolados do projeto. Uma das principais contribuições desse projeto é o estudo realizado sobre o tratamento de exceções em gramáticas, que pode ser encontrado em [CRE98].

Maiores detalhes sobre o método podem ser encontrados em [ATZ98, ATZ98a, ATZ98b, ATZ98c, CRE98, MEC99, MEC99a].

2.2 NoDoSe

NoDoSE (*the Northwestern Document Structure Extractor*) é uma ferramenta que extrai dados semi-estruturados de forma semi-automática. Atualmente, encontra-se na sua versão 2.0 [ADE99].

O processo de extração NoDoSe pode ser descrito através de três etapas: modelagem dos documentos, decomposição dos documentos e saída dos dados extraídos [ADE98].

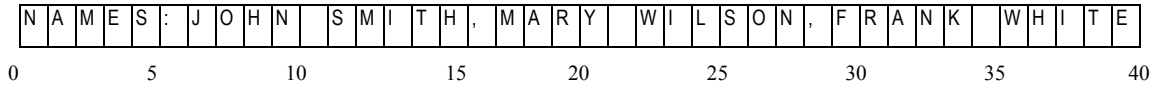
A etapa de modelagem consiste em definir a composição da estrutura do documento. Os componentes são representados por uma lista de pares formado pelo nome do componente e seu valor <nome, valor>.

Na segunda etapa, o usuário decompõe hierarquicamente o documento, utilizando a interface gráfica (*GUI*). Ele inicia indicando o nível mais alto da estrutura do documento – considerado nível de topo. A decomposição continua até encontrar os elementos mais simples, que não possuem mais composição. O resultado dessa etapa é uma representação da estrutura do documento. Os próximos documentos do mesmo tipo sofrem uma análise automática utilizando a estrutura gerada. Os prováveis erros que podem acontecer nos demais documentos são corrigidos pelo usuário através da mesma interface gráfica. Esses erros podem ser resultados de conflitos de nomes, ou elementos novos que não estão presentes no documento modelo. O processo encerra quando todo o conjunto de documentos tiver sido analisado.

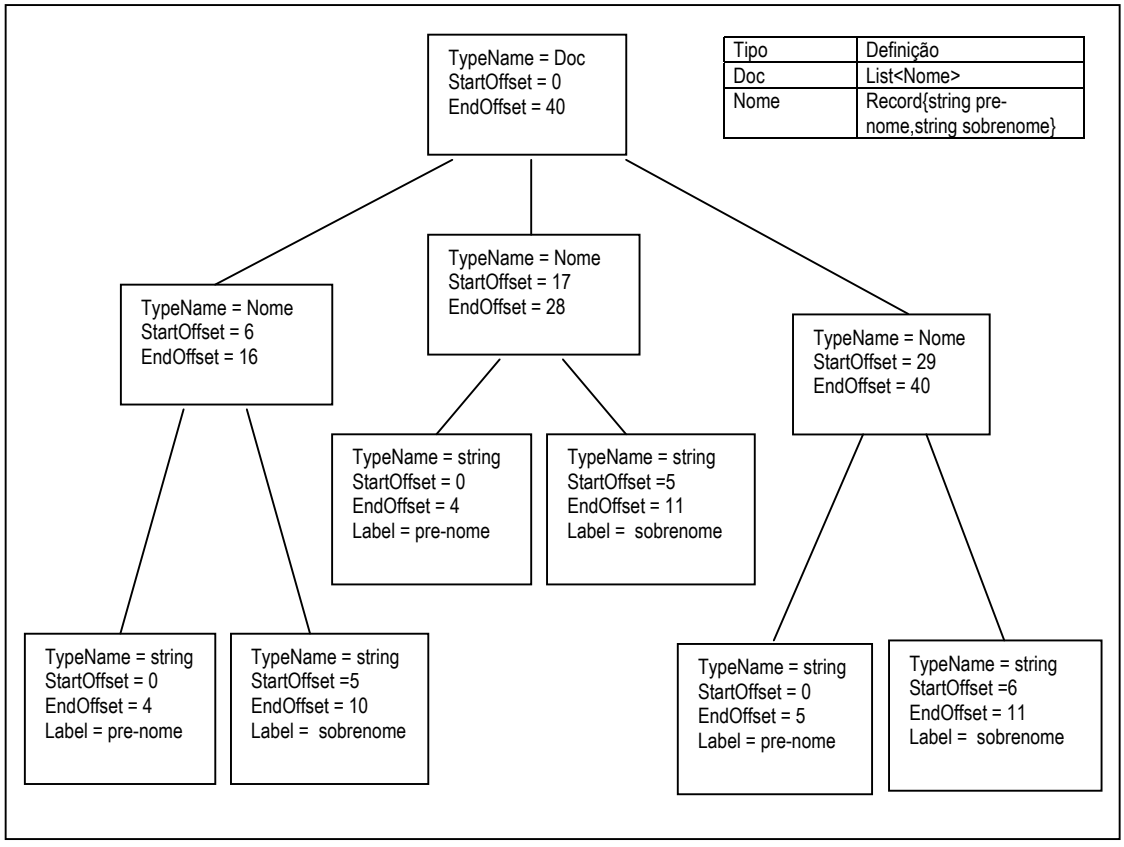
O último passo do processo de extração é especificar a forma em que os dados extraídos são armazenados. O usuário pode optar em transformar as informações em arquivos texto com forma tabular, em uma TABELA de banco de dados ou em linguagem XML.

Após as etapas de modelagem e decomposição dos documentos, é mantida uma estrutura interna que reflete a estrutura hierárquica do documento. Esta estrutura é similar a uma árvore, onde cada nodo corresponde a um componente do documento. Cada nodo carrega consigo uma série de informações referentes ao componente como: "*typename"* (tipo do nodo atômico ou complexo); "*startoffset"* e *"endoffset"* (posição do texto onde se encontra o componente); "*authorid"* (criador do nodo, o usuário ou a ferramenta); "*confidencevalue"* (nível de confiança de correção do nodo).

A FIGURA 2.3 (a) mostra um arquivo que contém uma lista de nomes, sendo que cada um é composto de pré-nome e sobrenome, e estão separados por uma vírgula. A estrutura deste documento está representada na árvore da FIGURA 2.3 (b). O nodo raiz é o "*Doc"* que representa uma lista de elementos do tipo "*Nome"*. A raiz tem três filhos, que corresponde a cada um dos nomes. Cada um desses filhos possui mais dois filhos, correspondendo aos dois campos de "pré-nome" e "sobrenome". Esses últimos nodos são atômicos do tipo "*string"* (cadeia de caracteres) e não possuem mais filhos. Observa-se que o "*startoffset"* (posição inicial) do nodo correspondente ao "pré-nome" possui o valor zero, porque é o primeiro elemento da lista e corresponde ao "*offset" (*posição final) de seu pai.



(a) Exemplo de uma Lista



(b) Árvore da Estrutura do Arquivo
FIGURA 2.3 – Representação de um Documento NoDoSe – Exemplo

NoDoSE pré-define seis tipos atômicos de dados: *integer*[3] (número inteiro), *decimal* (número real), *string* (cadeia de caracteres), *date* (data), endereço de correio

[3] Os termos foram mantidos em inglês por serem aceitos e amplamente usados nesta língua.

eletrônico e URL; e quatro tipos complexos: *set, bag, list* e *record*. Estes tipos podem construir tipos aninhados. Além desses tipos estruturados, o NoDoSE providencia quatro tipos para dados semi-estruturados: *SemiSet, SemiBag, SemiList, SemiRecord.*

Este projeto tem como objetivo proporcionar testes e estudos sobre extração de dados. Por essa razão, foi desenvolvido em uma arquitetura aberta, com componentes que possam ser trabalhados isoladamente e que estejam interligados entre si, através da linguagem Java. Alguns componentes principais da arquitetura NoDoSe: gerenciador de arquivos para habilitar os documentos a serem lidos; gerenciador de instâncias que mantém a estrutura do documento em árvore; gerenciador de documentos que armazena as informações a respeito de uma classe de documentos; gerenciador de *log* e transações para realizar os processos de desfazer e reprocessar ações já executadas (*undo/redo*); minerador de estrutura; interface gráfica (GUI) para que o usuário decomponha o documento; módulo de relatórios responsável por gerar o resultado da extração em formato XML, arquivo texto ou banco de dados; analisador de estatísticas para armazenar números de extração e impor maiores restrições às novas extrações (disponível somente na versão 2.0).

O componente chamado de minerador refere-se ao processo automático de extração dos dados após a demonstração da decomposição do documento. Apesar dos algoritmos para extração de arquivos texto e fontes HTML serem diferentes, os princípios básicos das duas abordagens são iguais. Para as instâncias trabalhadas, o NoDoSe identifica o tipo alvo, ou seja, o tipo do dado em estudo naquele momento.

O objetivo é decompor o documento até chegar na unidade indivisível e, assim, encontrar as teorias que delimitam essa unidade. A teoria de início é chamada de "*start theory"* e a de fim "*end theory"*. Algumas teorias utilizadas pelo minerador: [with marker "marca"] – inicia com a marca; [after marker "marca"] – inicia após a marca; [offset offset] – inicia na posição indicada; [with marker "marca"] – finaliza com a marca; [before marker "marca"]- finaliza antes da marca; [after lines nlinhas] – finaliza após um número fixado de linhas; [offset offset] – finaliza na posição fixada.

Quando o tipo alvo for um dado complexo, o minerador precisa encontrar a teoria que melhor se aplica aquele dado. No caso de um dado do tipo lista torna-se necessário gerar duas teorias: a teoria do cabeçalho e a de seus elementos. Para gerar a teoria do cabeçalho é preciso examinar todos os cabeçalhos conhecidos e encontrar o de sufixo mais longo e comum a todos. Tomando como exemplo a FIGURA 2.4, o sufixo comum para todos os cabeçalhos é a cadeia de caracteres ";Name:", assim, a teoria para cabeçalho passa a ser [With Marker ";Name:"]. Cada lista da FIGURA 2.4 corresponde ao nome de um time de futebol seguido por seus jogadores. Os nomes circulados são os elementos demonstrados pelo usuário. A teoria do cabeçalho é importante, pois ele precisa ser desconsiderado ao se buscarem os elementos. A geração da teoria dos elementos é feita de forma similar ao cabeçalho, com a diferença que os elementos possuem uma "*start theory"* e uma "*end theory"*. Para encontrar a melhor teoria, o algoritmo tenta encontrar um sufixo ou prefixo comum localizado nas lacunas entre um elemento e outro. Em nosso exemplo, o literal "Name" satisfaz a condição. Assim, a teoria candidata resultante da combinação da "*start theory"* e da "*end theory"* é < [After Marker "Name:"], [Before Marker "Name:]>.

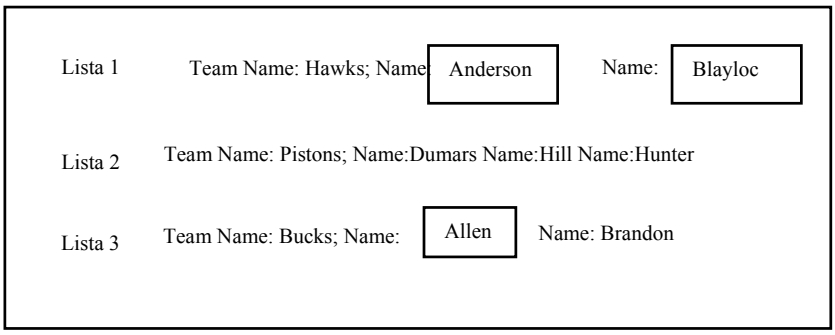


FIGURA 2.4 – Exemplo de um Dado Tipo Lista - NoDoSe

Com as teorias encontradas, o extrator passa para a etapa de avaliação da melhor teoria. A melhor teoria é a que identifica todos os cabeçalhos corretamente. Para os elementos, a procura inicia no primeiro caracter da unidade de texto. O "*start theory"* é utilizado para encontrar o primeiro elemento e, em caso de sucesso, procura seu "*end theory"*. A procura continua até que não existam mais elementos. Cada elemento encontrado é adicionado ao conjunto de predicados. Assim, ao final da busca, os elementos são comparados com os indicados pelo usuário. Para apurar a acuracidade da busca de elementos, são utilizadas algumas funções. A função *findfalsenegatives (predicset, actualset)* conta o número de elementos definidos pelo usuário os quais a teoria não comprovou. A função *findfalsepositivies (predictset, actualset)* conta o número de elementos encontrados que não estão corretos, os quais passam a ser novos elementos reconhecidos.

As teorias encontradas são aplicadas e as informações são extraídas. Os novos elementos extraídos que ainda não pertencem à árvore são adicionados nela pelo gerenciador de instâncias.

É importante destacar que para dados complexos, do tipo registro, cada campo possui seu próprio conjunto de teorias, uma vez que possuem tipos diferentes. Para evitar problemas de classificação, o minerador tenta encontrar uma ordem para os campos em suas instâncias. O minerador usa um algoritmo simples que armazena para cada campo, o conjunto de todos os campos que o precederam em um registro e um conjunto de todos os campos que o sucederam. Com esses dois conjuntos, é possível obter uma certa ordem.

O NoDoSe utiliza como regras de extração as teorias criadas. Estas, quando testadas e aplicadas, indicam os delimitadores da informação. Extrair informações de textos planos é mais difícil do que de páginas HTML, pois, dependendo do caso, não possuem indicação alguma sobre a estrutura do documento. O NoDoSe se diferencia por oferecer um processo semi-automático para a extração dos dados, além de possuir uma arquitetura aberta que facilita o estudo de seus componentes.

Maiores detalhes sobre a ferramenta podem ser encontrados em [ADE98, ADE99, NOD99].

2.3 TSIMMIS

O projeto TSIMMIS (***T**he **S**tanford-**I**BM **M**anager of **M**ultiple **I**nformation **S**ources*) [CHA94, GAR97, HAM95], da Universidade de Stanford, tem desenvolvido uma pesquisa extensa na área de extração e manipulação de vários tipos de dados heterogêneos, com a utilização do conceito de *wrappers* e mediadores [PAP95, HAM98].

Os *wrappers* são tradutores que atuam sobre cada informação e convertem os dados que estão dispostos sob um determinado tipo em um modelo comum de informação[WIE92]. No TSIMMIS, os dados extraídos das diversas fontes são empacotados em objetos OEM (Object Exchange Model). Esse modelo de dados é amplamente aceito em várias pesquisas, podendo ser estudado em [PAP95].

Um módulo do projeto é o tratamento de extração de informações semi-estruturadas, tendo como fonte de informação páginas HTML. O extrator do TSIMMIS recebe um arquivo que contém as especificações da localização dos dados de interesse na página HTML, e a forma de empacotar os dados extraídos em objetos OEM [HAM98]. A especificação da localização da informação está baseada em delimitadores que identificam o início e o fim de cada informação. Isto significa que se alguma das páginas HTML for alterada, esse arquivo de especificação também deve ser modificado. A montagem das especificações é feita de forma manual pelo usuário.

O arquivo de especificação consiste de uma seqüência de comandos na forma: [*variable, source, pattern]*. O termo *source* identifica o texto de entrada; *pattern* informa como encontrar a informação no texto; e *variables* são as variáveis que armazenam o resultado extraído.

A FIGURA 2.5 exibe uma página HTML com informações meteorológicas. Cada linha da página corresponde a uma cidade, e, para cada cidade, são identificadas as previsões de tempo e temperatura para dois dias.

O arquivo de especificação da FIGURA 2.6 contém uma lista de comandos, e cada um destes é delimitado por colchetes. Esse arquivo de especificação está dividido em 5 comandos básicos. O comando inicial (linhas 1 a 4) extrai o conteúdo de seu fonte URL e joga o conteúdo na variável "*root"*. O caracter "#" identifica que o conteúdo completo deve ser extraído. O segundo comando (linhas 5 a 8) especifica que os resultados da aplicação dos delimitadores sobre a variável "root" devem ser colocados na variável "*temperatures".* Os delimitadores descartam qualquer coisa antes da primeira marca </TR> dentro da segunda definição da marca<TABLE> . Desse modo, a variável "*temperatures"* possui os dados do bloco da linha inicial 22 da página HTML. O terceiro comando (linhas 9 a 12) especifica que o conteúdo da variável "*temperatures"* deve ser dividido em pedaços, considerando como delimitador a marca <TR ALIGN=left>. Cada pedaço desses corresponde a uma linha da TABELA de temperaturas. O resultado dessa extração é colocado em uma variável temporária "*_citytemp"* em que o sinal "_" (sublinha) identifica que se trata de uma variável temporária. O quarto comando (linhas 13 a 16) copia o conteúdo de cada célula da matriz que corresponde à variável "*_citytemp"* para a matriz "*citytemp"*. A identificação das células [1,0] determina que deve ser copiada desde a segunda célula, contando do

início (a célula inicial é 0) até a última célula. A primeira célula é descartada, pois contém dados do cabeçalho. O último comando (linhas 17 a 20) extrai os dados de cada célula da matriz e alimenta as variáveis específicas "*country", "c_url" e "city"*.

```
<HTML>
<HEAD>
<TITLE> INTELLICAST: europe weather <TITLE>
<A NAME = "europe"> </A>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH-509>
        <TR>
                <TD COLSPAN=11><I> Click on  acity for local forecasts </I> <BR>
                </TD>
        </TR>
        <TR>
                <TD COLSPAN=11><I> Temperature listed in degrees celsius</I> <BR>
                </TD>
        </TR>
        <TR>
                <TD COLSPAN=11><HR Noshade size=6 width=509>
                </TD>
        </TR>
        </TABLE>
        <TABLE CELLSPACING=0 CELLPADDING=0 WIDTH=514>
              <TR ALIGN=LEFT>
               <TH COLSPAN=2><BR></TH>
               <TH COLSPAN=21><I>Tue, Jan 25, 2000</I></TH>
               <TH COLSPAN=21><I>Wed, Jan 26, 2000</I></TH>
              </TR>
               <TR ALIGN=LEFT>
               <TH> <I> country </I></TH>
                       <TH> <I> city</I></TH>
               <TH> <I> forecast </I></TH>
               <TH> <I> hi/lo </I></TH>
               <TH> <I> forecast </I></TH>
               <TH> <I> hi/lo </I></TH>
                       </TR>
               <TR ALIGN=LEFT>
               <TD> Austria</TD>
               <TD> <A href=http://www.intellicast.com/weather/vie/>Viena</A></TD>
               <TD>snow </TD>
               <TD>-2/-7 </TD>
               <TD>snow </TD>
               <TD>-2/-7 </TD>
                       </TR>
              <TR ALIGN=LEFT>
               <TD> Belgium</TD>
               <TD> <A href=http://www.intellicast.com/weather/bru/>
Brussels</A></TD>
               <TD>ptcldy </TD>
               <TD>3/-4 </TD>
               <TD>ptcldy</TD>
               <TD>3/-4 </TD>
                       </TR>
                ...
        </TABLE>
</HTML>
```

FIGURA 2.5- Fonte HTML – Exemplo TSIMMIS

Ao final do processo de extração, as variáveis que contêm os dados de interesse são empacotadas em objetos OEM, como na FIGURA 2.7. Objetos OEM formam grafos que contêm um único objeto como raiz e vários ou nenhum objetos aninhados. Cada objeto contém um tipo, um valor e um descritor. O descritor identifica o significado do dado, o tipo descreve o valor que pode ser atômico ou um conjunto de subobjetos OEM. No exemplo deste texto, os filhos do objeto raiz são "*temperatures"* do tipo complexo, pois aninha objetos "*city_temp"* , também do tipo complexo. Esses, por sua vez, aninham objetos simples que contêm as informações propriamente ditas. É importante observar que as variáveis temporárias não constituem objetos em OEM.

Objetos OEM são armazenados no repositório LORE (*Lightweight Object Repository*) que já possui sua linguagem, a LOREL (*Lightweight Object Repository Language*) [MCH98]. Muitos outros projetos de pesquisa utilizam OEM como um dos conceitos básicos do processo.

```
[ [“root”,                                                                                                  1
   “get (´http://www.intellicast.com/weather/europe/´)”,                                                    2
   “#”                                                                                                      3
],                                                                                                          4
 [“temperatures”,                                                                                           5
 “root”,                                                                                                    6
“*<TABLE*<TABLE*</TR>#</TABLE>*”                                                                            7
],                                                                                                          8
 [“_citytemp”,                                                                                              9
 “split (temperatures,´<TR align=left>´)”,                                                                  10
“#”                                                                                                        11
],                                                                                                         12
[“city_temp”,                                                                                              13
 “ _citytemp [1:0]”,                                                                                       14
“#”                                                                                                        15
],                                                                                                         16
 [“country,c_url,city, weath_today,hdh_today,low_today,weath_tomorrow,Hgh_tomorrow,low_tomorrow”,          17
 “city_temp”,                                                                                              18
 “*<TD>#</TD>*HREF=#>#</A>*<TD>#</TD># / # </TD>*<TD>#</TD>*<TD># / #”                                     19
]]                                                                                                         20
```

FIGURA 2.6- Arquivo de Especificação – Exemplo TSIMMIS

```
Root        complex {
            Temperature         complex{
                        City_temp complex{
                                    Country     string      “Austria”
                                    City_url    url         http://www..
                                    City        string      “Viena”
                                    Weather_today           string      “snow”
                                    High_today              string      “-2”
                                    Low_today               string      “-7”
                                    Weather_tomorrow        string      “snow”
                                    High_tomorrow           string      “-2”
                                    Low_tomorrow            string      “-7”
                        }
                        City_temp complex{
                                    Country     string      “Belgium”
                                    City_url    url         http://www..
                                    City        string      “Brussels”
                                    Weather_today           string      “ptcldy”
                                    High_today              string      “3”
                                    Low_today               string      “-4”
                                    Weather_tomorrow        string      “ptcldy”
                                    High_tomorrow           string      “3”
                                    Low_tomorrow            string      “-4”
                        }
                        ...
            }
}
```

FIGURA 2.7 - Informações no Formato OEM – Exemplo TSIMMIS

O extrator do TSIMMIS é uma ferramenta flexível que permite a integração de vários dados disponíveis na *Web*. A ferramenta é totalmente manual, e o usuário precisa

analisar a fonte de dados e montar os arquivos de especificação. Além disso, qualquer mudança na página demanda uma alteração no *wrapper*.

2.4 DEByE

A ferramenta DEByE (*Data Extraction by Example)* faz parte de um projeto desenvolvido pela Universidade Federal de Minas Gerais, Brasil.

A extração realizada pela ferramenta DEByE contempla duas etapas básicas: a especificação dos objetos que servem de modelo, feita pelo usuário, e a extração propriamente dita, baseada nesta especificação [LAE99].

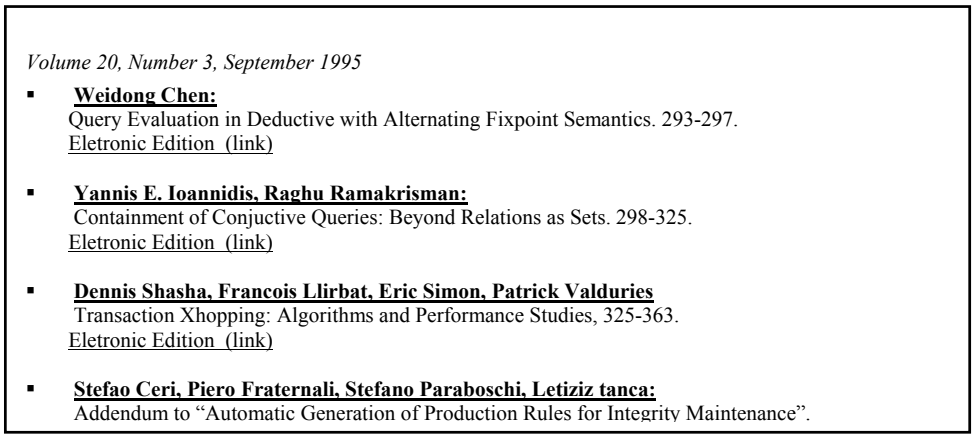
Para especificar alguns objetos de exemplo e descrever a estrutura implícita do documento, o usuário interage através de uma interface gráfica escrita em Java. A interface é constituída por três partes: "*Source Window"*- exibe o documento fonte de exemplo; "*Table Window"* - usada para montagem dos objetos de exemplo; "*Pattern Builder Window"*- exibe os objetos padrões especificados. O usuário marca porções do texto exibidos na janela "*Source Window"* e copia esses pedaços para as colunas da janela "*Table Window"*. Nessa etapa, o usuário se vale das operações de edição cortar e colar oferecidas pela ferramenta. Com pedaços do texto copiados para a janela "*Table Window"*, o usuário modela seus objetos de exemplo em estruturas tabulares e, se for necessário, aninhadas (quando os tipos são complexos). Com os objetos constituídos, é ativada a janela *"Pattern Builder Window"* que gera estruturas de dados chamadas padrões para extração de objetos ou OE *patterns* (*Object Extraction patterns).*

Os OE *patterns* contêm informações sobre a estrutura dos objetos a serem extraídos e sobre o contexto (posição no texto) associado a cada atributo componente dos objetos. As informações sobre o contexto constituem os AVP *pattern* (*Attribute-Value Pair*). Esses pares de atributo e valor relacionam o conteúdo da porção de texto analisada como valor e seu atributo correspondente como objeto. O mesmo atributo pode estar em vários pares, pois pode possuir vários valores. As palavras que cercam tal valor no texto são usadas como um contexto local para localizar o objeto correspondente a esse valor nos demais textos a serem extraídos.

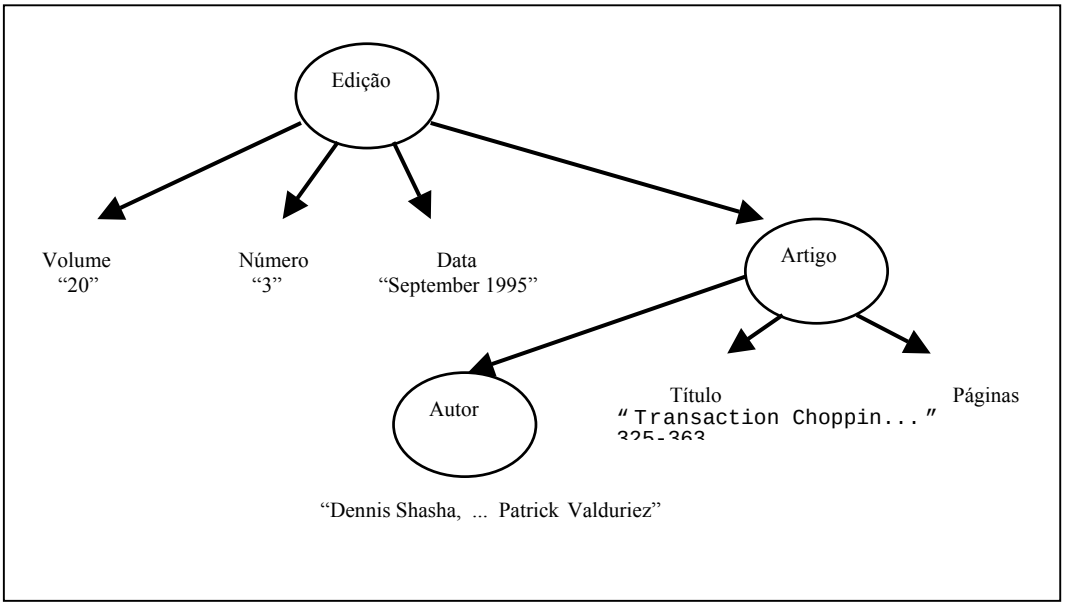
A FIGURA 2.8 (a) exibe uma página HTML que relaciona nomes de autores e seus livros. A FIGURA 2.8 (b) demonstra de forma visual e hierárquica a especificação feita pelo usuário, identificando os OE *patterns* do documento. O objeto raiz é a própria edição do periódico e contém três componentes atômicos (volume, número e data de edição) e um componente complexo (artigo). Este, por sua vez, é composto de três objetos atômicos (título, numeração das páginas e nome do autor).

Na FIGURA 2.9, o usuário escolhe "autor" e o valor "Eric Simon" como objeto de exemplo. Por isso, "Eric Simon" está circulado na FIGURA 2.9 (a) que exibe a porção da página e na FIGURA 2.9 (b) que exibe o fonte HTML. O padrão resultante dessa interação do usuário está demonstrado na FIGURA 2.9 (c) [LAE99a]. O valor do padrão foi substituído por um coringa, o caracter "*", que aceita como combinação qualquer seqüência de caracteres. Esse padrão AVP demonstrado não é utilizado de forma direta. A quantidade de termos a ser considerada no padrão é determinada de forma empírica. O sistema inicia a experiência considerando somente um termo à esquerda e um à direita. Em nosso exemplo, o termo à esquerda seria ">" e à direita

"<". Tenta procurar todos os nomes de autores possíveis com um padrão de apenas um termo em cada lado. Compara a quantidade de valores encontrados em tal pesquisa com o número que o usuário informa que deveria ter sido encontrado. Se o número encontrado for maior, é acrescentado um termo à esquerda e um à direita para um novo teste de padrão. O processo continua até que a quantidade de informações não seja maior que a informada pelo usuário.



(a) Página WEB para Modelo de Objetos



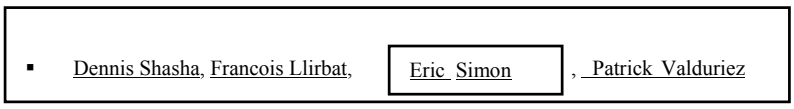
(b) Objetos Modelados
FIGURA 2.8 – Objetos Extraídos – DEByE

A partir dos OE *patterns* gerados, o módulo de extração pode reconhecer e extrair os objetos de páginas com estrutura e contexto similares ao exemplo. Os objetos extraídos são armazenados em bancos de dados tradicionais ou em formato XML.

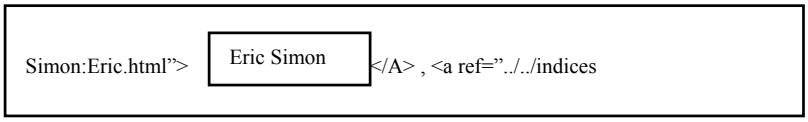
A ferramenta trabalha com duas estratégias de extração: a *top-down* e a *bottom-up*. A estratégia *top-down* [LAE99a], como já diz o nome, parte do ponto mais alto da estrutura e vai refinando até chegar ao seu nível mais baixo. O extrator procura, em cada página do conjunto de páginas similares, objetos que combinem com o padrão

pré-determinado no exemplo. Cada novo objeto é reconhecido em sua totalidade, pois cada um deles é decomposto, visitando todos os seus componentes, até chegar ao nível hierárquico mais baixo (nível atômico).

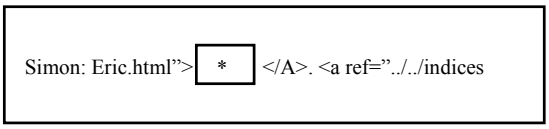
A estratégia *bottom-up* [LAE2000] trabalha com a idéia contrária a *top-down*. Para cada página do conjunto de páginas similares são extraídos todos os AVP *patterns* e armazenados em uma estrutura do tipo lista. Como os valores dos pares AVP são atômicos, o algoritmo extrai todos os objetos do nível mais baixo de decomposição. A partir destes valores são feitas as composições de objetos. A extração inicia dos objetos mais simples que estão nos níveis inferiores da hierarquia para compor os níveis mais altos. A estratégia *bottom-up* é considerada mais completa, pois consegue extrair objetos que estão fora de sua ordem hierárquica, além de trabalhar com objetos parciais.



(a) Página WEB com Valor Demarcado



(b) Fonte HTML com Padrão Identificado



(c) Padrão AVP

FIGURA 2.9 – Exemplo de Padrão AVP - DEByE

Foram realizados experimentos com o DEByE que demonstraram a facilidade de montagem dos padrões através da interface gráfica. Ele possibilita o uso de estruturas aninhadas, o que enriquece a qualidade da informação. Em um primeiro momento foi avaliada a estratégia *top-down.* Entretanto, a *bottom-up* se mostrou mais ágil e mais correta para o tipo de estrutura trabalhada. Observa-se nos resultados apresentados que, após uma modelagem bem feita, o extrator consegue extrair a totalidade dos objetos, ou seja, 100% deles.

Os experimentos e resultados obtidos com essa ferramenta podem ser encontrados em [LAE99].

2.5 W4F

O *World Wide Web Wrapper Factory (W4F)* é uma ferramenta para geração de extratores de fontes HTML, que gera os resultados extraídos em formato XML [AZA99]. O W4F é composto por três módulos independentes: recuperação de páginas que carrega as páginas da WWW, extração das informações, e mapeamento das informações extraídas em formato XML.

A ferramenta é totalmente baseada nas marcas de formatação HTML (*tags)*. Ela analisa o documento HTML, o decompõe, usando as marcas de formatação, e constrói uma árvore abstrata que corresponde a sua estrutura hierárquica. Uma árvore consiste de raiz, nodos internos e folhas. Cada nodo da árvore corresponde a uma marca (ou *tag) HTML.* Para extrair as informações, é realizada uma navegação sobre a árvore. Uma forma de navegação é através da estrutura hierárquica do documento, através do operador ".". Esse caminho inicia pela raiz e avança todas as marcações a serem percorridas até chegar à informação desejada. Utilizando a expressão "html.head.title", extrai o nodo correspondente à marca "<title>" que pertence ao nodo da marca "<head>" que compõe o nodo raiz , a marca "<html>". Outra abordagem de navegação é através do fluxo do documento, utilizando o operador "->". Utilizando a expressão "html->pcdata [1]", é encontrado o segundo dado do texto atravessando a árvore a partir da raiz.

A ferramenta oferece um suporte visual para auxiliar a criação das regras de extração. O usuário marca a porção do texto que contém a informação desejada e a ferramenta exibe o caminho para chegar à informação. Tal caminho pode ser aplicado diretamente na regra. Essa interface somente trata os caminhos através da navegação pela hierarquia do documento. Caso o usuário deseje utilizar o caminho pelo fluxo do documento, expressões regulares ou índices de extensão, deve fazer a montagem manual.

A linguagem utilizada pelo extrator chama-se HEL (HTML *Extraction Language*). A HEL possui algumas propriedades que auxiliam o processo de extração, como: agregação de ".txt" – retorna o conteúdo do nodo, ou seja, o valor do atributo; getAttr – método que retorna os atributos do nodo como um endereço associado; numberof – método que retorna a quantidade de filhos.

A ferramenta W4F trabalha com estruturas complexas, as quais podem ter arbitrárias dimensões, profundidade e níveis de aninhamento. Para auxiliar o usuário nessas situações, a ferramenta W4F oferece algumas facilidades: especificadores de extensão, que definem um subconjunto de filhos de um determinado nodo; expressões regulares com os operadores "match" e "split", que conferem maior granularidade à consulta; operador *fork,* que permite reunir várias informações espalhadas ao longo do texto, buscadas por uma seqüência múltipla de subcaminhos.

Utilizando como exemplo a página Web, que relaciona informações sobre pessoas e sua localização[4], pretende-se obter o nome das pessoas agrupadas por município. A FIGURA 2.10 exibe uma amostra da TABELA que deve retornar. Os nomes em itálico correspondem aos municípios seguidos de seus códigos de endereçamento postal. As demais linhas são a relação do nome das pessoas seguido por seu telefone e endereço. Para responder à consulta, é preciso identificar os municípios e os campos seguintes que correspondem aos nomes de pessoas, como está demonstrado em HEL:

[4] Disponível em http://www.pageszoom.com

```
Html.body.table [i:*] ( .tr [0].td [1].txt
                        #->table [j:*].tr [0] ( .td [1].txt
                        # .td [2].txt
                        # .td [3].txt) )
WHERE html.body.table [i].tr [0].td [1].b [0].numberOf (i) != 0          // condição 1//
AND   html.body.table [i]->table [j].tr [0].td [1].b [0].numberOf (i) = 0, !; //condição 2 //
```

O resultado inicial é uma lista (table [i:*]) de pares. Os elementos do primeiro par da lista devem satisfazer a condição 1 que identifica os municípios, e estes são as linhas correspondentes aos valores 0, 2, 6, 8, 12,14 do índice de extensão. Para cada um desses valores de i, a regra busca as próximas TABELAs (table [j:*]) que corresponde à condição 2 e representa a relação das pessoas. Essa condição 2 termina com o símbolo de corte "!" que busca outras TABELAs até falhar a condição. Para cada vez que a condição 2 é satisfeita, o conteúdo extraído é dividido em três elementos (nome, telefone e endereço).

ESPALION (12500)		
Sahuguet Sylvie	05 65 44 79 11	21 av St. Côme
GABRIAC (12340)		
Sahuguet Jean	05 65 44 90 33	Le Bourg
Sahuguet Paul	05 65 48 52 33	rte Espalion
Sahuguet Pierre-Marie	05 65 48 51 62	lot Causse
MURET LE CHATEAU (12330)		
Sahuguet Henri	05 65 46 94 31	Le Bourg
RODEZ (12000)		
Sahuguet Nadine	05 65 68 59 34	49 r Grandet
Sahuguet Paul	05 65 42 06 17	18 r Chapelle
Sahuguet Rémi	05 65 42 17 5	La Parisiense 6 av Paris
SAINT GENIEZ DÓLT (12130)		
Sahuguet Philippe	05 65 47 41 73	Champ de la Place 253 av Espalion
SALLES LA SOURCE (12330)		

FIGURA 2.10 – Parcial de uma Página HTML - W4F

A HEL necessita de uma forma estruturada para armazenar as informações, chamada de estruturas NSL (*Nested String Lists*). As estruturas NSL representam a composição de um dado complexo, pois podem ser do tipo lista de NSL. Para os dados atômicos, aceita os tipos nulos e cadeia de caracteres (string simples). Uma NSL representativa do exemplo anterior (FIGURA 2.10) é a seguinte:

```
[ [ "Espalion (12500)", [ [ "Sahuguet Sylvie",        "05 65 44 79 11",      "21 av St. Come"] ],
  [ "Gabriac (12340)",  [ [  "Sahuguet Jean",         "05 65 44 90 33",      "Le Bourg"      ],
                          [  "Sahuguet Paul",         "05 65 48 52 33",      "rte Espalion"  ],
                          [  "Sahuguet Pierre-Marie", "05 65 48 51 62",      "lot Causse"    ] ], ...
]
```

O extrator interpreta a NSL e gera o resultado obtido do processo de extração no formato XML.

Essa ferramenta depende totalmente das marcas de formatação. O processo de extração está baseado na estrutura hierárquica dessas marcas. Qualquer alteração no documento implica em modificações no extrator.

Exemplos e experimentos feitos com a ferramenta estão ilustrados em [AZA99a, AZA99b].

2.6 JEDI

JEDI (*Java based Extraction and Dissemination of Information)* é um projeto que trata da criação de *wrappers* e mediadores para extrair, combinar e reconciliar informações de diferentes fontes. JEDI é um projeto amplo composto por vários componentes: um extrator para fontes externas de dados, um modelo para descrever estruturas de fontes heterogêneas de dados, e uma linguagem de consulta e manipulação que integra visões de múltiplas fontes [HUC98].

A ferramenta JEDI utiliza gramáticas livres de contexto para descrever a estrutura dos documentos. A gramática é um conjunto de regras que possuem um nome e um corpo, e este contém a expressão relacionada. As expressões podem ser combinadas formando produções mais complexas:

- E1 E2 E3 – seqüência de três expressões.
- E? – Expressão opcional.
- E* - Expressão pode ser repetida opcionalmente
- E+ - Expressão repetida
- E1|E2 – Valida a primeira ou a segunda expressão.
- (E) – Agrupamento da expressão.

As produções terminais utilizam expressões regulares para realizar a combinação de dados:

- "." (ponto final) – Combina qualquer caracter.
- [A-Z] – Combina somente letras maiúsculas.
- [^A-Z] – Combina qualquer caracter fora da extensão de letras maiúsculas.
- "texto" – Combina a seqüência de caracteres entre aspas.

Para realizar algumas validações semânticas, a ferramenta JEDI disponibiliza predicados que possam verificar a extensão de um determinado dado. Tais validações são realizadas pelas cláusulas "if " e "accept". A cláusula "if" valida o dado parametrizado em "$$" . Se a validação retornar verdadeira, a produção é aceita, caso contrário é rejeitada.

A ferramenta JEDI utiliza variáveis para armazenar informações que serão utilizadas em momentos futuros. Para trabalhar com essas variáveis, disponibiliza o operador "=" (assinala o valor diretamente a variável) e o operador "+=" (acumula o valor na variável). O exemplo que segue agrega blocos de codificação à gramática. O programa resultante dessa junção extrai os dados e os mapeia para elementos do tipo *XML* que, em um segundo momento, são transformados para a linguagem XML.

O exemplo utilizado na FIGURA 2.11 extrai o dia, o mês e o ano separadamente e compõe a data completa. É possível observar dois aspectos do extrator: como uma produção pode invocar outras produções; e a utilização de variáveis para armazenar informações que são utilizadas mais tarde, como é o caso de "m", "d" e "y". Esse

exemplo está gerando as informações extraídas em formato XML, como pode ser visto no comando composto “do”.

A gramática gera um autômato com várias possibilidades de estados e transações. A partir da leitura do documento, cada caracter é validado no estado atual do autômato. Se o estado for validado, uma transição ocorre, sendo que a leitura do documento também avança. Caso o estado não seja validado, o algoritmo tenta a validação dos demais.

```
rule day is [0-9] + end
        rule month      is [0-9] + end
        rule year       is [0-9] + end
        rule date: res is
                //associa a variável res ao resultado da produção
                m=month ( ) “/” d=day ( ) “/” y=year ( )
                // assinala as variáveis mês, dia e ano
                do
                        res=xml_element (“date”);
                        res.addChild (xml_element (“month”,m));
                        res.addChild (xml_element (“year”,y));
                        res.addChild (xml_element (“day”,d));
                end
        end
        rule date is
                (
                  .+  // salta  caracteres os quais não correspondem ao formato de date
                  | list += date ( )
                )+
                do
                        res = xml_element (“dates”);
                        res.addChildren (list);
                        res.prettyPrint ( ) ;
                end
        end
```

FIGURA 2.11 – Exemplo de Especificação - JEDI

2.7 Embley

A idéia básica do projeto coordenado por Embley é extrair informações de dados semi-estruturados através de um modelo conceitual de instâncias, denominado de ontologia. Esse modelo contém palavras-chave ou padrões de combinação que identificam as informações no documento. As informações são extraídas e estruturadas de acordo com a ontologia, ficando disponível para consultas futuras. Segundo [EMB99], esse método fundamenta-se no fato de que muitos documentos *Web* contêm constantes reconhecíveis que, juntas, podem descrever o conteúdo do documento.

A arquitetura do processo de extração de dados pode ser descrito em três módulos fundamentais: analisador da ontologia, extrator de dados e gerador de resultados.

O analisador da ontologia prepara as informações necessárias para a atuação do extrator. É necessário obter a ontologia do domínio em questão que possui os objetos

relacionados e os *data-frames* com o vocabulário. A FIGURA 2.12 exibe um documento HTML que relaciona obituários. Essa página noticia o óbito, informando o nome da pessoa, seus familiares e informações do cerimonial. Uma ontologia representativa do domínio relacionado a esta página pode ser visualizado na FIGURA 2.13. O modelo de instâncias dos objetos é baseado no *Object-oriented Systems Model (OSM)* [EMB92]. Os objetos são representados por retângulos ponteados, quando são léxicos e por retângulos com linha cheia, quando são não léxicos. "Idade" e "Data Aniversário" são objetos léxicos, pois possuem representação direta. "Pessoa Falecida" e "Velório" são objetos não léxicos, pois são formados por outros objetos. Os relacionamentos entre os objetos são identificados pela linha que conecta os retângulos. O conceito de generalização/especialização é representado pelo sinal de dois pontos após o nome do conjunto de objetos, como, por exemplo, "Data Aniversário" é uma especialização de "Data".

Brian Fielding Frost

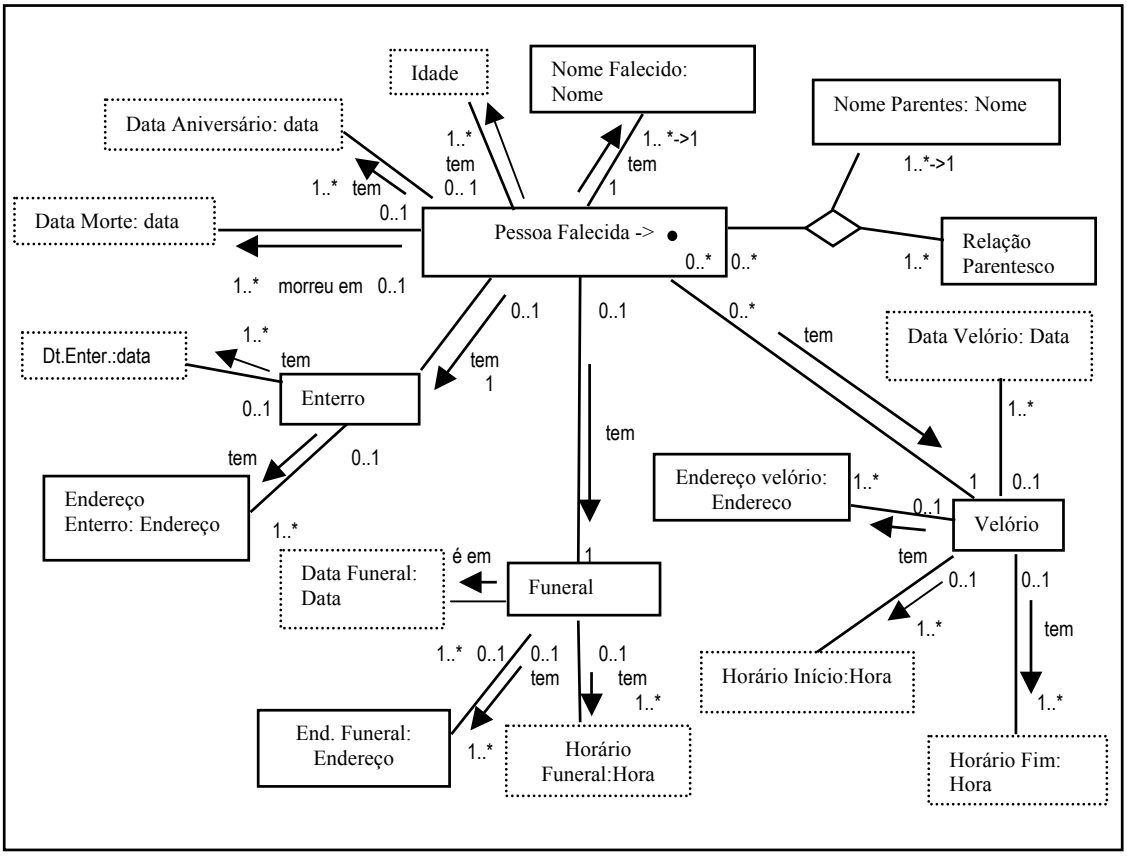
Our beloved Brian Fielding Frost, age 41, passed away Saturday morning, March 7, 1998, due to injuries sustained in na automobile accident. He was born August 4, 1956 in Salt Lake City, to Donald Fielding and Helen Glade Frost. He married Susan Fox on June 1, 1981.

He is survived by Susan; sons Jordan (9), Travis (8), Bryce (6); parents, three brothers, Donald Glade (Lynne), Kenneth Wesley (Ellen), Alex Reed, and two sisters, Anne (Dale) Elkins and Sally (Kent) Briton. A son, Michael Brian Frost, preceded him in death.

Funeral services will be held at 12 noon Friday, March 13, 1998 in the Howard Stake Center, 350 South 1600 East. Friends may call 5-7 p.m. Thursday at Wasatch Lawn Mortuary, 3401 S. Highland Drive, and at the Stake Center from 10:45-11:45 ªm. Friday. Interment at Wasatch Lawn Memorial Park.

FIGURA 2.12 – Página HTML de um Obituário – Exemplo de Embley

Além do modelo, a ontologia fornece conhecimento sobre tais objetos através de seus *data-frame*s [EMB98]. Se o *data-frame* está associado a um conjunto de objetos léxicos, ele descreve o padrão da cadeia de caracteres para suas constantes. Por exemplo, um *data-frame* "Data" combina datas usando expressões regulares, um *data-frame* "Nome" usa uma combinação de expressões regulares e um vocabulário que expressa pré-nomes e sobrenomes. A FIGURA 2.14 exibe uma parcial dos *data-frames* da ontologia exemplificada, relacionando os objetos "Nome", "Nome Parentes" e "Parentesco". O número entre colchetes corresponde à expectativa máxima do tamanho da constante do *data-frame*. Um *data-frame* declara os padrões das constantes, os padrões de palavras-chave e o vocabulário para as constantes. Para o *data-frame* "Nome" são utilizadas listas de nomes comumente usadas e armazenadas em arquivos externos. A regra para "Nome", no exemplo da FIGURA 2.14, expressa que o primeiro nome deve corresponder a um dos nomes de "Pre-Nome" que estão em um arquivo de nomes, e este é seguido por um ou mais espaços em branco, seguido por um dos nomes de "Sobrenome" que estão em outro arquivo de nomes. A segunda regra para "Nome" especifica que deve combinar um conjunto de letras iniciado por uma letra maiúscula. Seguindo o exemplo da FIGURA, "Nome Parentes" é uma especialização do *data-frame* "Nome". Como em muitos casos, o nome do cônjuge do parente aparece entre parênteses no meio do nome, e somente interessa o nome do parente, a segunda regra filtra e descarta o que está entre parênteses.




```
Nome matches [80] case sensitive
        Constant
                {extract pre-nome, "\s+" , Sobrenome;} ...
                {extract " [A-Z] [a-zA-Z]*\s+  ( [A-Z]\.\s+)?", Sobrenome;{,   ...
        Léxicon {
                Pre-nome case insensitive;
                Filename "PRE-NOME.DICT";
                },{
                Sobrenome case insensitive;
                Filename "sobrenome.dict";
                };
end;
Nomes Parentes matches [80] case sensitive
        Constant { extract Pre-nome, "\s*\ (", Prenome, "\)\s*", Sobrenome;
                        Substitute "\s*\ ( [^)*\)" -> ""; ...
end; ...
Relação Parentesco matches [14]
        Constant
                {extract "brother";             Context "\bbrothers?\b";},
                {extract "sister";                      Context "\bsisters?\b";}, ...
                {extract "step-?father";        Context "\bstep-?father\b"; filter "-";},
                {extract "\bstepfather\b";} ...
        Keyword
                "\bspouse\b",
                "\bmarried\b", ...
end; …
```

FIGURA 2.14 - Amostra de *Data-Frame* de Obituários – Exemplo de Embley

O módulo extrator inicia seu trabalho a partir das informações obtidas da ontologia (modelo e *data-frame).* O primeiro passo é construir uma árvore da página

HTML baseando-se nas suas marcas de formatação (*tags)*. Essa árvore é chamada de *tag tree* e representa a estrutura hierárquica do documento HTML através de suas marcas de formatação (*tags)*. Deduzindo-se que as informações de interesse estão localizadas na parte mais larga da árvore, aplica-se uma série de heurísticas para escolher uma marca entre as marcas candidatas. Com a marca desejada identificada, o sistema marca as linhas que pertencem a essa marca no documento HTML, inserindo os caracteres "#####" e removendo outras marcas de formatação. O resultado do processo pode ser visto na FIGURA 2.15 que exibe o fonte HTML já com as marcas de formatação eliminadas e com a seqüência "#####" identificando a porção do texto que contém a informação desejada.

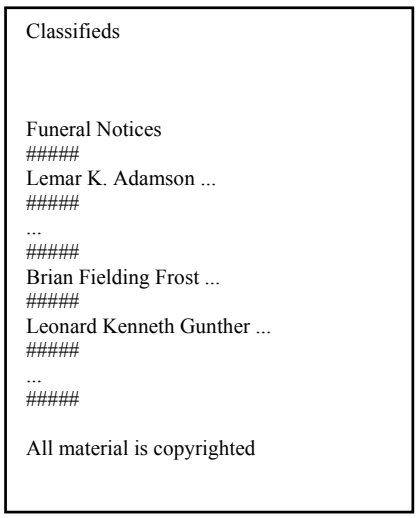


FIGURA 2.15 – Página HTML com Informação Identificada – Exemplo de Embley

A partir das informações identificadas, o gerador de resultados alimenta uma TABELA de registros de dados formado por tuplas que identificam a posição para cada constante ou palavra-chave reconhecida. Essas tuplas são formadas pelo descritor/cadeia de caracteres/posição. Como pode ser visto na FIGURA 2.16, o descritor é o nome da constante ou palavra-chave. O segundo campo é o valor que corresponde ao descritor extraído do documento. O último campo é a posição inicial em que a informação se encontra no texto. Para resolver os conflitos encontrados nessa etapa são aplicadas novamente heurísticas para determinar a melhor opção. Um exemplo de conflito aparece na FIGURA 2.16, onde "DataAniversário" e "DataMorte" estão relacionados ao mesmo valor de data – "dia 4 de agosto de 1956". O gerador de instâncias do banco de dados usa essa TABELA de registros (FIGURA 2.16) juntamente com a descrição prévia dos registros (FIGURA 2.17) para gerar as tuplas que alimentam o banco de dados. O gerador de instâncias do banco de dados executa esse processo de geração criando comandos de inserção para o banco, reorganizando a informação extraída em um formato estruturado.

Apesar de selecionar a região de interesse através da classificação das marcações (*tags)* candidatas, o extrator não é dependente do formato do documento. Esse critério apenas isola uma porção do texto. O fator imprescindível da extração é a associação correta das palavras-chave nos *data-frames* ligados à ontologia, o que permite torná-lo um método independente de contexto. Ele pode ser facilmente estendido a outros tipos de documentos, como textos planos, por exemplo.

Os experimentos mostram bons resultados para documentos ricos em dados e limitados em sua extensão. Os resultados podem ser obtidos em [EMB98, EMB98a, EMB99, EMB99a, EMB99b].

Nome Parentes | Brian Fielding Frost | 1 | 20
NomeFalecido | Brian Fielding Frost | 1 | 20
Nome Parentes | Brian Fielding Frost | 36 | 55
NomeFalecido | Brian Fielding Frost | 36 | 55
Keyword (idade) | age | 58 | 60
Idade | 41 | 62 | 63
DataAniversario | March 7, 1998 | 96 | 108
DataMorte | March 7, 1998 | 96 | 108
DataEnterro | March 7, 1998 | 96 | 108
DataFuneral | March 7, 1998 | 96 | 108
DataVelorio | March 7, 1998 | 96 | 108
RelacaoParentesco | parent | 172 | 212
Keyword (DataAniversario) | born | 172 | 175
DataAniversario | August 4, 1956 | 177 | 190
DataMorte | August 4, 1956 | 177 | 190
DataEnterro | August 4, 1956 | 177 | 190
DataFuneral | August 4, 1956 | 177 | 190
DataVelorio | August 4, 1956 | 177 | 190
NomeParentes | Donald Fielding | 214 | 228
NomeFalecido | Donald Fielding | 214 | 228
NomeParentes | Helen Glade Frost | 234 | 250
NomeFalecido| Helen Glade Frost | 234 | 250
Keyword (relaçaoparentesco) | married | 257 | 263
RelaçaoParentesco | spouse | 257 |263

FIGURA 2.16 - Entradas na TABELA de Registros – Exemplo de Embley

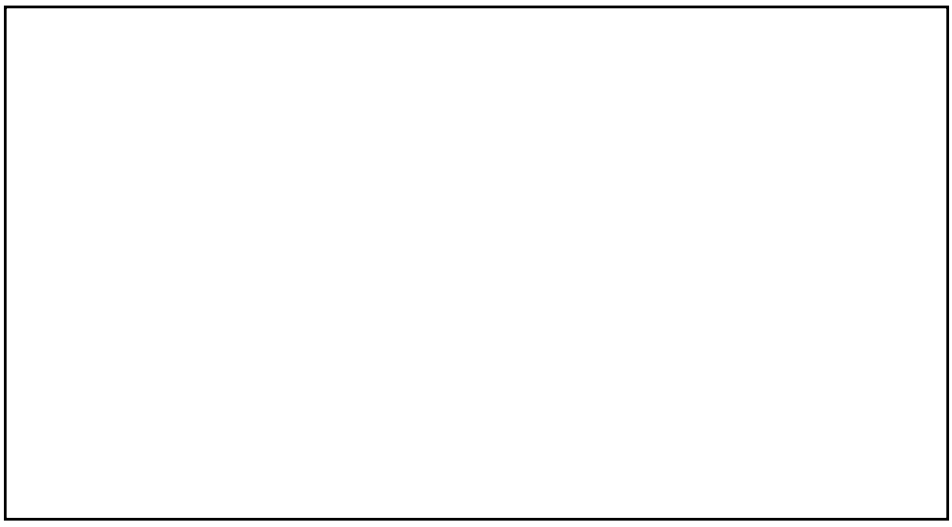


FIGURA 2.17 - Parte da Descrição de Registros Gerados - Exemplo de Embley

2.8 Comparativo dos Métodos

O objetivo da comparação é analisar os métodos segundo os critérios descritos no início desta sessão. Esse trabalho não objetiva eleger o melhor método, porque cada um tem vantagens sobre os outros conforme o aspecto que for considerado. O resumo dessa comparação dos métodos de extração está relacionado na TABELA 2.1. A seguir, o comparativo dos métodos segundo os critérios anteriormente estabelecidos:

a) Tipos de Documentos Manipulados

A maioria das ferramentas utiliza como entrada do processo de extração documentos no formato HTML. Além de ser um tipo de documento amplamente difundido na WWW, possui as marcas de formatação que oferecem indicações de sua estrutura. Documentos HTML são mais regulares quando construídos por ferramentas de geração que precisam de regras para a formatação das páginas [ADE99].

Os extratores do Editor e do TSIMMIS foram criados para tratar de documentos HTML, até porque fazem parte de projetos para a WWW. A ferramenta W4F é um método exclusivo para HTML, pois sua fundamentação é a estrutura hierárquica das marcas, descartando a possibilidade de lidar com outros tipos de documentos. A ferramenta NoDoSe tem a preocupação fundamental de estudar textos planos. Seu método de extração baseia-se na estrutura hierárquica dos documentos e nas teorias que deduzem os delimitadores da informação. Ele trata também de documentos HTML, mas sem utilizar exclusivamente as marcas de formatação. O método de Embley e do DEByE, apesar de sempre fazerem referências a páginas HTML, também não dependem das marcas de formatação, podendo ser estendidos a outros formatos.

b) Recuperação da Estrutura

A maioria dos métodos estudados emprega a abordagem de esquema a *posteriori*, ou seja, definem a estrutura a partir das instâncias dos dados. Essa abordagem faz com que as mudanças no extrator sejam mais freqüentes, pois seu modelo de dados varia conforme as instâncias de sua base. O projeto LORE [GOL97, ABI2000, ABI97] proporcionou um estudo completo sobre esta abordagem, através do *Data Guides*[MCH98, GOL99].

O único método estudado que utiliza a abordagem de esquema a *priori* é o de Embley. Ele associa um modelo conceitual (ontologia) ao domínio no qual estão inseridos os documentos. Dessa forma, as regras de extração contemplam o relacionamento semântico dos objetos. Essa última abordagem torna os extratores mais estáveis, pois refletem o domínio do problema que é menos susceptível a alterações do que as instâncias do banco. Além disso, possui relações semânticas que auxiliam na formulação e implementação de consultas.

c) Dependência de Contexto

Os extratores são baseados em regras que utilizam delimitadores para localizar a informação, são seus termos predecessores ou sucessores.

A maioria das abordagens depende exclusivamente da vizinhança que circunda a informação desejada. Algumas delas utilizam-se de outras regras, reduzindo o grau de dependência dos delimitadores.

O extrator W4F é totalmente dependente de contexto, pois a busca da informação está baseada exclusivamente nas marcas de formatação do HTML. Um grande problema de extratores baseados exclusivamente em marcas de formatação é o fato dos documentos HTML não serem bem formados. Muitos documentos HTML não utilizam marcas de fim para encerrar um bloco aberto por sua marca de início

correspondente, o que pode provocar erros na extração. As outras abordagens utilizam-se de delimitadores da informação, que são indicadores do contexto onde a informação está localizada. Entretanto, ferramentas como o NoDoSe e o DEByE utilizam também o conceito de dados complexos para compor a informação, não dependendo somente do contexto. O extrator de Embley é o que mais se aproxima da independência, pois utiliza, no lugar dos delimitadores, palavras-chave ou constantes que, normalmente, acompanham a informação.

Quanto menor for o grau de dependência do contexto, mais estável é o extrator, pois exige menos alterações em seu código.

d) Nível de Automaticidade do Processo

Nenhuma das abordagens estudadas é totalmente automática. Uma vez que dados semi-estruturados não possuem uma estrutura definida, é necessário que as abordagens contemplem uma fase de recuperação ou definição da estrutura implícita. É essa definição que necessita da interferência humana. Ferramentas totalmente manuais tornam o processo de extração exaustivo, exigindo um usuário experiente em informática. A característica principal que indica uma ferramenta semi-automática é a capacidade de aprender a estrutura com alguns exemplos, podendo extrair informações automaticamente de documentos semelhantes.

A ferramenta NoDoSe é semi-automática, pois, através da demonstração de um elemento relacionado ao objeto, tenta construir e validar suas teorias confrontando com os exemplos feitos pelo usuário. As ferramentas DEByE e Editor também são semi-automáticas, pois o usuário demonstra a exemplificação de um documento, e a ferramenta passa a tratar de forma automática os documentos semelhantes. A abordagem de Embley também pode ser classificada como semi-automática, pois o conhecimento é embutido nas ontologias e toda extração que utilizar aquela ontologia tem acesso a esse conhecimento prévio. O DEByE e o NoDoSe oferecem ferramentas com interface gráfica para facilitar a interação com o usuário.

e) Tipos de Dados

Esse item não trata especificamente do reconhecimento de todos os tipos de dados explicitados pela ferramenta. É importante que existam estruturas capazes de representar dados aninhados ou complexos. Dados complexos podem ser compostos por outros dados complexos ou atômicos. Essa estrutura de aninhamento é importante para referenciar o relacionamento entre as informações.

Pode-se observar que o extrator Editor trata somente tipos de dados atômicos. O NoDoSe explicita todos os tipos de dados atômicos e complexos com que opera. Pode-se deduzir que os demais métodos utilizam dados complexos, pois utilizam artifícios de aninhamento das informações.

f) Regras de Extração

As formas de representação das regras de extração são as mais variadas. Tais regras são baseadas normalmente em variações de gramáticas e expressões regulares. As expressões regulares são mais flexíveis para trabalhar, mas não oferecem o poder de

representar uma composição de objetos como pode ser feito com as gramáticas. As gramáticas têm uma exata descrição da estrutura do documento, mas se adaptam dificilmente a irregularidades [CRE98]. Os extratores TSIMMIS e W4F utilizam somente delimitadores de contexto em uma variação de expressões regulares. NoDoSe utiliza teorias que delimitam a informação e são representadas em expressões regulares. Os extratores Editor e JEDI utilizam gramáticas para representar a complexidade das regras. O extrator DEByE utiliza os OE *patterns,* que possuem os padrões de combinação e a composição de dados complexos. Já o extrator de Embley utiliza os *data-frames* para registrar as palavras-chave que não deixam de ser combinações.

g) Processo de Extração

Utilizando as regras de extração como base, os extratores adotam estratégias próprias para o algoritmo de extração. Genericamente, os algoritmos podem ser classificados em processos de extração *top-down* ou *bottom-up*. Processos de extração *top-down* partem dos dados mais complexos para os mais simples, e a extração *bottom-up* faz o caminho contrário. Esta é uma forma generalizada de enxergar os algoritmos, pois cada ferramenta possui seus próprios artifícios.

A ferramenta Editor conduz o processo de extração através do programa montado pela reestruturação do documento. A ferramenta NoDoSe utiliza um minerador que valida suas teorias. O extrator TSIMMIS possui um programa de especificação que processa a extração do início ao fim do documento. O projeto DEByE aplicou duas abordagens de extração, top-down e bottom-up, e escolheu como melhor técnica a extração bottom-up.

h) Resultado da Extração

Todas as abordagens extraem as informações e as organizam em um novo formato estruturado de dados. Dessa forma, o usuário pode utilizar outras ferramentas já existentes para acessar a informação já organizada.

A linguagem XML aparece como a alternativa preferida dos métodos. Pela sua estrutura, padronização e difusão justifica-se essa escolha. As ferramentas NoDoSe, DEByE, W4F, JEDI e Embley geram o resultado em XML. As ferramentas DEByE e Embley também oferecem a opção de gerar TABELAs em bancos de dados. O extrator Editor somente gera o resultado em bancos de dados. O extrator TSIMMIS mantém o resultado em OEM, que é o modelo padrão de seu ambiente de tratamento de dados heterogêneos.

TABELA 2.1 – Quadro Comparativo Métodos de Extração

Critérios	Editor	NoDoSe	TSIMMIS	DEByE	W4F	JEDI	Embley
Tipos de Documentos Manipulados	HTML	HTML/Textos	HTML	HTML	HTML	Fontes Heterogêneas	HTML/textos
Recuperação da Estrutura	*Posteriori*	*Posteriori*	*Posteriori*	*Posteriori*	*Posteriori*	*Posteriori*	*Priori*
Nível de Dependência de Contexto	Alto	Médio	Alto	Médio	Alto	Médio	Baixo
Nível de Automaticidade	Semi-automático	Semi-automático	Manual	Semi-automático	Manual	Não se refere	Semi-automático
Interface Gráfica	Não	Sim	Não	Sim	Sim	Não	Não informa
Tipos de Dados	Atômicos e complexos	Atômicos e complexos	OEM-complexos	OE – complexos	Cadeia de caracteres e listas	Complexos e atômicos	Complexos e atômicos
Regras de Extração	Gramática	Teorias com delimitadores	Padrões de combinação	*OE Patterns*	Marcas de HTML (*tags)*	Gramática	*Data-frames* e ontologia
Processo de Extração	Segue programa montado	Minerador através de teorias com delimitadores	Segue programa montado	Top-Down e Bottom-up	Através da árvore de navegação	Autômatos Finitos Não-determinístico	Combinação *data-frames*
Resultado da Extração	TABELAs bancos de dados	XML, arquivos texto ou banco de dados	OEM	Bancos de dados ou XML	XML	XML	Bancos de dados
Arquitetura Aberta	Sim – vários componentes	Sim – vários componentes	Sim- não trata de outros componentes	Não contempla	Não contempla	Sim	Não trata outros componentes

3 Extração Semântica Através de Exemplos e Modelo Conceitual

A utilização de um modelo conceitual do domínio possibilita o reconhecimento das relações semânticas dos objetos, agregando mais conhecimento ao extrator, além dos aspectos sintáticos. O modelo é importante nas diversas fases de manipulação de dados semi-estruturados: consulta, extração, materialização e na própria integração de seus vários componentes. Por essas razões, o presente trabalho segue a abordagem de esquemas *a priori*.

Conforme demonstrado na Seção 2, o único método que utiliza esquemas *a priori* é o de Embley, através do conceito de ontologias. Embley desenvolveu um método voltado a documentos com pouca estrutura e sem delimitadores, como textos escritos livremente. Por tratar desse tipo de documento, o método opera com extração de registros, não oferecendo tratamento de listas aninhadas (tipos complexos).

Diferente de Embley, o método proposto no presente trabalho objetiva extrair informações de documentos que possuem uma estrutura hierárquica definida, como é o caso de páginas HTML. Por se tratar de documentos com alguma estrutura implícita, é possível fazer com que o extrator aprenda com algum exemplo e possa extrair de forma automática as informações de documentos semelhantes. Este processo de extração manipula informações com estruturas mais complexas que Embley, pois trata de dados atômicos e complexos, como listas aninhadas.

Pode-se dizer, de forma resumida, que o presente trabalho de dissertação apresenta um método para extração semântica de dados semi-estruturados através de exemplos e modelo conceitual.

É uma ferramenta de extração semântica porque todo processo está baseado em um modelo conceitual, chamado ontologia, que possui conhecimento sobre o domínio do problema, descrevendo seus objetos, relacionamentos e restrições. O conhecimento vai sendo enriquecido com a evolução do processo de extração. Conforme descrito anteriormente, o método se enquadra na abordagem de esquemas *a priori*. Os documentos são agrupados em classes, e cada classe pertence a um determinado domínio, relacionando-se desta forma com sua ontologia. A Subseção 3.1 deste capítulo trata do conceito de ontologias, e a Subseção 3.2 de classes de documentos.

Em dados semi-estruturados é difícil a utilização de um processo de extração totalmente automático, pelo fato de não existir uma definição prévia de sua estrutura. É necessária a interferência humana na etapa inicial de recuperação da estrutura implícita dos documentos. O resultado desse processo de exemplificação é uma gramática que possui conhecimento sobre a estrutura do documento e que contém as regras necessárias ao processo de extração. A proposta desse trabalho é dita semi-automática porque o usuário deve definir um primeiro exemplo da estrutura da classe de documentos, como está descrito na Subseção 3.3 desse capítulo. A partir da gramática, a ferramenta extrai de forma automática as informações dos demais documentos da classe, como está descrito na Subseção 3.4.

As informações extraídas são organizadas em uma nova forma estruturada de dados – *XML (eXtensible Markup Language),* conforme explicado na Subseção 3.5. A partir dessa nova estruturação, o usuário pode obter respostas objetivas a suas consultas, através de outras ferramentas e linguagens apropriadas.

A FIGURA 3.1 ilustra as principais etapas do processo de extração, descritas resumidamente nesse texto inicial. As principais etapas do processo de extração são as seguintes:

a) *Montagem do Primeiro Exemplo*: define a estrutura do documento em uma gramática, a partir de uma ontologia e da interação com o usuário.
b) *Extração Automática:* passa a operar sobre os demais documentos da classe. A partir da gramática definida extrai automaticamente as informações.
c) *Geração do Resultado:* organiza as informações extraídas em formato XML.

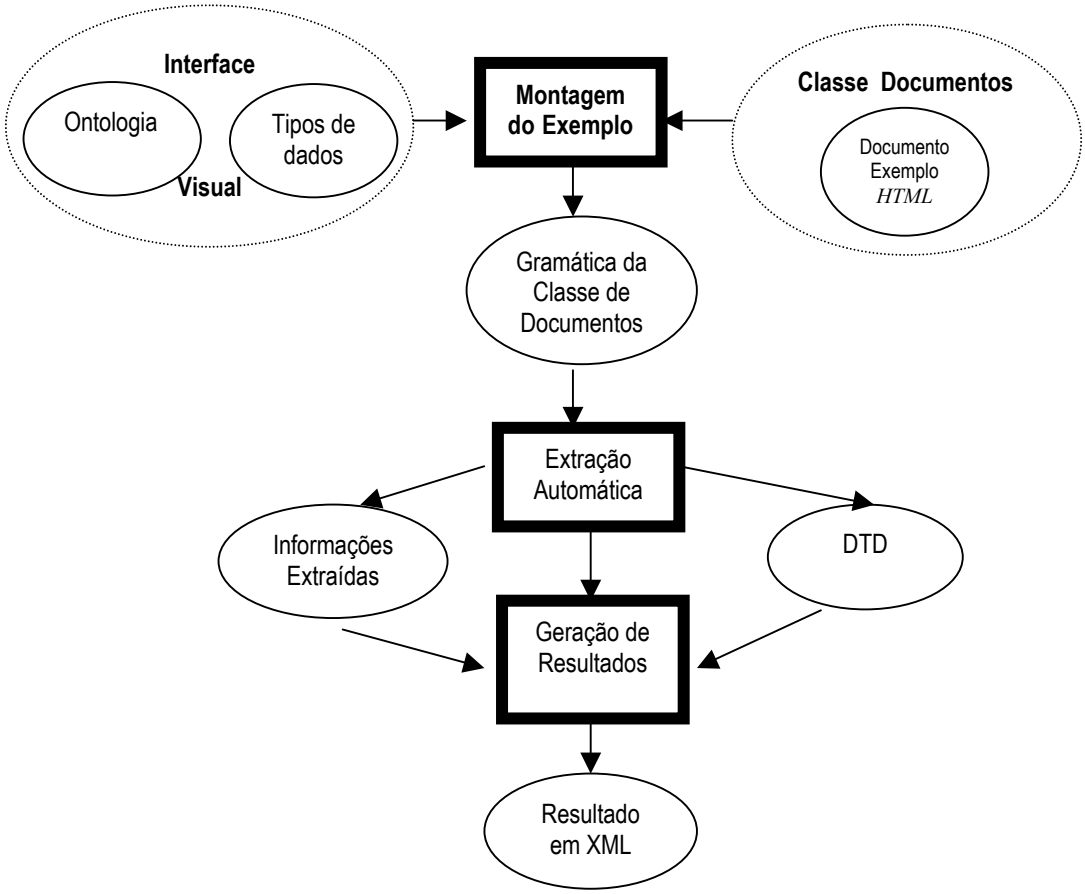


FIGURA 3.1 – Arquitetura do Processo de Extração

3.1 Modelo Conceitual – Ontologias

Ontologias estão sendo utilizadas em várias pesquisas como modelos para representação de dados semi-estruturados [EMB98, ERD99]. O conceito de ontologia supre a carência de um esquema pré-estabelecido de representação das informações, oferecendo um modelo semântico aos conceitos. Maiores detalhes sobre a conceituação e classificação de ontologias podem ser encontrados em [GRU2000, GUA97, GUA2000, GUA94, GUA95].

A ontologia fundamenta-se em representar o conhecimento de um determinado domínio e não de uma solução isolada para um problema particular. O conhecimento sobre um domínio é mais permanente e universal, possibilitando que um número maior de aplicações possa compartilhar das informações.

No método proposto, os documentos são associados a ontologias. Todo o conhecimento que a ontologia tem sobre o domínio passa a servir como um guia para o processo de montagem do primeiro exemplo, o qual define a estrutura do documento. A ontologia fornece os objetos que podem ser utilizados no processo de extração. Os relacionamentos entre os objetos e as restrições de integridade delimitam as relações de aninhamento e os tipos de dados que podem ser assumidos para cada objeto.

Os processos de extração vão ocorrendo e novos conhecimentos são agregados à ontologia, como os tipos de dados normalmente definidos para o objeto e os delimitadores mais utilizados para buscar a informação. Esse novo conhecimento auxilia não somente na fase de montagem do primeiro exemplo, mas também no processo de extração automática, como será explicado nas seções seguintes.

Essa metodologia trabalha com dois tipos de ontologia: particular e universal. A ontologia particular possui os objetos daquele determinado domínio. Já a ontologia universal possui os objetos que sempre são definidos da mesma forma, independente do domínio da aplicação. Um exemplo de um objeto que se enquadra em uma ontologia universal é o nome de uma pessoa. Um nome de pessoa pode ser definido como um conjunto de caracteres alfabéticos iniciado por um caracter maiúsculo e, sempre é assim definido, independente do domínio. Quando esse objeto for empregado em uma ontologia particular, ele herda as definições que possui na ontologia universal.

Não existe preocupação, neste momento, em escolher a melhor metodologia para a representação de uma ontologia, até porque não existe um consenso nessa questão, e a variedade de proposições é muito grande [LOP99, CRA99, VER82]. Uma vez que a proposta é oferecer uma ferramenta visual para o usuário interagir, a opção foi por uma representação gráfica que segue um modelo de diagrama E-R (Entidade-Relacionamento) modificado.

Um exemplo desse modelo de representação pode ser encontrado na FIGURA 3.2. O do domínio modelado refere-se a publicações literárias e contém os seguintes objetos: “anoedição”, “autores” e seus dados de identificação, “título”, “editora” e “preço”. Um objeto “anoedição” possui várias publicações. Um objeto “publicação” é escrito por vários “autores”, possui apenas um “título”, é publicado por um objeto “editora”, e pode ser vendido por um “preço”. Para cada objeto “autor” de “publicação” há um “endereço profissional” e um ”endereço residencial”, existe um único objeto “nome” para identificá-lo e um “e-mail”. Essa representação será utilizada em todas as exemplificações feitas no decorrer do texto.

Nessa representação, o domínio do mundo real é representado por um grafo, os nodos do grafo são os objetos, e os arcos são os relacionamentos entre eles. Em cada arco é descrita a cardinalidade do relacionamento associado. Os objetos podem ser léxicos ou não léxicos. Os objetos léxicos são a informação atômica representada no documento, identificados por retângulos com linhas pontilhadas, como, por exemplo, os

objetos “preço” e “título”. Os objetos não léxicos são a informação complexa ou composta, não representada diretamente no documento, indicados por retângulos com linhas cheias, como, por exemplo, os objetos “autor” e “publicação”.

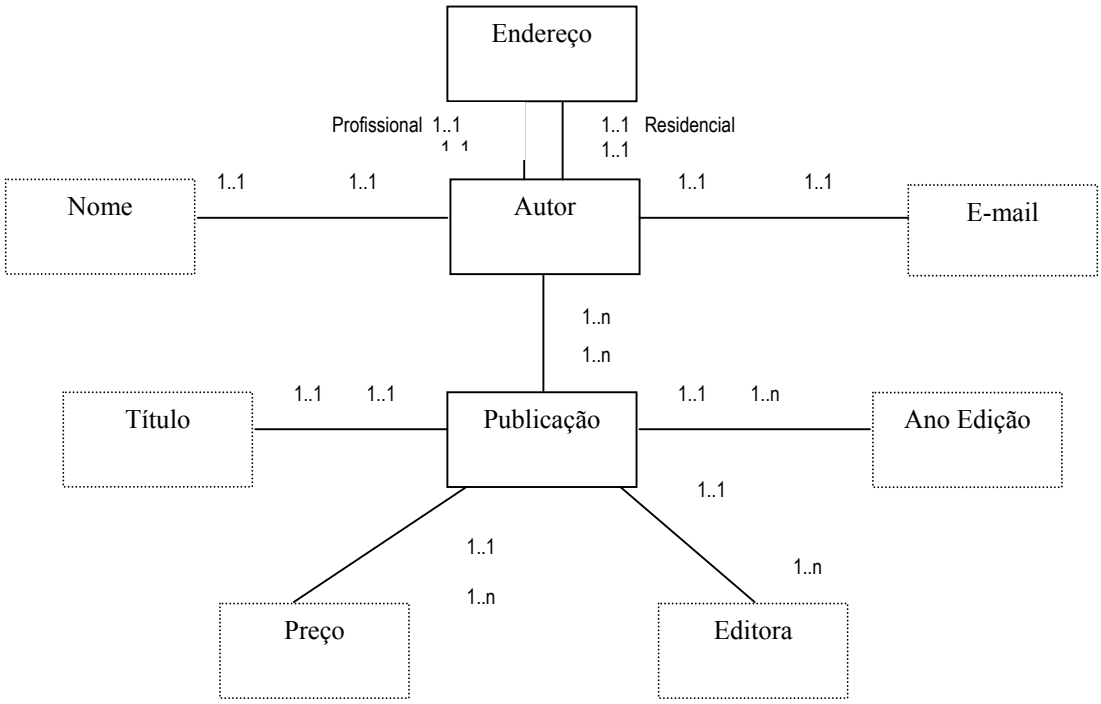


FIGURA 3.2 – Exemplo de uma Ontologia – Domínio de Publicações

3.2 Relação Documentos x Ontologia

A diversidade de formatos para documentos com dados semi-estruturados obriga a focar apenas um tipo, a fim de estudar melhor suas características. Esse trabalho enfoca especificamente documentos no formato HTML. Documentos HTML são compostos por marcas de formatação (*tags)* que delimitam regiões do texto, dando indicações sobre sua estrutura Além dessa facilidade, é um tipo de documento amplamente difundido na *WWW*, o que dispensa maiores comentários.

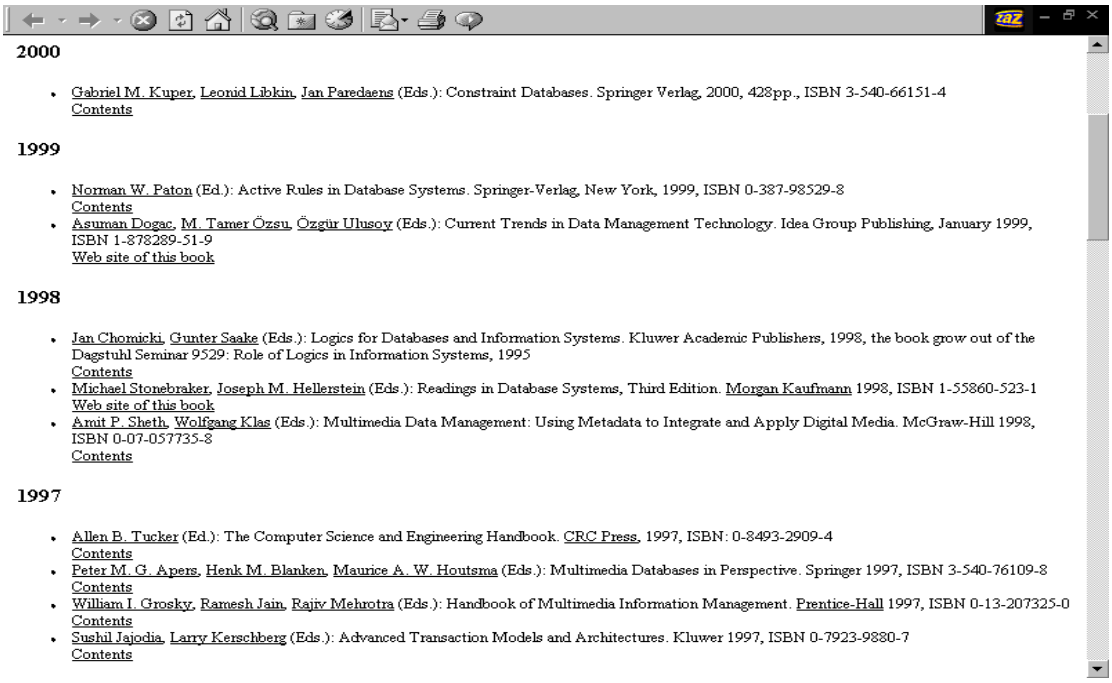
Neste trabalho, os documentos devem ser agrupados em classes, e cada classe de documentos é relacionada a uma determinada ontologia. Uma classe é formada por documentos que possuem uma estrutura semelhante de distribuição das informações, e que possam ser identificados pela mesma ontologia.

Uma classe de documentos deve ser associada à somente uma ontologia, mas uma única ontologia pode ser relacionada a várias classes de documentos. Existem situações em que um único documento pode ser relacionado a várias ontologias. Neste caso, o mesmo documento participa de mais de uma classe.

A ferramenta possui parâmetros que devem ser informados quando uma nova classe é criada, bem como quando ocorrer uma nova associação de documentos à classe. Os principais parâmetros a serem especificados e mantidos são:

- *Nome da Classe:* Especifica o nome da classe de documentos.
- *Descrição da Classe:* Permite detalhar melhor o domínio representado por aquela classe de documentos.
- *Ontologia Associada:* Nome do arquivo que contém a representação gráfica da ontologia que está associada à classe.
- *Documentos Exemplos:* São os nomes dos arquivos que contêm os documentos usados para exemplificar o processo de extração. Conforme a diversidade dos documentos da classe pode ter um ou mais documentos usados como exemplo.
- *Documentos da Classe:* Neste item, ficam associados o nome de todos documentos ou endereços de páginas da *WWW (World Wide Web)* que fazem parte desta classe de documentos.
- *Gramática Associada:* Este campo não é preenchido pelo usuário. Ele serve para identificar o nome do arquivo que contém a gramática gerada pela fase de modelagem do primeiro exemplo.
- *Programa de Extração:* Este campo não é preenchido pelo usuário. Ele mantém o nome do programa gerado a partir da gramática da montagem do primeiro exemplo – o extrator, o qual fará a extração automática dos demais documentos da classe.

No decorrer deste texto, a classe de documentos e a ontologia utilizadas como exemplo referem-se ao domínio que retrata publicações. A página da *WWW*[5] utilizada como exemplo para essa ontologia está exibida na FIGURA 3.3. A FIGURA 3.4 demonstra uma visão parcial do fonte HTML. Essa página relaciona publicações classificadas pelo ano de edição. Para cada publicação, são citados todos seus autores, o título da publicação, a editora e alguns dados adicionais.



2000

- Gabriel M. Kuper, Leonid Libkin, Jan Paredaens (Eds.): Constraint Databases. Springer Verlag, 2000, 428pp., ISBN 3-540-66151-4
  Contents

1999

- Norman W. Paton (Ed.): Active Rules in Database Systems. Springer-Verlag, New York, 1999, ISBN 0-387-98529-8
  Contents
- Asuman Dogac, M. Tamer Özsu, Özgür Ulusoy (Eds.): Current Trends in Data Management Technology. Idea Group Publishing, January 1999, ISBN 1-878289-51-9
  Web site of this book

1998

- Jan Chomicki, Gunter Saake (Eds.): Logics for Databases and Information Systems. Kluwer Academic Publishers, 1998, the book grow out of the Dagstuhl Seminar 9529: Role of Logics in Information Systems, 1995
  Contents
- Michael Stonebraker, Joseph M. Hellerstein (Eds.): Readings in Database Systems, Third Edition. Morgan Kaufmann 1998, ISBN 1-55860-523-1
  Web site of this book
- Amit P. Sheth, Wolfgang Klas (Eds.): Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media. McGraw-Hill 1998, ISBN 0-07-057735-8
  Contents

1997

- Allen B. Tucker (Ed.): The Computer Science and Engineering Handbook. CRC Press, 1997, ISBN: 0-8493-2909-4
  Contents
- Peter M. G. Apers, Henk M. Blanken, Maurice A. W. Houtsma (Eds.): Multimedia Databases in Perspective. Springer 1997, ISBN 3-540-76109-8
  Contents
- William I. Grosky, Ramesh Jain, Rajiv Mehrotra (Eds.): Handbook of Multimedia Information Management. Prentice-Hall 1997, ISBN 0-13-207325-0
  Contents
- Sushil Jajodia, Larry Kerschberg (Eds.): Advanced Transaction Models and Architectures. Kluwer 1997, ISBN 0-7923-9880-7
  Contents

FIGURA 3.3 Página sobre Publicações - Exemplo

[5] A página extraída é do endereço http://www.informatik.uni-trier.de/~ley/db/books/collections/index.html

```
<html><head><title>Collections</title></head><body bgcolor="#ffffff" text="#000000" link="#000000">
<a href="../../index.html"><img alt="dblp.uni-trier.de" src="../../Logo.gif" border=0 height=60 width=170></a>
<h1>Collections</h1><hr>

<h2>2001</h2>
...
<h2>2000</h2>
<ul>
<li><a href="../../indices/a-tree/k/Kuper:Gabriel_M=.html">Gabriel M. Kuper</a>,
<a href="../../indices/a-tree/l/Libkin:Leonid.html">Leonid Libkin</a>,
<a href="../../indices/a-tree/p/Paredaens:Jan.html">Jan Paredaens</a>  (Eds.):
Constraint Databases.
Springer Verlag, 2000, 428pp., ISBN 3-540-66151-4<br>
<a href="../../conf/cdb/cdb2000.html">Contents</a>
</ul>

<h2>1999</h2>
<ul>
<li><a href="../../indices/a-tree/p/Paton:Norman_W=.html">Norman W. Paton</a>  (Ed.):
Active Rules in Database Systems.
Springer-Verlag, New York, 1999,
ISBN 0-387-98529-8<br>
<a href="patton99.html">Contents</a>
<li><a href="../../indices/a-tree/d/Dogac:Asuman.html">Asuman Dogac</a>,
<a href="../../indices/a-tree/=/=Ouml=zsu:M=_Tamer.html">M. Tamer &Ouml;zsu</a>,
<a href="../../indices/a-tree/u/Ulusoy:=Ouml=zg=uuml=r.html">&Ouml;zg&uuml;r Ulusoy</a>  (Eds.):
Current Trends in Data Management Technology.
Idea Group Publishing, January 1999,
ISBN 1-878289-51-9<br>
<a href="http://www.idea-group.com/dogac.htm">Web site of this book</a>
</ul>
```

FIGURA 3.4 – Parcial do Fonte *HTML* sobre Publicações - Exemplo

3.3 Montagem do Primeiro Exemplo

Observando a arquitetura desenhada na FIGURA 3.1, a primeira etapa do processo é a montagem do primeiro exemplo.

É nessa etapa que o usuário interage com a ferramenta para definir a estrutura da classe de documentos e, conseqüentemente, suas regras de extração. A ontologia guia o processo, oferecendo conhecimento básico para o usuário. O resultado dessa etapa é a gramática contendo todas as regras para os futuros processos de extração automáticos da classe de documentos.

Esta Subseção descreve e conceitua a fase de montagem do primeiro exemplo, não entrando em detalhes sobre a interação com o usuário, que pode ser encontrada na Seção 5 deste trabalho.

3.3.1 Processo de Montagem do Primeiro Exemplo

Como já foi mencionado, o método trabalha sobre documentos HTML. O processo de montagem do primeiro exemplo está fundamentado sobre algumas características básicas deste tipo de documento.

a) As informações estão distribuídas no texto de forma organizada.
b) As informações devem estar representadas por objetos da ontologia associada.
c) As informações estão representadas de forma hierárquica no texto.

d) Este tipo de documento pode ser desenvolvido de maneira manual ou gerado por ferramentas apropriadas. A geração de documentos *HTML* por ferramentas de informática, garantem alguma estrutura a esses documentos, pois um programa sempre segue uma seqüência lógica de construção, não dependendo da livre criatividade do usuário [ADE99].
e) A ferramenta necessita da definição de delimitadores que circundam a informação desejada. Não existe nenhuma obrigação de utilizar as marcas de formatação (*tags)* como delimitadores. A ferramenta aceita qualquer porção do texto.

Partindo do princípio de que as informações estão representadas de forma hierárquica, o usuário decompõe o texto em porções cada vez menores. Obrigatoriamente, deve ser feita uma decomposição da porção mais ampla para a menor unidade de texto. Entretanto, não é necessário seguir uma ordem de direção dentro do mesmo nível de decomposição. A ferramenta controla a ação de decomposição criticando a posição do texto em que a nova porção foi demarcada. Essa posição deve estar contida na porção maior. Porém, nunca permite trabalhar sobre uma porção maior que engloba uma porção menor já montada. Esse controle garante a correção das relações de aninhamento dos objetos.

O processo de montagem do primeiro exemplo pode ser acompanhado na FIGURA 3.5. Os dados são representados por elipses, os processos do usuário por um retângulo de linha simples e os processos da ferramenta por um retângulo de linhas mais largas.

O usuário decompõe o texto e marca cada uma das regiões de interesse, contornando-as. Para cada uma dessas marcações associa um objeto/relacionamento da ontologia.

A partir dessa associação, a ferramenta identifica a cardinalidade existente no relacionamento, o tipo do objeto/relacionamento (léxico ou não léxico) e, caso esta ontologia já tenha sido utilizada em outras montagens, o conhecimento dos tipos de dados que pode assumir. Com tal identificação, a ferramenta disponibiliza ao usuário somente os tipos de dados compatíveis àquela situação. O usuário seleciona o tipo de dado a ser associado à porção do texto e ao objeto/relacionamento marcados. Os tipos de dados utilizados pela ferramenta estão descritos na Subseção 3.3.3.

O usuário pode ou não definir os delimitadores que circundam a informação. Nem sempre se faz necessário definir delimitadores, porque as relações de aninhamento dos objetos, em alguns casos, são suficientes para localizar as informações. Os delimitadores e o aninhamento das informações são o guia no processo de extração automática. Os delimitadores são esclarecidos na Subseção 3.3.4.

Outro fator que auxilia na extração é o casamento dos caracteres encontrados com a combinação de caracteres possível para o tipo de dado que está em questão. O usuário pode restringir mais ainda o escopo da informação informando alguns parâmetros para a informação, conforme pode ser visto na Subseção 3.3.5.

Cada uma das interações com o usuário gera uma produção, ou regra de extração na gramática da classe de documentos. A gramática é a informação de saída da fase de montagem do primeiro exemplo e de entrada para a fase de extração automática. A

gramática utilizada está descrita na Subseção 3.3.6. O usuário não visualiza as regras de extração pela gramática, mas em um formato mais simplificado. As regras são exibidas em uma estrutura tabular que exibe os dados complexos e sua composição até chegar nos dados atômicos, que é exemplificado na Seção 5. A hierarquia é representada por uma indentação nas linhas. Apesar da gramática não ser visualizada pelo usuário, ela é a principal e única referência para o processo automático de extração.

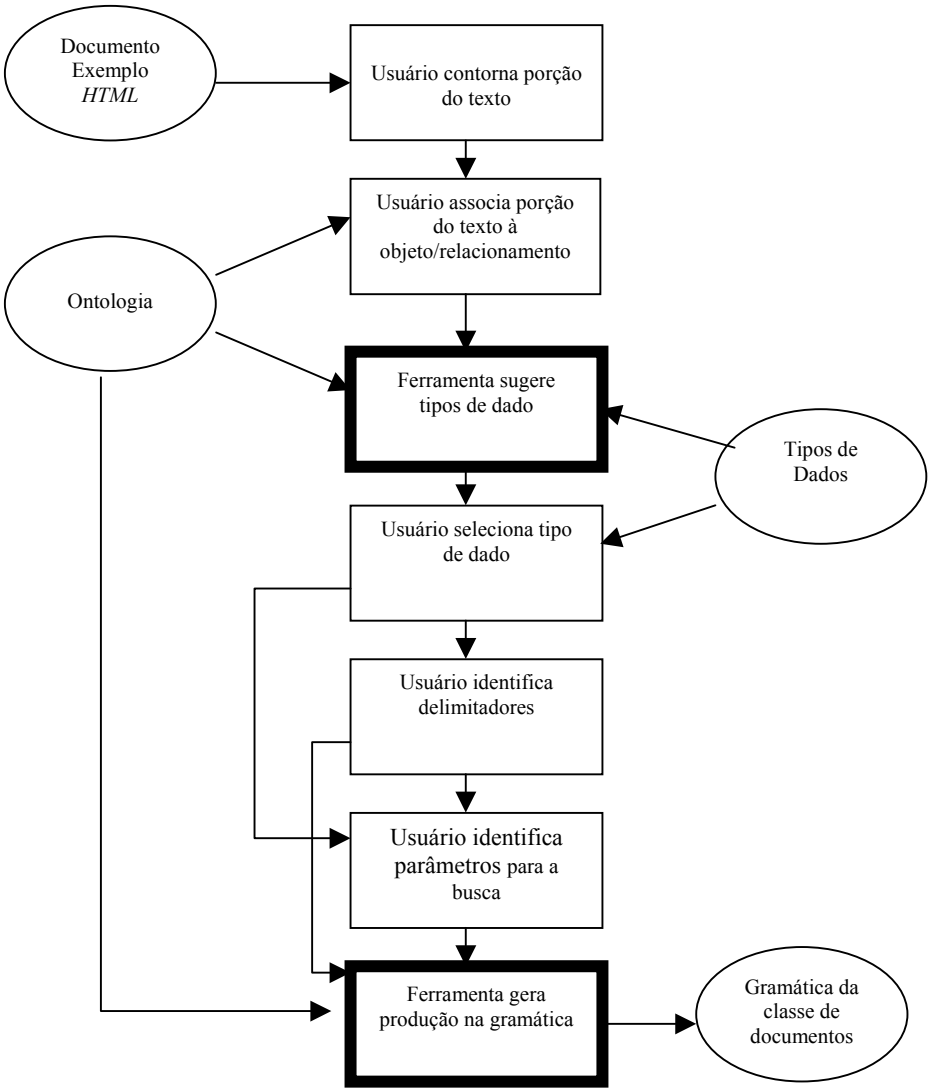


FIGURA 3.5 – Fluxo do Processo de Montagem do Primeiro Exemplo

Utilizando o documento ilustrado nas FIGURAs 3.3 e 3.4, e a ontologia da FIGURA 3.2, pode-se esboçar o seguinte exemplo de montagem das regras de extração:

a) O usuário precisa definir a porção de interesse mais ampla do texto. Nesse caso, a lista “obras”. O objeto que representa a porção mais ampla do texto não está representado na ontologia, pois normalmente refere-se ao domínio como um todo. Como se trata de uma lista, precisa demarcar o primeiro elemento.
b) A lista “obras” é formada por “anoedição”, um objeto atômico, e por outro objeto complexo, a lista “publicação”.
c) “Publicação” é formada por outra lista de “autor” e por dois objetos atômicos “titulo” e “editora”.

d) “Autor” é uma lista formada por objetos atômicos “nome”.

O exemplo de decomposição ilustra apenas uma das possibilidades existentes. O importante é controlar somente a regra de decomposição, não importando a ordem dentro do mesmo nível de decomposição.

Abaixo, são exemplificadas algumas seqüências válidas para montagem dos exemplos para o documento sobre publicações:

a) O objeto “anoedição” está posicionado mais à esquerda de “publicação”, mas pode ser definido depois.

```
1=> obras
      7=> ano de edição
      2=> publicação
            3=> autor
                  4=> nome
            5=> editora
            6=> titulo
```

b) Esta é a seqüência mais próxima do fluxo do documento.

```
1=> obras
      2=> ano de edição
      3=> publicação
            4=> autor
                  5=> nome
            6=> editora
            7=> titulo
```

c) O usuário pode optar em definir por último os componentes de publicação.

```
1=> obras
      3=> ano de edição
      2=> publicação
            4=> autor
                  5=> nome
            6=> editora
            7=> titulo
```

d) O usuário não tem interesse em identificar os autores isoladamente, bem como o título das publicações.

```
1=> obras
      5=> ano de edição
      2=> publicação
            3=> autor
            4=> editora
```

A seguir, são demonstradas algumas seqüências que desrespeitam a regra de decomposição, sendo invalidadas pela ferramenta:

a) “Autor” somente pode ser definido após seu objeto complexo “publicação”.

```
1=> obras
      3=> publicação
            2=> autor
```

b) O usuário somente está interessado nos autores da publicação, desconsiderando título e editora. Para isto precisa definir o objeto "autor" e depois "nome".

1=> obras
2=> ano de edição
3=> publicação
5=> autor
4=> nome

c) "Titulo" somente pode ser definido após o objeto "publicação".

1=> obras
3=> publicação
2=> titulo

Para a ferramenta, basta a exemplificação de apenas um documento pertencente à classe de documentos. Cabe ao usuário analisar se existem documentos pertencentes à classe que possuem alguma regra de extração diferente das já exemplificadas. Caso existam, basta apenas exemplificar essas regras que mudam. O próprio processo de extração automática pode sinalizar ao usuário informações não identificadas na extração, para que as regras de extração sejam aprimoradas.

3.3.2 Influência da Ontologia

Já foi mencionada várias vezes a importância da ontologia na montagem das regras de extração. Neste ponto, faz-se necessário salientar os itens em que a ontologia tem influência direta sobre a montagem das regras:

a) Relaciona todos os objetos possíveis de serem abstraídos daquela classe de documentos.
b) Identifica como os objetos estão relacionados entre si, ou seja, seus relacionamentos.
c) Sugere tipos de dados possíveis de serem assumidos para o objeto/relacionamento, analisando:
   - Tipo do objeto – não léxico e léxico.
   - Restrições de cardinalidade dos relacionamentos entre os objetos.
   - Tipos de dados já associados ao objeto da ontologia em outros processos de extração. Essa informação pode ser fornecida, pois a ontologia vai armazenando o conhecimento aprendido em processos anteriores.
d) Todos os objetos identificados para extração devem estar modelados na ontologia, mas nem todos os objetos da ontologia são utilizados naquela classe de documentos.
e) A ontologia, através da cardinalidade de seus relacionamentos, obriga que o usuário monte o exemplo utilizando sempre os objetos não léxicos antes dos objetos léxicos relacionados a ele. Em outras palavras, seguir o processo de decomposição.

3.3.3 Tipos de Dados

A ferramenta somente disponibiliza os tipos de dados de acordo com as restrições impostas pela ontologia, como já foi citado anteriormente. Conforme o tipo de dado selecionado, a ferramenta realiza uma validação: a informação localizada deve combinar todos seus caracteres com a seqüência parametrizada pelo tipo.

Os dados são classificados em dois grandes grupos: complexos ou atômicos.

TABELA 3.1 – Resumo dos Tipos de Dados

Tipos de Dados				
	Tipo	Parâmetros Possíveis	Delimitadores	Específicos do tipo
COMPLEXO	Lista		1. Esquerda 2. Direita 3. Separador de Elementos	1.Vazia/Obrigatória 2. Primeiro elemento
	Tupla		1. Esquerda 2. Direita 3. Separador de Elementos	1. Ordenada 2. Desordenada 2.1 Seqüências Possíveis
ATÔMICO	Palavra	1.Busca Thesaurus 2. Tamanho Mínimo 3. Tamanho Máximo	1. Esquerda 2. Direita	
	Nome Próprio	1. Busca Thesaurus 2. Tamanho Mínimo 3. Tamanho Máximo	1. Esquerda 2. Direita	
	Conjunto Palavra	1. Busca Thesaurus 2. Tamanho Mínimo 3. Tamanho Máximo	1. Esquerda 2. Direita	
	Conjunto Nome Próprio	1. Busca Thesaurus 2. Tamanho Mínimo 3. Tamanho Máximo	1. Esquerda 2. Direita	
	Inteiro	1. Tamanho Mínimo 2. Tamanho Máximo 3. Valor Mínimo 4. Valor Máximo	1. Esquerda 2. Direita	1. Separador milhar
	Decimal	1. Tamanho Mínimo 2. Tamanho Máximo 3. Valor Mínimo 4. Valor Máximo	1. Esquerda 2. Direita	1. Separador milhar 2. Separador decimal

Cada um deles contém vários tipos de dados que estão relacionados na segunda coluna da TABELA 3.1.

A característica principal dos dados do tipo complexo é o fato de ser composto por outros dados que podem ser também complexos ou atômicos. É através desse tipo de dado que se torna possível a representação das relações de aninhamento dos dados e sua estrutura hierárquica. Os tipos de dados complexos são associados aos objetos do tipo não léxicos da ontologia, como pode ser visto na TABELA 3.2.

Os tipos complexos são divididos em listas e tuplas.

a) Lista

É um conjunto de elementos do mesmo tipo. Cada elemento pode ser complexo ou atômico, mas obrigatoriamente devem ser do mesmo tipo. Nessa opção, o usuário deve parametrizar se a lista pode ser vazia ou se precisa ter obrigatoriamente um elemento. Em seguida, o usuário pode informar quais são os delimitadores da lista (à esquerda e à direita) e o delimitador de separação dos elementos. É obrigatório que o usuário demonstre para a ferramenta o primeiro elemento da lista.

b) Tupla

Uma tupla pode conter elementos de tipos diferentes. Uma tupla pode ser ordenada – quando os elementos estão dispostos sempre na mesma ordem, ou desordenada. O

usuário precisa identificar se a tupla é ordenada ou desordenada. Na desordenada, deve exemplificar as possíveis variações que podem ocorrer.

Os tipos atômicos são associados diretamente aos objetos léxicos da ontologia (TABELA 3.2). Para todos os tipos atômicos a ferramenta solicita ao usuário os delimitadores à esquerda e à direita, não sendo obrigatório, porém, o seu preenchimento.

TABELA 3.2 – Associação Objetos Ontologia x Tipos de Dados x Gramática

Objetos da Ontologia	**Tipo de Dado Montado**	**Gramática**
Não Léxico	Complexo	Não-terminal
Léxico	Atômico	Terminal

Os tipos de dados atômicos disponibilizados pela ferramenta são (TABELA 3.1):

a) *Palavras:* No tipo "palavras", a ferramenta assume um único conjunto de caracteres alfabéticos (maiúsculos ou minúsculos), numéricos e alguns caracteres especiais.

b) *Nomes Próprios:* Neste tipo, a ferramenta obriga que a primeira letra da palavra seja maiúscula, não importando as demais. Como se trata de nomes próprios, aceita letras maiúsculas, minúsculas, o ponto para nomes abreviados e alguns caracteres especiais.

c) *Conjunto de palavras:* Significa que existe um conjunto de palavras (item a) e que estas estão separadas por espaços em branco.

d) *Conjunto de Nomes Próprios:* Significa que é um conjunto de nomes próprios (item b), podendo ter espaço em branco entre eles.

e) *Inteiro:* O tipo de dado é numérico, podendo assumir caracteres de "0" a "9". O usuário especifica se existe separador de milhar e se este separador é um ponto ou uma vírgula.

f) *Decimal:* O tipo de dado é numérico, assumindo caracteres de "0" a "9". Devem ser identificados o caracter separador da parte inteira da decimal (vírgula ou ponto) e o caracter separador de milhar (ponto ou vírgula).

Os tipos de dados atômicos identificam a seqüência de caracteres válidos para a informação em questão, e são transformadas em expressões regulares válidas na gramática.

A ferramenta pode disponibilizar a criação de novos tipos para dados atômicos. Fica a critério do usuário identificar a seqüência de caracteres válidos para aquele tipo. Por exemplo, o usuário pode criar o tipo de dado "data", identificando que os caracteres a serem aceitos são números de "0" a "9" e o sinal de barra ("/"), montando o seguinte formato "99/99/9999".

Utilizando o documento da FIGURA 3.4 associado aos objetos da ontologia da FIGURA 3.2, pode-se exemplificar alguns tipos de dados para os objetos:

- “publicação”: Tipo complexo “lista”, formado pelos elementos “Autor”, “Titulo” e “Editora”.
- “anoedição”: Tipo atômico “inteiro”.
- “autor”: Tipo complexo “lista”, formado pelos elementos “Nome”.
- “editora”: Tipo atômico “nomes próprios”.

Para cada tipo de dado montado, o usuário precisa especificar se é um dado obrigatório ou não. Essa informação é importantíssima para o processo de extração.

A TABELA 3.3 demonstra as definições de tipos de dados que podem ser feitas para o documento da FIGURA 3.3 associado à ontologia da FIGURA 3.2.

A primeira coluna da TABELA identifica o objeto relacionado, e a segunda coluna, o tipo de dado definido para o objeto. Excluindo o cabeçalho e o rodapé da página, o texto todo foi parametrizado como um tipo complexo, ou seja, uma lista chamada “obras”. Cada elemento dessa lista é composto pelo “anoedição” (tipo atômico) e por uma lista chamada “publicação” (tipo complexo). Cada elemento da lista “publicação” é formado por uma outra lista chamada “autor” e por dois tipos atômicos “título” e “editora”. A lista “autor”, por sua vez, é formada por elementos atômicos do tipo “nome” e possui um delimitador intermediário entre os elementos “,” (uma vírgula). A TABELA 3.3 permite visualizar além das regras montadas, o aninhamento das informações.

TABELA 3.3 – Exemplo- Regras Construídas na Montagem do Primeiro Exemplo

Nome do Dado (ontologia)	Tipo do Dado	Parâmetros	Delimitadores		
			Entre elementos	Esquerda	Direita
Obras	Lista	Não Vazia		<hr>	<hr>
Ano Edição	Inteiro	Tamanho=9999 Valor= 1500 a 2999		<h2>	</h2>
Publicação	Lista	Não Vazia			* *
Autor	Lista	Não Vazia	,	*	(Eds.): ou (Ed.): ou (Hrsg.):
Nome	Cjto.Nomes Próprios	Tamanho= 1 a 999 Valida (arq.nomes)		<a* href=*>	
Título	Cjto.Nomes Próprios	Tamanho=1 a 999			.
Editora	Cjto.Nomes Próprios	Tamanho= 1 a 999			, ou .

3.3.4 Delimitadores

O processo de extração utiliza as relações de aninhamento para encontrar a informação desejada. A composição dos dados nem sempre é suficiente para identificar a localização exata da informação. Dessa forma, o método utiliza o conceito de delimitadores.

Os delimitadores são porções do texto, formados por um conjunto de caracteres que circundam a informação desejada. Podem existir delimitadores à esquerda e/ou à direita. Nos tipos complexos, ainda pode ser definido o separador dos elementos de uma lista ou uma tupla.

Durante o processo de extração é combinada exatamente a seqüência de caracteres impostas pelo delimitador contra o documento. É possível utilizar o símbolo coringa “*” (asterisco) que permite a combinação de qualquer seqüência de caracteres. Por exemplo, o delimitador “*<ul>” combina qualquer cadeia de caracteres até encontrar a seqüência exata “<ul>”.

É importante que o usuário utilize critérios rigorosos na definição dos delimitadores. O processo de extração respeita a estrutura hierárquica do documento, mas mesmo assim, pode ocorrer confusão quando são utilizados delimitadores iguais para informações diferentes dentro do mesmo nível hierárquico do documento.

A coluna quatro da TABELA 3.1 relaciona, para cada tipo de dado, a possibilidade de definição de delimitadores: à esquerda, à direita, ou separador de elementos quando tipos complexos.

Seguindo os exemplos do texto (FIGURAs 3.2 e 3.3), o usuário pode definir os seguintes delimitadores para os dados. Os delimitadores à esquerda são visualizados na quinta coluna da TABELA 3.3, e os delimitadores à direita, na sexta coluna da mesma TABELA. A seguir, exemplos de delimitadores possíveis para os objetos:

- “obras”: É uma lista que possui como delimitador à esquerda e à direita os caracteres “<hr>”.
- “anoedição”: Possui como delimitador à esquerda “<h2>” e à direita “</h2>”.
- “publicação”: É uma lista com delimitador à esquerda “
” e à direita “*”. O símbolo asterisco (“*”) identifica que existem outros caracteres antes do delimitador propriamente dito.
- “autor”: É uma lista com delimitador à esquerda “*<li>” e à direita vários delimitadores alternativos: “ (Eds.):” ou “ (Ed.):” ou “ (Hrsg.):”. Possui um delimitador separador dos elementos da lista que é “,”.
- “nome”: Possui como delimitador à esquerda “,a*href=*>” e à direita “”.
- “título”: Somente possui delimitador à direita “.” , pois está localizado logo após o delimitador à direita de “Autor”.
- “editora”: Somente possui delimitador à direita “.” ou “,” pois está localizado logo após o delimitador à direita de “Titulo”.

3.3.5 Parâmetros

A ferramenta já possui a identificação de onde encontrar a informação através das relações de aninhamento e dos delimitadores. Ainda pode combinar que a seqüência de caracteres pertencentes à informação combine com o conjunto determinado pelo tipo do dado. Para aumentar ainda mais a consistência e a correção da informação, pode-se estabelecer alguns parâmetros de validação da informação.

Os parâmetros somente podem ser informados para os tipos de dados atômicos. A seguir, os parâmetros disponibilizados pela ferramenta:

a) Tamanho mínimo da informação em quantidade de caracteres.

b) Tamanho máximo da informação em quantidade de caracteres.

c) Valor mínimo que a informação pode assumir para ser válida. Somente válido para tipos atômicos “inteiros” e “decimal”.

d) Valor máximo que a informação pode assumir para ser válida. Somente válido para tipos atômicos “inteiro” e “decimal”.

e) Validação da informação em um determinado *thesaurus*, ou seja, para a informação ser válida deve existir no arquivo citado. Este parâmetro somente é válido para os tipos atômicos “palavras”, “nomes próprios” e seus conjuntos. É um parâmetro muito interessante para nomes de lugares como cidades e países.

Na TABELA 3.1, coluna três, estão resumidos os parâmetros possíveis para cada tipo de dado disponibilizado pela ferramenta.

Tomando como exemplo o documento da FIGURA 3.4, associado à ontologia da FIGURA 3.2, pode-se criar os parâmetros necessários para cada objeto atômico. Os parâmetros necessários ao processo de extração estão relacionados na coluna três da TABELA 3.3:

- “anoedição”: Como conFIGURA uma informação de ano, basta possuir quatro dígitos numéricos. Dessa forma, para criar mais restrições, é possível parametrizar os “tamanhos mínimo” e “máximo” da informação em quatro (4). Os valores mínimo e máximo podem variar de “1900” a “2999”.
- “Nome”: Os nomes devem pertencer a um *thesaurus,* ou seja, devem estar cadastrados em um arquivo de nomes válidos previamente montado. No exemplo, os nomes devem estar presentes no *thesaurus* “arq_nomes”.
- “Título”: Por se tratar de um nome próprio, pode-se obrigar que possua pelo menos uma letra, parametrizando o tamanho mínimo em um caracter e o tamanho máximo sem limites.
- “Editora”: Pode ser parametrizado da mesma forma que “título”.

3.3.6 Gramática

O processo de montagem do primeiro exemplo deve gerar as regras de extração com conhecimento suficiente para que o extrator possa buscar e extrair as informações de forma correta e automática. As regras devem possuir informações a respeito das relações de aninhamento dos objetos, dos tipos de dados escolhidos, dos delimitadores das informações e dos parâmetros definidos.

Neste trabalho, as regras de extração são representadas em uma gramática da classe de documentos. Está sendo utilizado o conceito de gramática livre de contexto ou BNF (*Backus-Naur Form*) [PRI2000, AHO95, MEN2000]. Outros projetos utilizam gramáticas para representar seus documentos, como pode ser visto em [CRE98, HUC98].

As gramáticas foram escolhidas pelo poder de expressão que possuem. Expressões regulares foram analisadas em um primeiro momento, mas elas não são suficientes para representar o aninhamento das informações do documento – tipos complexos [AHO95]. As gramáticas possibilitam representar estruturas complexas e oferecem possibilidade de ter regras alternativas para uma única informação. Além disso, tem relação direta com o formato das *DTD* utilizadas para gerar o resultado em *XML*.

Alguns problemas encontrados no processamento de gramáticas de livre contexto, como a recursividade e ambigüidade[AHO95], precisam ser tratados pela ferramenta. Tais características não podem ser eliminadas, pois são essenciais para a representação de dados complexos e para condições alternativas. Dessa forma, é preciso tratá-las para que não causem erros de processamento durante o processo de extração. Sempre que encontra uma produção ambígua, a ferramenta cria uma segunda produção para representar a ambigüidade. A recursividade foi tratada da mesma forma, pois, com tipos complexos, ela se faz necessária. Uma segunda produção foi criada invocando novamente a produção original. Esses processos podem ser melhor entendidos nos exemplos.

Na FIGURA 3.6 está definida a gramática genérica utilizada pela ferramenta. Alguns símbolos básicos são utilizados pela gramática e precisam ser esclarecidos:

- ":=" – Símbolo de atribuição. Separa o nome das regras da produção.
- "{}" – O par de chaves indica que é um termo não-terminal, ou seja, que existe uma nova produção para o termo. Os não-terminais estão associados aos tipos de dados complexos ou objetos não léxicos da ontologia, conforme resumo da TABELA 3.2.
- "|" – Identifica que existe uma regra alternativa para a mesma produção, ou seja, a produção pode ser validada por uma ou outra regra.
- "&" – Sentido de vazio. Indica que nenhuma regra precisa ser validada. Normalmente está associada ao símbolo de alternação ("|") para fazer sentido. O vazio sempre é utilizado quando um tipo de dado for especificado para uma informação não obrigatória.
- " []" – O par de colchetes encerra uma expressão regular que identifica a combinação de caracteres que a informação pode assumir. Ela deriva diretamente do tipo de dado escolhido pelo usuário. Se a escolha for tipo de dado palavra, a expressão regular derivada é [A-Za-z0-9-/:;]. Na TABELA 3.4 estão descritas as expressões regulares associadas a cada tipo de dado oferecido pela ferramenta.
- " ( )" – O par de parênteses contém os parâmetros que auxiliam na extração das informações. São eles: tamanho mínimo, tamanho máximo, valor mínimo, valor máximo e busca em *thesaurus*.
- "*" – Símbolo utilizado nos delimitadores como um coringa. Significa que não importa o caracter a ser combinado.

A gramática é formada por uma lista de declarações ou produções. Cada produção é uma linha da gramática composta por duas partes básicas: o nome da produção ou do objeto que está antes do sinal de atribuição (":=") , e as regras da produção. As regras podem ser formadas por termos não-terminais e terminais.

- Não-terminais – São termos que apenas invocam outra produção. Sempre estão contidos em um par de chaves ("{}"). Estão diretamente associados aos tipos de dados complexos ou objetos não léxicos. A TABELA 3.2 deixa bem claro o relacionamento dos dados relacionados aos objetos da ontologia com os tipos de dados da ferramenta e com os termos da gramática.
- Terminais – São termos que encerram na produção. São os tipos atômicos ou léxicos da ontologia (TABELA 3.2). Os terminais são subdivididos em dois grupos:
  - ✓ Delimitador – São os delimitadores que circundam a informação.
  - ✓ Informação – Engloba as combinações de caracteres possíveis que a informação pode assumir, conforme o tipo de dado estabelecido. Este termo terminal pode ser acompanhado de parâmetros.

```
gramatica := {lista_declaracoes}
lista_declaracoes := {nome-objeto} := {producao}{ llista_declaracoes1}
llista_declaracoes1 := {lista_declaracoes} | &
producao:= {não_terminal} {lproducao1} | {terminal} {lproducao1}
lproducao1 := NEWLINE | {producao}| | {producao}| | & | &
não_terminal := {    {nome-objeto}    }
terminal := {delimitador}  | {informação}
informação:= [ {combinacoes} ] parametros {informação1}
informação1:= informação | &
parametros:= (tammin=   {inteiro}  tammax= {inteiro}) {lparametros1}{lparametros2}
lparametros1:= (valmin= {decimal} valmax= {decimal} )   |  &
lparametros2:= (busca= {nome} ) |            &
Combinacoes := A-Z {lcombinacao1} | a-z {lcombinacao1} | 0-9 {lcombinacao1} | 0-9.0-9
                        {lcombinacao1} |  0-9,0-9 {lcombinacao1} | space {lcombinacao1} ...
lcombinacao1 := combinacoes | &
Numero:= inteiro | decimal
inteiro :=  0 {linteiro1} | 2{linteiro1} |...| 9{linteiro11}
linteiro1 := inteiro | &
decimal :=  0 {ldecimal1} | 1{ldecimal1} |...| 9{ldecimal1}
ldecimal1 := decimal | , {decimal} | &
nome-objeto := A {lnome-nome-objeto1} |B {lnome-objeto1}| ....| Z {lnome-objeto1}
lnome-objeto1 := {nome-objeto} | {nome-objeto} {inteiro} | &
delimitador:= @ {ldelimitador1} | $ {ldelimitador1} | < {ldelimitador1} |
                        ...   (todos caracteres possíveis como > ,  * ,  # ,  % ,   ( ,  ) ,  - , _ ,  + , = , ´ , { , }
                        , [ , ] , ? , & ,  ,  , . , > , . , : , ;  , / , \ , & , :  , A  ...Z , a ... z, 0 ...9

ldelimitador1 := {delimitador} | &
```

FIGURA 3.6 – Gramática Genérica

A gramática genérica, representada na FIGURA 3.6, é formada por uma lista de declarações ou produções. Cada elemento da lista de declarações é formado por um nome da produção, o sinal de atribuição e as regras da produção. Uma produção é uma tupla formada pelo conjunto de termos terminais e não-terminais. Um não-terminal é composto por um par de chaves que contém o nome de outra produção ou objeto. Um terminal pode ser um delimitador de uma informação. Um delimitador possui os caracteres que indicam a proximidade da informação. A informação é formada por um par de colchetes que possui as combinações de caracteres possíveis que a informação pode assumir, e pode ser seguido ou não de parâmetros. Parâmetros, por sua vez, é um par de parênteses que contém os parâmetros que restringem a informação a ser encontrada.

TABELA 3.4 – Correspondência dos Dados Atômicos com Produções da Gramática

Palavra	[A-Za-z0-9-/:;]
Nome Próprio	[A-Z] [A-Za-z.]
Conjunto Palavra	[A-Za-z0-9-/:;space]
Conjunto Nome Próprio	[A-Z] [A-Za-z.space]
Inteiro	[0..9]
Decimal	[0-9,0-9] ou [0-9.0-9] conforme opção

As produções recebem como nome o mesmo utilizado nos objetos da ontologia. Entretanto, algumas produções são geradas pela ferramenta para criar o conceito de dados complexos e para tratar os problemas de recursividade e ambigüidade. Para essas

produções adicionais é utilizado o mesmo nome da produção original acrescentando-se a letra “l” em seu início e um número seqüencial crescente no final. Por exemplo: como a gramática é uma lista de declarações – “lista_declarações”, seria preciso que ela invocasse a si mesma, o que causaria problemas de recursividade. Dessa forma, é criada uma produção adicional – “llista_declarações1”. A produção original invoca em seu final a produção adicional “llista_declarações1”, e esta invoca a produção original “lista_declarações” ou uma combinação vazia. Esta alternação de poder ser uma combinação vazia é necessária para validar o último elemento da lista.

Para cada tipo de dado criado pelo usuário, na fase de montagem do primeiro exemplo, é gerada uma produção na gramática. Seguindo o exemplo, ao longo deste texto, pode-se observar na FIGURA 3.7 a gramática gerada pela ontologia de obras publicadas (FIGURA 3.2).

A geração da gramática ilustrada na FIGURA 3.7 segue algumas regras que são destacadas e exemplificadas:

a) O símbolo de partida da gramática corresponde à porção mais ampla do texto e é a única produção que não está associada diretamente a um objeto da ontologia. O usuário dá um nome a esta produção, que pode ser o domínio da aplicação. No exemplo do texto, utiliza-se como símbolo de partida o nome “obras”.

b) Os objetos léxicos (atômicos) da ontologia passam a ser na gramática os terminais. Como o usuário determina os tipos de caracteres possíveis para a informação a ser extraída, essas combinações são representadas na gramática através de expressões regulares como as usadas na linguagem PERL [SCH97] e fechadas entre um par de colchetes “ [ ]”. Na gramática gerada do exemplo, os termos terminais são “anoedição”, “nome”, “titulo” e “editora”.

c) Os delimitadores passam a ser na gramática também terminais. Eles sempre acompanham um termo terminal, que representa a informação ou um termo não-terminal. Na gramática gerada, são muitos os delimitadores: “<ul>”, “*<li>*”. O asterisco “*” é empregado para combinar qualquer caracter até encontrar a seqüência de caracteres indicadas no delimitador.

d) Os conceitos não léxicos (complexos) passam a ser na gramática os não-terminais e são identificados por um par de chaves “{}”. Existem não-terminais de dois tipos: os relacionados aos objetos não léxicos da ontologia e os criados pela ferramenta na criação da representação de tipos complexos e no tratamento de recursividade e ambigüidade. Os não-terminais associados a objetos que podem ser citados do exemplo: publicação e autor.

e) Cada objeto da ontologia que possui regras de extração, transforma-se em uma produção na gramática. Para cada regra de extração alternativa do objeto, é criada uma regra alternativa na produção. Dessa forma, o mesmo objeto pode ter várias regras de produções a serem combinadas alternativamente. A possibilidade de combinar uma ou outra regra da produção é identificada pelo caracter “|” (pipe). Este conceito fica claro observando a produção “lautor2”.

f) Para os objetos não léxicos (listas e tuplas), são geradas, no mínimo, duas produções adicionais. A produção original que leva o nome do objeto na ontologia

identifica o tipo do dado (lista ou tupla) e seus delimitadores (à esquerda e à direita), como é o caso da produção "publicação". A primeira produção adicional "lpublicação" identifica a composição de um elemento da lista e invoca uma segunda produção adicional "lpublicação1" responsável por dar continuidade à lista. Essa segunda produção adicional existe para tratar a recursividade necessária para encontrar os demais elementos daquele dado complexo. O tratamento de recursividade utilizado pela gramática não permite que ocorram interações infinitas. Os nomes das produções adicionais levam o nome da original acrescentando a letra "l" no início e um número seqüencial crescente no final. São exemplos dessa situação: "lobras" e "lobras1", "lautor", "lautor1" e "lautor2",

g) A ferramenta cria produções extras para eliminar o problema de ambigüidade existente nas gramáticas. Caso contrário, o extrator teria duas possibilidades de busca que iniciam com o mesmo conjunto de caracteres. Sempre que uma produção possui produções alternativas, e o início delas é exatamente igual, a ferramenta mantém na produção original a parte igual e cria uma segunda do tipo não-terminal. Esta não-terminal passa a ser uma nova produção que contém somente as partes que eram diferentes e alternativas na produção original. O nome da produção sempre será o nome da produção original acrescida de um número seqüencial crescente no final. Exemplos desta situação são "titulo1" e "editora1".

h) Para tratar exceções na gramática, a ferramenta disponibiliza ao usuário a possibilidade de informar se o dado é obrigatório ou não, ou seja, se a produção obrigatoriamente deve ser validada. Quando o dado não é obrigatório, a produção é criada com a possibilidade de não ser validada, representada pelo símbolo de vazio – "&".

```
obras:=          *<hr>* {lobras} *<hr>*
lobras:=         {anoedicao} {publicacao} {lobras1}
lobras1:=        {lobras} | &

anoedicao:=      <h2> [0-9]  (tammin=4 tammax=4) (valmin=1500 valmax=2999) </h2>

publicacao:=     <ul> {lpublicacao} *</ul>*
lpublicacao:=    {autor} {titulo} {editora} {lpublicacao1}
lpublicacao1:=   {lpublicacao} | &

autor:=          *<li>* {lautor} {lautor2}
lautor:=         {nome} {lautor1}
lautor1:=        , {lautor} | &
lautor2:=        (Eds.): |  (Ed.): |  (Hrsg.): | &

nome:=           *<a*href*>* [A-Z]  (tammin=1 tammax=1) [A-Za-z.]  (tammin=1 tammax=999) </a>

titulo:=         [A-Z]  (tammin=1 tammax=1) [A-Za-z0-9]  (tammin=1 tammax=999) {titulo1}
titulo1:=        .

editora:=        [A-Z]  (tammin=1 tammax=1) [A-Za-z0-9.]  (tammin=1 tammax=999) {editora1} | &
editora1:=       , | . | &
```

FIGURA 3.7 – Gramática Exemplo Gerada na Montagem do Primeiro Exemplo

3.4 Extração Automática

A gramática gerada pela etapa de montagem do primeiro exemplo (FIGURA 3.1), contém todas as regras de extração que espelham a estrutura do documento e das informações. Essas regras de extração possuem conhecimento sobre as relações de aninhamento dos dados, a combinação possível de caracteres para cada informação definida pelo tipo de dado escolhido, os delimitadores que circundam a informação, funcionando como filtros, e os parâmetros de restrição estabelecidos.

Para os demais documentos pertencentes à mesma classe, o processo de extração passa a ser desenvolvido de forma automática. A ferramenta somente solicita a intervenção do usuário quando ocorrem falhas no processamento. O exemplo de uma falha que exige a intervenção é quando uma regra de extração obrigatória não for validada.

Esse processo assemelha-se muito aos algoritmos utilizados pelas ferramentas que geram programas compiladores para qualquer linguagem. A diferença básica é que essas ferramentas possuem fases bem distintas que separam a análise léxica da sintática, e esta da semântica. Já em extração de documentos as fases de análise léxica e sintática se fundem em uma só, e a análise semântica não existe [HUC98]. A união das duas fases é necessária, porque a interpretação dos dados e de seus delimitadores dependem de seu contexto sintático. Os processos são semelhantes, principalmente, no fato de gerarem um programa fonte a partir de uma dada gramática, que depois é executado. Nesse projeto, o programa gerado é o próprio extrator da classe de documentos.

O processo de extração automático pode ser dividido em duas etapas principais:

- Geração do programa extrator.
- Execução da extração.

3.4.1 Geração do Programa Extrator

A gramática representa diretamente a estrutura hierárquica do documento. Dessa forma, as produções da gramática iniciam indicando porções mais amplas do documento e vão sendo detalhadas até chegar na informação propriamente dita.

Partindo deste raciocínio, o algoritmo avança sua análise da gramática de cima para baixo e da esquerda para a direita. O funcionamento da geração do programa extrator pode ser visto em detalhe no algoritmo simplificado da FIGURA 3.8. A geração do programa ainda pode ser dividida em duas partes.

a) Interpretação da gramática

A leitura da gramática é feita de cima para baixo, da esquerda para a direita. Como a gramática representa a estrutura hierárquica do documento, a primeira produção a ser interpretada é a que identifica a porção mais ampla do documento. A primeira produção também é chamada de símbolo de partida. No exemplo utilizado neste texto, a interpretação inicia na produção com o nome “obras”.

A ferramenta utiliza dois tipos de listas internas para representar a estrutura da gramática. A primeira lista é chamada de lista mestra e é composta por nodos que contêm os nomes de todas produções da gramática. No algoritmo (FIGURA 3.8), pode-se observar que é sempre inserida nesta lista a parte esquerda da produção (anterior ao símbolo de atribuição) que é o nome da produção propriamente dita.

Para cada nodo desta lista mestra é encadeada uma segunda lista que relaciona as regras da produção. Isso significa que para cada produção, é criada uma lista particular. Cada uma das listas particulares faz referência à lista mestra sempre que um não-terminal for solicitado. Seguindo o exemplo utilizado da FIGURA 3.7, temos uma lista mestra onde os nodos terão os seguintes valores: “obras”, “lobras”, “lobras1”, “anoedição”, “publicação”, “lpublicação”, “lpublicação1”, “autor”, “lautor”, “lautor1”, “lautor2”, “nome”, “titulo”, “titulo1”, “editora” e “editora1”. Cada um desses nodos possui uma referência para sua lista particular, relacionando a parte direita da produção também totalmente decomposta em termos.

Na decomposição da parte direita da produção, são identificados os termos e estes são inseridos na lista particular identificando o tipo de termo que está representando. A identificação é essencial durante a execução do programa de extração. Os tipos de termos e nodos que podem ser identificados são os seguintes:

i. *Nodo do tipo não-terminal*
Ocorre sempre que a cadeia de caracteres estiver contida entre um par de chaves (“{ } “). Identifica que é uma referência à outra produção, representando desta forma um tipo de dado complexo, ou seja, composto por outros dados. Tal nodo deve ter um apontamento para a lista mestra, identificando onde se localiza o termo desta produção.

ii. *Nodo do tipo terminal*
Esse tipo de nodo representa os dados do tipo atômico, não possuindo mais decomposição alguma. Este tipo pode ainda ser dividido em subtipos:

- Informação: Este tipo de termo aparece inserido entre um par de colchetes (“[]”). Ele identifica que nesse termo estão presentes as combinações de caracteres possíveis que a informação a ser extraída pode ter, ou seja, os valores que a informação pode assumir.
- Parâmetros: Este tipo de termo somente pode aparecer seguindo um termo do tipo de informação. Identificado por um par de colchetes (“( )”), identifica os parâmetros que devem consistir a informação, restringindo mais as possibilidades e dando maior segurança ao processo de extração. Tais parâmetros já estão explicados na seção anterior.
- Ou: Identificado pelo símbolo”|”, indica que existe uma produção alternativa a ser combinada na seqüência.
- Vazio: Identificado apenas pelo símbolo “&”, indica que a produção não precisa obrigatoriamente ser validada. Sempre aparece depois de um termo de alternação.
- Delimitador: Quando não se encaixar em nenhum dos termos acima, será um termo do tipo delimitador. O termo contém a cadeia de caracteres que indicam a proximidade da informação.

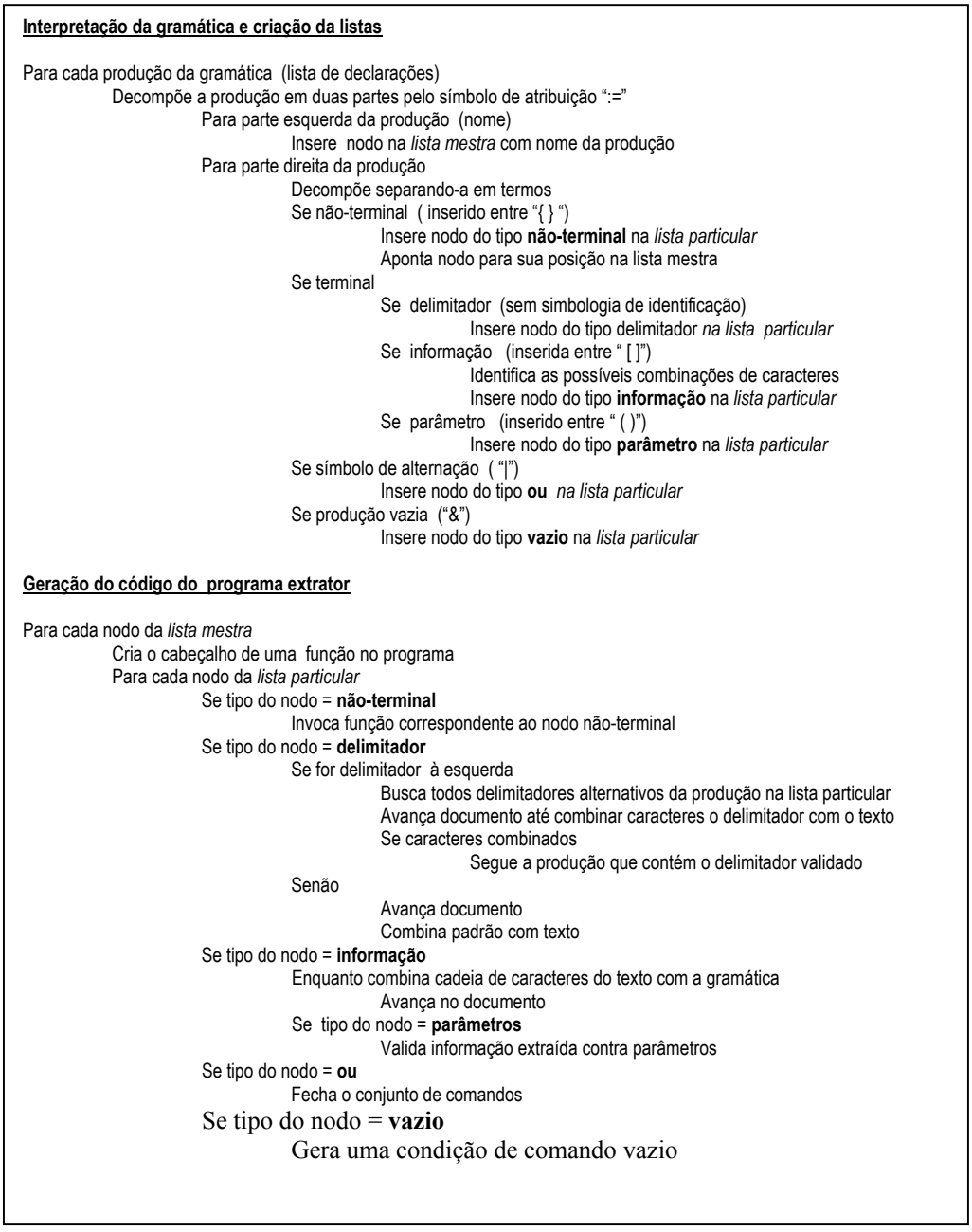


FIGURA 3.8 – Algoritmo Reduzido da Geração do Código Extrator

b) Geração do Código do Extrator

A lista mestra juntamente a suas listas particulares diretamente encadeadas fornecem a estrutura do código do programa extrator a ser gerado.

Seguindo o algoritmo da FIGURA 3.8, pode-se observar que ele possui dois grandes blocos de comando: um bloco que conduz a varredura da lista mestra e outro que conduz a varredura das listas particulares.

Cada nodo da lista mestra transforma-se em uma função no código do programa que pode retornar verdadeiro ou falso (em caso de falha). O padrão é sempre retornar falha da função. A função é forçada a ser válida quando suas instruções obtêm sucesso ou quando é utilizada uma produção do tipo vazio.

Para cada nodo da lista mestra, o algoritmo busca sua lista particular encadeada até chegar no final da mesma. Cada nodo da lista particular está identificado com um tipo, e é esse tipo que determina a ação a ser gerada no código do programa extrator:

i. Nodo do tipo não-terminal

Gera uma chamada à função com o nome desse nodo. Por exemplo, na produção "lobras" (FIGURA 3.7), é gerada uma chamada à função "anoedição". No retorno com sucesso da chamada é gerada uma outra chamada à função "publicação" e, dentro desta, uma nova à "lobras1".

ii. Nodo do tipo terminal

- Informação – Gera comandos de avanço no documento enquanto a cadeia de caracteres for compatível com as combinações propostas pela gramática. A produção "anoedição" (FIGURA 3.7) contém a seqüência " [0-9]" a qual indica que a informação pode conter números de "0" a "9". Desta forma, são geradas instruções para avançar no documento enquanto os caracteres pertencerem a esse grupo de valores.
- Parâmetros – Produz comandos de teste comparando a informação extraída com os parâmetros. Se não for compatível com os parâmetros gera uma instrução de falha no programa. Na mesma produção "anoedição" (FIGURA 3.7), são geradas instruções para validar que o tamanho da informação a ser extraída seja 4 e que o escopo dos valores que a informação pode ter vai de "1500" a "2999".
- Ou – Cria instruções para encerrar o bloco de comandos, pois indica que a produção encerrou e inicia uma nova. Na produção "lautor2" (FIGURA 3.7), todas os termos são do tipo delimitador. Dessa forma, gera instruções de avanço no documento. Cada vez que encontrar um nodo do tipo ou, o bloco de instruções em andamento é encerrado.
- Vazio – Gera uma instrução de validação da função como verdadeira independentemente de alguma produção ser combinada ou não. Na produção "lautor2" (FIGURA 3.7), não é obrigatório combinar algum dos delimitadores, o que permite a utilização de uma produção vazia. Neste caso, é gerada uma instrução que sempre valide como verdadeira a função "lautor2". É importante lembrar que o padrão é sempre devolver a validação como falsa.
- Delimitador - Se for um delimitador à direita da informação, basta avançar no documento enquanto a cadeia de caracteres combinar. Se o delimitador for à esquerda da informação, é preciso validar ao mesmo tempo todos os delimitadores à esquerda daquela produção. O primeiro delimitador a ser validado totalmente contra o texto seleciona a produção que continuará a ser investigada. A produção "nome" (FIGURA 3.7) possui um delimitador de combinação à esquerda e um à direita. Como não existe alternação na produção, somente pode existir um delimitador à esquerda. Dessa forma, o programa gera instruções de avanço no documento até encontrar a seqüência de caracteres "<a". Depois dela, continua avançando até encontrar a seqüência "href", e depois até chegar na seqüência ">". A informação deve estar logo após a última seqüência de caracteres tratada. No exemplo, o avanço é até encontrar determinada seqüência de caracteres porque foi utilizado o símbolo coringa "*". Já na

produção “anoedição”, o avanço deve ocorrer no documento e encontrar de forma direta, sem saltar nenhum caracter, a seqüência “<h2>”.

É importante observar que apenas dois tipos de nodo provocam avanço no documento a ser processado: nodos do tipo informação, que é o valor a ser extraído, e nodos do tipo delimitador, que contêm a cadeia de caracteres a ser validada como filtro da informação.

3.4.2 Execução da Extração

O código fonte do programa de extração está gerado. Basta, agora, executar o programa para extrair as informações relevantes do texto.

O código fonte do programa é um espelho da gramática, e esta é um espelho da representação hierárquica do documento. Dessa forma, pode-se considerar que a estratégia de extração empregada é a *top-down.*

Na estratégia *top-down,* a busca da informação se faz dos não-terminais para os terminais (usando a linguagem de gramáticas), ou dos dados complexos para os atômicos (usando a linguagem dos tipos de dados), ou dos objetos não léxicos para os léxicos (utilizando a linguagem de ontologias). Em resumo, segue a estrutura hierárquica do documento, partindo do todo para o detalhe, decompondo-o em partes cada vez menores. No momento em que extrai a informação atômica, faz um retorno recursivo, recompondo os dados mais complexos sucessivamente até chegar no topo da estrutura hierárquica.

Utilizando o exemplo da gramática da FIGURA 3.7, a extração parte da produção “obras”. O processo explicado pode ser acompanhado na FIGURA 3.9 ilustrado em forma de árvore. Encontra um nodo terminal do tipo delimitador e avança no texto. Segue e encontra o nodo não-terminal “lobras”, que é invocado. Por sua vez, é composto por “anoedição”, “publicação” e “lobras1”. Busca, então, as associações de “anoedição”. “Anoedição” é um nodo não-terminal formado por três nodos terminais. O primeiro é um delimitador que provoca avanço no texto até encontrar a seqüência “<h2>”. O segundo é um nodo do tipo informação que provoca a extração da informação e sua validação contra o tipo de dado e os parâmetros. O terceiro nodo é do tipo delimitador que provoca mais um avanço no texto até combinar “</h2>”. Como não existem mais níveis nesses nodos, retorna para o dado complexo que o chamou, no caso, “lobras”, compondo a informação desse nodo. O programa segue nas demais composições de lobras. Quando todas encerrarem (“publicação” e “lobras1”), o dado complexo lobras está totalmente montado. Então retorna para seu complexo superior que é “obras”. Neste ponto, somente falta tratar o último terminal que é um delimitador e, assim, está também composta a informação “obras”.

Como a gramática utilizada neste exemplo (FIGURA 3.7) possui várias produções alternativas, as informações a serem extraídas e o fluxo da extração em si podem variar conforme o documento trabalhado em dado momento. Dessa forma, o que está representado na FIGURA 3.9 é uma das alternativas possíveis do processo de extração.

É importante salientar que o processo de extração realiza uma série de validações para dar segurança à informação extraída. A maioria das validações é imposta pela gramática: as relações de aninhamento das informações; as combinações possíveis de caracteres que a informação pode assumir, conforme o tipo de dado determinado para ela; os delimitadores que devem circundar a informação, funcionando como filtros; e os parâmetros de restrição estabelecidos.

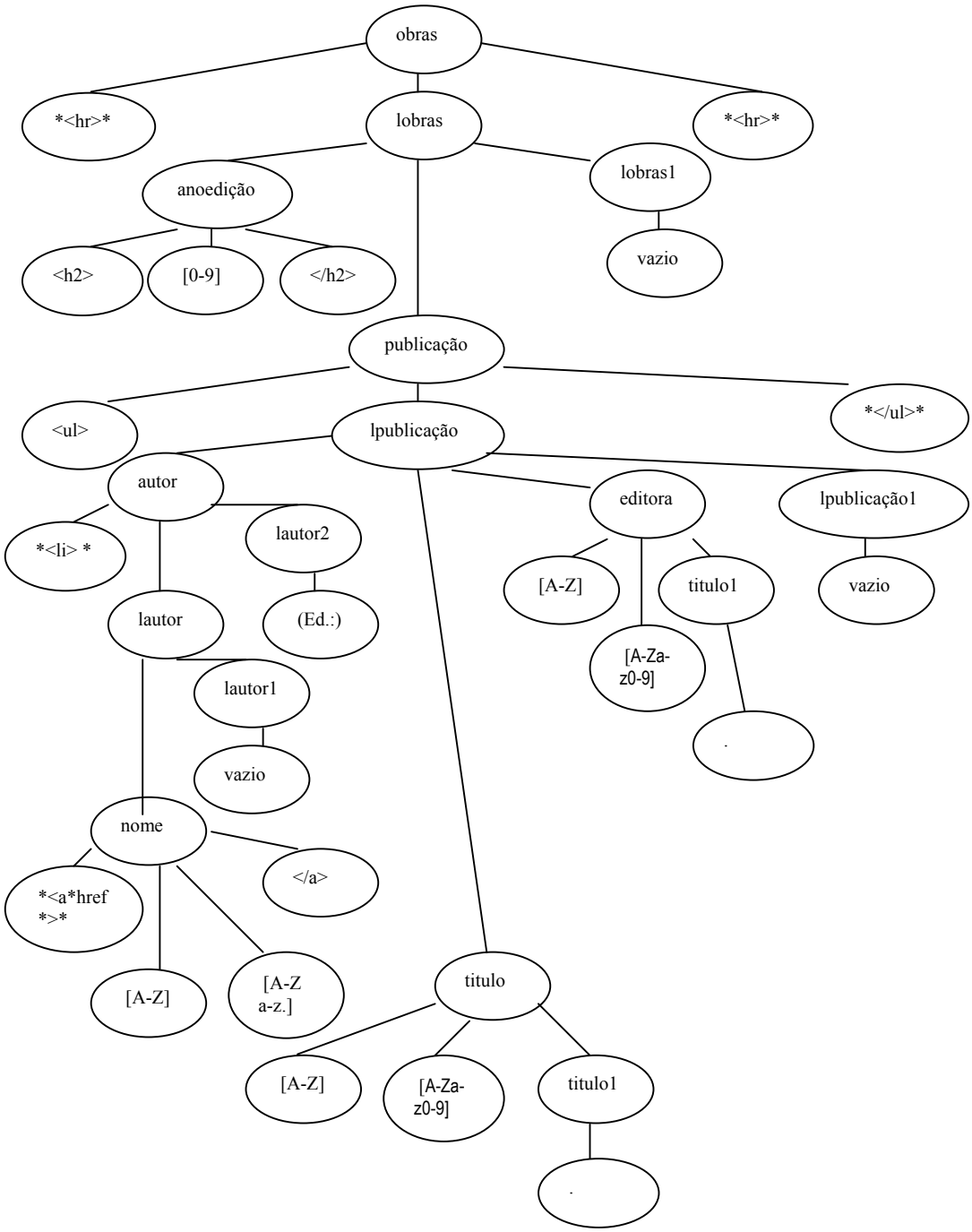


FIGURA 3.9 – Caminho da Extração Automática

Além das validações impostas pela gramática, o processo de extração pode ir agregando novo conhecimento à ontologia. O extrator armazena uma série de resultados obtidos do processo de extração e transforma os resultados em estatísticas. Estas são utilizadas nos próximos processos de extração, o que restringe mais ainda o escopo da informação, dando maior segurança e correção ao processo.

Os dados estatísticos capturados são os seguintes, podendo ser estendidos conforme a evolução do projeto:

- Tamanho médio, mínimo e máximo em que o conceito foi encontrado nos documentos.
- Tipo de dado que representa o conceito. Essa estatística armazena os tipos de dados, ou as combinações possíveis de caracteres que podem ser empregados para representar a informação no documento. Tal estatística é importante também na fase de modelagem do primeiro exemplo, para sugerir o tipo de dado mais adequado àquele conceito da ontologia.
- Quantidade média de vezes em que o documento não foi extraído dos documentos, pois não era informação que fazia parte dele.
- Quantidade média de vezes em que este conceito precedeu um outro conceito nos documentos. Essa estatística é relevante quando as produções possuem alternativas de validação. Usando os termos da gramática, este número nos fornece a quantidade de vezes em que foi validada uma ou outra produção.
- Quantidade média de vezes em que o conceito sucedeu um outro conceito. Neste item vale as mesmas considerações feitas no item anterior.

Tanto as violações das restrições da gramática como as da estatística são notificadas ao usuário. Entretanto, elas exigem ações diferentes.

As violações das restrições da gramática exigem uma tomada de decisão no momento em que ocorreu a violação. O usuário precisa interferir para determinar se ignora o problema encontrado ou se refaz a gramática, alterando ou criando novas produções. Por exemplo, a gramática aguardava por uma lista com pelo menos um elemento, e esta lista não foi encontrada. Provavelmente, a produção da gramática deve passar a aceitar uma lista vazia. Esses problemas ocorrem quando a estrutura do documento não é exatamente igual ao usado como exemplo. Desta forma, o usuário vai enriquecendo a gramática.

As violações das restrições impostas pelas estatísticas armazenadas notificam o usuário com uma mensagem de aviso, sem exigir uma ação imediata. Por exemplo, a estatística mantém que o tamanho máximo que a informação foi encontrada nos documentos é de 50 caracteres e, nesse documento, a mesma é extraída possuindo um tamanho de 200. A ferramenta avisa o usuário de que o tamanho de tal conceito extrapolou a média estatística.

Além das validações de restrições da gramática e da estatística, a ferramenta relaciona para o usuário todas as informações que possuem regras de extração e que não foram encontradas no documento. A ferramenta disponibiliza para o usuário o ambiente de montagem do exemplo para que possa criar ou alterar regras. A gramática é

novamente gerada, ficando disponível para as próximas extrações. Essa interação possibilita o aprimoramento do extrator.

3.5 Geração do Resultado

Com as informações extraídas, é preciso transformá-las em um formato estruturado de dados. Essa fase está sendo chamada de geração do resultado da extração.

O formato a ser utilizado precisa atender a algumas premissas básicas: ser estruturado, flexível, bem formado e aceito como padrão na comunidade de informática. Assim, a informação extraída é disponibilizada em um formato universal, possibilitando ao usuário utilizar outras ferramentas ou linguagens de consulta para manipular as informações.

Considerando tais questões, as informações são estruturadas em formato *XML (eXtensible Markup Language)* [W3C2000, ABI2000, HOL01]. A linguagem *XML* é similar a *HTML*, amplamente difundida na *World Wide Web*. Tem o propósito de facilitar a troca de informações entre diversas aplicações, plataformas e ambientes de rede, o que dá a ela um caráter de padrão universal. Também é amparada por um órgão mundial W3C [W3C2000] que assegura a padronização da linguagem através da boa formação e validade do documento. A boa formação de XML é assegurada pelas recomendações do W3C. A validade do documento é assegurada através das definições de esquemas como as DTD (*Document Type Definition).* Já existem muitas linguagens de transformação e consulta de documentos XML[FLO99, GOL98].

A característica fundamental da *XML* é que consegue impor uma estrutura ao documento através do aninhamento dos objetos, aqui chamados de elementos. As marcas de formatação (*tags)* identificam o significado do elemento que a segue [ABI2000, W3C2000].

Em *XML* existem várias formas para representar o esquema de dados utilizado na linguagem. Nesta ferramenta é empregado o conceito de *DTD (Document Type Definiton*) [W3C2000, HOL01]. A organização do processo de extração através da gramática, que decompõe a estrutura hierárquica do documento, executa com naturalidade a transformação dos dados em formato *XML* seguindo a *DTD.* A própria estratégia de extração segue um caminho que gera a informação extraída no formato da *DTD,* ou seja, declara a informação mais complexa e vai detalhando-a até chegar na informação atômica – estruturas aninhadas.

Para gerar a *DTD* e, consequentemente, as informações em *XML*, a ferramenta precisa de algumas determinações:

- Somente são geradas na DTD as produções oriundas da ontologia. As produções geradas pela ferramenta para resolver problemas de ambigüidade ou recursividade na gramática são desconsideradas.
- Cada produção validada da gramática se transforma em um elemento da *DTD.*
- O nome do elemento é o nome da própria produção, ou seja, do objeto da ontologia.

- Produções do tipo lista ou tupla (dados complexos) geram o sinal de "+" ao lado do nome do elemento, indicando repetição.
- Os não-terminais são identificados já na composição do elemento.
- Os terminais são identificados com o tipo usado pela *DTD* que é o "#PCDATA".
- O aninhamento utilizado pela *DTD* tem relação direta com a representação da gramática.

Uma única ontologia pode gerar diversas DTD conforme os objetos que estiverem representados na classe de documentos. Uma única classe de documentos também representada por uma gramática pode gerar várias DTD. Apesar de possuir a mesma estrutura hierárquica, existem regras alternativas para o mesmo objeto, ou mesmo, alguns objetos não são obrigatórios.

Seguindo o exemplo da gramática da FIGURA 3.7, pode-se exemplificar as possíveis *DTD* que a mesma gramática pode gerar conforme as informações representadas no documento. Na FIGURA 3.10, pode-se ver duas possibilidades de DTD. Na FIGURA 3.10 (a), a DTD representa uma extração completa com todos os conceitos pertencentes à gramática. Já a FIGURA 3.10 (b) representa um documento em que a editora não é discriminada nas publicações. Outras variações permitidas pela gramática se devem ao fato de os objetos "publicação" e "autor" ter um elemento apenas ou serem uma lista de elementos. Essas variações não provocam DTD diferentes, pois são representadas pelo conceito de uma lista com, no mínimo, um elemento.

```
<!Element obras+  (anoedição, publicação)>
        <!Element anoedição  (#pcdata)>
        <!Element publicação+  (autor, titulo, editora)>
                <!Element autor+  (nome)>
                        <!Element nome  (#pcdata)>
                <!Element titulo  (#pcdata)>
                <!Element editora  (#pcdata)>
```

a) Primeira Possibilidade

```
<!Element obras+  (anoedição, publicação)>
        <!Element anoedição  (#pcdata)>
        <!Element publicação+  (autor, titulo, editora)>
                <!Element autor+  (nome)>
                        <!Element nome  (#pcdata)>
                <!Element titulo  (#pcdata)>
```

b) Segunda Possibilidade

FIGURA 3.10– Exemplo de Possíveis DTD Geradas

Unindo o esquema oferecido pela *DTD* às informações extraídas, o resultado de todo processo gera um documento *XML*. A partir desse documento *XML* gerado, várias manipulações e consultas podem ser feitas, uma vez que existem muitas ferramentas de consulta aplicadas a documentos XML. Utilizando o exemplo, a FIGURA 3.11 demonstra o resultado parcial de uma extração em *XML* do documento da FIGURA 3.3, relacionado à gramática da FIGURA 3.7. Observa-se que os resultados

em XML demonstram a informação e sua estruturação. O objeto “obras” formado pelos objetos “anoedição” e “publicação”. O objeto “publicação” composto pelos objetos “autor”, “titulo” e “editora”.

```
<?xml version=”1.0”?>
<!doctype advent system esquema>
<obraspublicadas>
        <obras>
                <anoedicao>1999</anoedicao>
                <publicacao>
                        <autor>
                                <nome> Norman W. Paton</nome>
                        </autor>
                        <titulo> Active Rules in Database Systems</titulo>
                        <editora> Springer Verlag</editora>
                </publicacao>
                <publicacao>
                        <autor>
                                <nome> Asuman Dogac</nome>
                                <nome> M.Tamer Ozsu</nome>
                                <nome> Ozgur Ulusoy</nome>
                        </autor>
                        <titulo> Current Trends in data Management Technology</titulo>
                        <editora> Idea Group Publishing</editora>
                </publicacao>
        </obras>
        <obras>
                ...
        </obras>
</obraspublicadas>
```

FIGURA 3.11 – Resultado Gerado em XML

4 Experimentos e Resultados

Para validar o método de extração semântica de dados, foi desenvolvido um protótipo da ferramenta em linguagem C++ e realizados alguns experimentos.

Utilizando a arquitetura proposta neste trabalho (FIGURA 3.1), o protótipo contempla as fases de extração automática e de geração de resultados. Ele não atende à fase de montagem do primeiro exemplo, que depende de uma grande interação com o usuário, e envolve mais conceitos de interfaces homem-máquina do que do processo de extração em si. Essa etapa inicial está toda planejada e detalhadamente descrita na Seção 5, ficando como sugestão para implementação em trabalhos futuros. O programa parte do princípio de que o usuário já montou seu primeiro exemplo utilizando o texto, a ontologia e os tipos de dados, os quais geraram uma gramática. Como essa parte não está implementada, as gramáticas foram montadas manualmente, porém seguindo o raciocínio proposto neste trabalho. A partir da gramática, o protótipo extrai as informações desejadas e as reorganiza em formato XML.

Os experimentos realizados têm dois objetivos principais. O primeiro é mostrar que a gramática proposta contempla a maioria das situações de representação de dados semi-estruturados, desde regras alternativas para a mesma informação, até a composição exata de objetos complexos e aninhados. O segundo objetivo é demonstrar a capacidade do programa extrator em identificar e extrair automaticamente todos objetos definidos que pertencem aos documentos.

Várias páginas da *Web* foram utilizadas para experimentos. Selecionamos algumas mais significativas para demonstrar os resultados alcançados.

A página da base de dados db&lp com as referências da ACM TODS[6] foi utilizada como exemplo ao longo de todo este texto. A apresentação da página pode ser visualizada na FIGURA 3.3 e uma parte do fonte HTML na FIGURA 3.4. A ontologia sobre a qual foi definida está apresentada na FIGURA 3.2. A gramática gerada pela primeira fase que é a principal entrada dessa ferramenta está totalmente descrita na FIGURA 3.7.

Observando a FIGURA 3.3 é possível identificar que a página relaciona uma série de publicações, classificadas pelo ano de edição. Para cada ano de edição podem existir várias publicações. Cada publicação possui várias informações: os autores da publicação, o título e a editora. A informação sobre a editora não é obrigatória, pois algumas publicações não a relacionaram. Apesar de ser uma página extensa, pode-se resumi-la em quatro objetos atômicos: “anoedição”, “nome”, “título” e “editora”. Os objetos atômicos não fornecem toda a informação necessária, por isso precisam ser agrupados em objetos complexos que aninham as informações. Dessa forma, são compostos três objetos complexos: “autor” que é o conjunto de autores da publicação, “publicação” que lista todas publicações e suas informações, e “obras” que se refere ao conjunto de publicações. Esses sete objetos são distribuídos em quatro níveis hierárquicos. Em resumo, como pode ser visto na FIGURA 3.7: o objeto “obras”,

[6] Disponível em http://www.informatik.uni-trier.de/~ley/db/books/collections/index.html

correspondente a toda página, é uma lista composta pelos objetos “anoedição” e “publicação”; “anoedição” é um objeto atômico”; “publicação” é uma lista formada pelos objetos “autor”, “titulo” e “editora”; “autor” é uma nova lista formada pelo objeto atômico “nome”; “titulo” e “editora” são objetos atômicos.

Essa composição já é suficiente para demonstrar o aninhamento da estrutura, e o funcionamento do extrator em relação a objetos complexos e ao tratamento da recursividade da gramática. Esta página possui algumas outras particularidades relevantes. O objeto “editora” não é um campo obrigatório, e na gramática é representada por regras alternativas para a mesma produção. Essa condição permite verificar o comportamento do extrator perante objetos que não estão representados no documento. Para encontrar o objeto “titulo”, são possíveis quatro regras de combinação dependendo do delimitador utilizado. O “titulo” pode estar situado logo após a seqüência de caracteres “ (Eds.):”, “ (Ed.):”, “ (Hrsg.):” ou nenhuma seqüência, simplesmente após o objeto “autor”. Neste item, é importante observar que nem sempre existem delimitadores para as informações a serem extraídas. Como a ferramenta trata de estruturas aninhadas, a simples composição delas pode fornecer a informação desejada.

Ao observar o fonte HTML indicado, pode-se observar que os sete objetos definidos na gramática aparecem repetidas vezes no documento HTML:

- Objeto “obras”: Ocorre uma única vez, pois é o documento completo.
- Objeto “anoedição”: São 16 objetos, cada qual correspondente aos anos de publicação.
- Objeto “publicação”: São 34 objetos, pois o mesmo “anoedição” pode ter vários objetos “publicação”. Por exemplo, para o objeto “anoedição”, com valor “1996”, existem 6 objetos “publicação” e para “2001” existe apenas 1 objeto “publicação”.
- Objeto “autor”: Como o objeto “autor” é a lista de autores de uma mesma publicação e é um objeto obrigatório, existe na mesma proporção que “publicação”, ou seja, 34.
- Objeto “nome”: Como a mesma publicação pode possuir vários autores, este objeto é o que mais ocorre no texto, já que existem 78 objetos “nome”.
- Objeto “titulo”: Como cada “publicação” possui um título e este objeto é obrigatório. Existem também 34 objetos “titulo”.
- Objeto “editora”: Este objeto não é obrigatório, por isso seu número pode ser menor que o de “publicação”, são 21 objetos, ou seja, 13 “publicação” não possui o objeto “editora” associado.

O extrator foi executado e devolveu como resultado um documento XML com todos os objetos extraídos. Uma visão parcial dos resultados extraídos e formato XML pode ser vista na FIGURA 3.11. Todos os objetos do documento foram extraídos. O extrator buscou os objetos atômicos e remontou os objetos complexos de forma correta. Além disso, ele encontrou todos os objetos “titulo” que tinham vários delimitadores opcionais. Da mesma forma, o objeto “editora”, que nem sempre estava presente no documento, foi extraído de forma correta e, quando não ocorria, o extrator conseguiu seguir o fluxo do processo sem maiores problemas. Em nenhum momento, foi necessário intervir no processo de extração. A TABELA 4.1 apresenta um resumo dos objetos extraídos das páginas citadas.

O outro experimento feito foi com a página de venda de produtos da empresa Submarino[7] . Além das características de extração demonstradas no exemplo anterior, a página da Submarino tem uma particularidade em especial: uma página inicial possui várias páginas encadeadas e todas com a mesma estrutura, ou melhor, todas pertencem à mesma classe de documentos. Cada uma destas páginas encadeadas trata da venda de um produto específico: equipamentos eletrônicos, aparelhos de telefonia, produtos de informática, livros, CD, DVD, brinquedos e jogos. Apesar de serem produtos independentes, a estrutura do documento é sempre a mesma. Como estes documentos estão relacionados à mesma ontologia e possuem semelhante estrutura hierárquica pertencem à mesma classe de documentos, basta o usuário exemplificar apenas uma delas. Dessa forma, com a mesma gramática sobre ofertas de vendas, a ferramenta pode extrair objetos de qualquer uma das páginas. É importante destacar que outras páginas comerciais podem possuir esta mesma estrutura sobre ofertas de vendas, o que implicaria em um incremento da classe de documentos.

Para que a descrição do exemplo não fique muita extensa, especifica-se apenas o experimento com duas páginas da Submarino: equipamentos de informática e livros. A aparência da página sobre equipamentos de informática consta na FIGURA 4.1 e uma parte do fonte HTML na FIGURA 4.3. O mesmo ocorre com a página sobre livros nas FIGURAs 4.2 e 4.4 respectivamente.

A estrutura das páginas pode ser classificada em três níveis de aninhamento: o objeto “ofertas” é composto pelos objetos “tipo oferta”, “oferta principal” e “oferta simples”. Objeto “oferta principal”, por sua vez, é formada por “produto”, “preço”, “condição especial de pagamento” e “comentários”. O objeto “Oferta simples” é composto apenas por “produto” e “preço”. Essa estrutura de aninhamento pode ser visualizada na gramática da FIGURA 4.5.

Ao observar as páginas pode-se observar que elas estão separadas em duas colunas distintas. Dessa forma, o objeto complexo “ofertas” é uma lista formada por três listas “tipo oferta”, “oferta principal” e “oferta simples”. O objeto “tipo oferta” é uma lista, pois normalmente possui uma seqüência de dois tipos de oferta, uma para cada coluna. Dessa mesma forma está estruturado o objeto “oferta principal”. O objeto “oferta simples” também é uma lista, mas, nesse caso, o número de repetições é indefinido, apresentando normalmente três repetições. Isso não é relevante, pois o objeto está definido como uma lista, onde a única restrição é o fato de possuir pelo menos um elemento. Outra particularidade destas páginas é que os objetos atômicos “preço” e “condição de pagamento” nem sempre estão presentes nas ofertas principais, não constituindo-se dessa forma, em campos obrigatórios.

[7] Disponível em http://www.submarino.com.br

FIGURA 4.1 - Página da Submarino – Oferta de Equipamentos de Informática

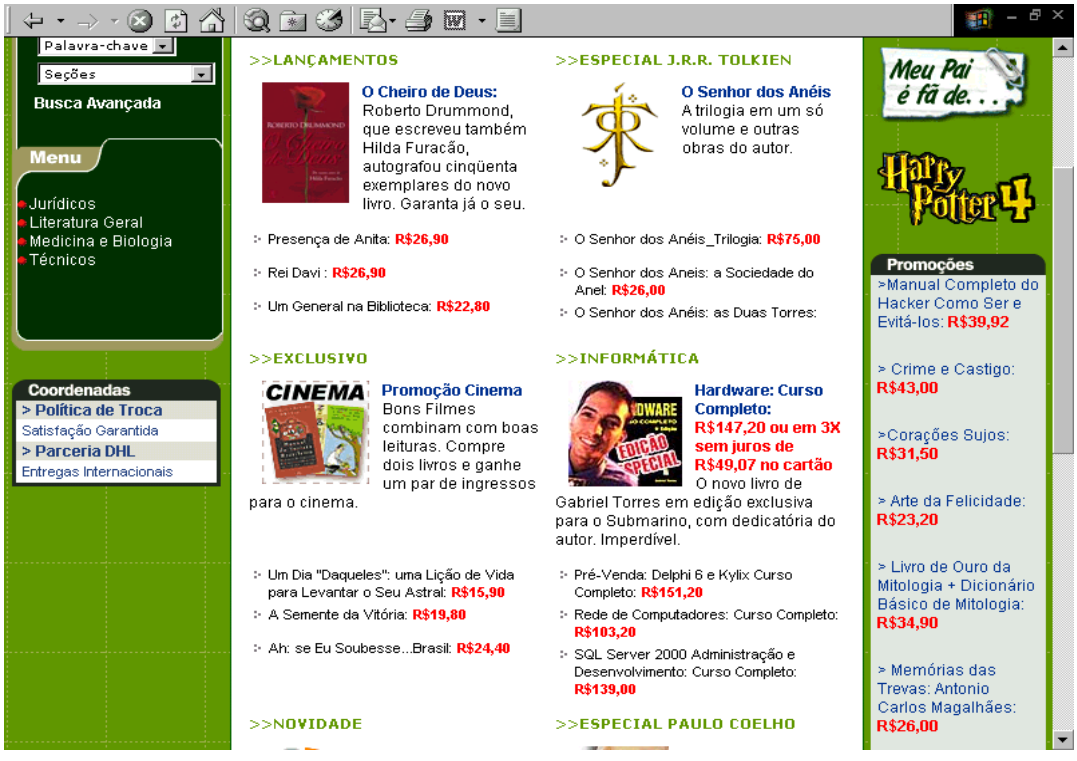


FIGURA 4.2 - Página da Submarino – Oferta de Livros

```
<!---------- INICIO CONTEUDO INFORMATICA 1---------------------------------------------->

<table width="100%" border="0" cellpadding="6" cellspacing="0" align="center">
  <tr valign="top">
    <td width="50%"><a class=infounderhat
href="/software_speciallist.asp?Query=ProductPage&ProdTypeId=10&Top3rdRowKey=807&PROMOID=2906"
>&gt;&gt;DESKTOPS</a></td>
    <td width="50%"><a class=infounderhat
href="/software_speciallist.asp?Query=ProductPage&ProdTypeId=10&Top3rdRowKey=807&PROMOID=2059"
>&gt;&gt;PALMTOPS</a></td>
  </tr>
  <tr valign="top">
    <td width="50%" span class="text"><a
href="/software_productdetails.asp?Query=ProductPage&ProdTypeId=10&ProdId=154321&ST=HI04"><img
src="/images/software/br/pqa154321.jpg" border="0" align="left" hspace="10"><b>5000LA
      Celeron 766 Mhz Compaq:
      <!--inicio preco-->
      <span class='price'>R$1,799.00 ou em 6x sem juros de R$299.83 no cartão</span>
      <!--fim preco-->
      </b></a><br>
      64 MB de mem&oacute;ria expans&iacute;vel, Processador Intel 766 Mhz, 20
      MB de disco r&iacute;gido, CD-ROM de 48x e Windows Millenium Edition instalado.</td>
    <td width="50%" span class="text"><a
href="/software_productdetails.asp?Query=ProductPage&ProdTypeId=10&ProdId=139958&ST=HI01"><img
src="images/software/br/pqa139958.jpg" border="0" align="left" hspace="10"><b>Palm
      Vx:
      <!--inicio preco-->
      <span class='price'>R$799.00 ou em 6x sem juros de R$133.17 no cartão</span>
      <!--fim preco-->
      </b></a><br>
      Armazena 400 e-mails, 1000 notas e 8 anos de compromissos.</td>
  </tr>
  <tr valign="top">
    <td width="50%">
      <table width="100%" border="0" cellspacing="0" cellpadding="2">
        <tr valign="top">
          <td width="9"><img src="http://www.submarino.com.br/images/br/bullet.gif" border="0" align="top">
          </td>
          <td width="100%" align="left"><a
href="/software_productdetails.asp?Query=ProductPage&ProdTypeId=10&ProdId=144600&ST=HI0401" class="bullet">Presario
            5WV253 AMD Duron 700MHz:
            <!--inicio preco-->
            <span class="price">
            R$2,799.00
            <!--fim preco-->
            </span></a></td>
        </tr>
        <tr valign="top">
          <td width="9"><img src="http://www.submarino.com.br/images/br/bullet.gif" border="0" align="top">
          </td>
          <td width="100%" align="left"><a
href="/software_productdetails.asp?Query=ProductPage&ProdTypeId=10&ProdId=144602&ST=HI0402" class="bullet">5WV293
            AMD Athlon 950MHz Compaq:
            <!--inicio preco-->
            <span class="price">
            R$4,199.00
            <!--fim preco-->
            </span></a></td>
        </tr>
        <tr valign="top">
          <td width="9"><img src="http://www.submarino.com.br/images/br/bullet.gif" border="0" align="top">
          </td>
                    <td width="100%" align="left"><a
href="/software_productdetails.asp?Query=ProductPage&ProdTypeId=10&ProdId=156039&ST=HI0403" class="bullet">Toshiba
            Lince 9170 CDT :
            <!--inicio preco-->
```

FIGURA 4.3 - Amostra Fonte HTML - Equipamentos de Informática

```
<!---------- INICIO CONTEUDO LIVROS 1 ---------------------------------------------------------->

<table width="100%" border="0" cellpadding="6" cellspacing="0" align="center">
<tr valign="top">
<td align="left" width="50%"><a class=booksunderhat
href="http://www.submarino.com.br/books_newreleases.asp?Query=ProductPage&ProdTypeId=1&Top3rdRowKey=10&TEMPLATEI
D=HEADER"
>&gt;&gt;LAN&Ccedil;AMENTOS</a></td>
<td align="left" width="50%"><a class=booksunderhat
href="/local/books/books_senhor_aneis.asp?QUERY=ProductPage&ProdTypeId=1"
>&gt;&gt;ESPECIAL J.R.R. TOLKIEN</a></td>
</tr>
<tr valign="top">
<td width="50%" class="text"><a
href="http://www.submarino.com.br/books_productdetails.asp?Query=ProductPage&ProdTypeId=1&ProdId=155989&HL01"><img
src="https://www.submarino.com.br/images/books/cover_tn/pq155989.jpg" border="0" align="left" hspace="10"><b>O Cheiro de
Deus …Autografado..Os 50 primeiros.</b> <span class='price'>R$29.90</span></a><br>
Roberto Drummond, que escreveu também Hilda Furacão, autografou cinqüenta exemplares do novo livro. Garanta já o seu.
</td>
<td width="50%" class="text"><a href="/local/books/books_senhor_aneis.asp?QUERY=ProductPage&ProdTypeId=1&HL02"><img
src="/images/books/br/botao_senhor_aneis.gif" border="0" align="left" hspace="10"><b>O Senhor dos Anéis</b></A><br>A trilogia
em um só volume e outras obras do autor.<br>

</td>
</tr>
<tr valign="top">
<td width="50%">
<table width="100%" border="0" cellspacing="0" cellpadding="2">
<tr valign="top">
<td width="9" align="left"><img src="/images/br/bullet.gif" border="0" align="top">
</td>
<td align="left" valign="top" width="100%"><a
href="http://www.submarino.com.br/books_productdetails.asp?Query=ProductPage&ProdTypeId=1&ProdId=155987"
class="bullet">Presença de Anita:<!--inicio preco--><span class="price"> R$24.90
<!--fim preco--></span></a></td>
</tr>
<tr valign="top">
<td width="9"><img src="/images/br/bullet.gif" border="0" align="top">
</td>
<td align="left" valign="top" width="100%"><a
href="http://www.submarino.com.br/books_productdetails.asp?Query=ProductPage&ProdTypeId=1&ProdId=156184"
class="bullet">Rei Davi:<!--inicio preco--><span class="price"> R$26.90
<!--fim preco--></span></a></td>
</tr>
<tr valign="top">
<td width="9"><img src="/images/br/bullet.gif" border="0" align="top">
</td>
<td align="left" valign="top" width="100%"><a
href="http://www.submarino.com.br/books_productdetails.asp?Query=ProductPage&ProdTypeId=1&ProdId=156263"
class="bullet">Um General na Biblioteca:<!--inicio preco--><span class="price"> R$22.80
<!--fim preco--></span></a></td>
</tr>

</tr>
</table>
</td>
<td width="50%">
<table width="100%" border="0" cellspacing="0" cellpadding="2">
<tr valign="top">
<td width="9" align="left"><img src="/images/br/bullet.gif" border="0" align="top">
</td>
<td align="left" valign="top" width="100%"><a
href="http://www.submarino.com.br/books_productdetails.asp?Query=ProductPage&ProdTypeId=1&ProdId=153872"
class="bullet">O Senhor dos Anéis_Trilogia:<!--inicio preco--><span class="price"> R$75.00
<!--fim preco--></span></a></td>
```

FIGURA 4.4 - Amostra Fonte HTML - Livros

```
ofertas:=          *INICIO CONTEUDO* {tipo} {ofprincipal} {ofsimples} {lofertas1}

lofertas:=         {ofertas}|&

tipo:=             <tr valign=*>*  {ltipo}   </tr>

ltipo:=           <td*&gt;&gt;* [A-Z]  (tammin=1tammax=999) </a></td> {ltipo1}

ltipo1:=           {ltipo}|&

ofprincipal:=      <tr valign=*>* {lofprincipal} </tr>

lofprincipal:=     <td *&ProdTypeId=*<b>* {produto} {preco} {comentario} </td> {lofprincipal1}

lofprincipal1:=    {lofprincipal}|&

produto:=          [A-Za-z0-9.]  (tammin=1 tammax=999)

preco:=            *price*>*R$ [0-9.0-9]  (tammin=1 tammax=999) {condpagto} </span> | &

condpagto:=        ou [A-Za-z0-9.] (tammin=1 tammax=999)|&

comentario:=       </b></a><br> [A-Za-z0-9.]  (tammin=1 tammax=999) |
                   </a><br> [A-Za-z0-9.]  (tammin=1 tammax=999)

ofsimples:=        <tr valign=*>* {lofsimples} {lofsimples} </tr>

lofsimples:=       <td*<table*>* {lofsimples1} </table>*</td>*

lofsimples1:=      <tr valign=*>*<td*</td>* {produtosp}    {precosp} </td>*</tr>* {lofsimples2}

lofsimples2:=      {lofsimples1}|&

produtosp:=   <td*bullet*>* [A-Za-z0-9.]  (tammin=1tammax=999){produtosp1}

produtosp1:=       <b>|&

precosp:=          *price*>*R$ [0-9.0-9]  (tammin=1tammax=999)
                   <!--fim preco-->*</span>*{precosp2}

precosp2:=         </b></a>|</a>
```

FIGURA 4.5 – Gramática da Página Submarino

A página que trata de equipamentos de informática contém os seguintes objetos:

- "ofertas": É único, pois refere-se a toda página.
- "tipo oferta": São 10 tipos de ofertas diferentes.
- "oferta principal": Como existe apenas uma "oferta principal" para cada "tipo" também são 10 objetos.
- "oferta simples": São 30 objetos "oferta simples" pois normalmente são definidas 3 para cada tipo.
- "produto": Existem 10 objetos "produto" para "oferta principal" e 30 para "oferta simples".
- "preço": Nem todos os preços de "oferta principal" são apresentados. Dessa forma são 9 objetos "preço" para "oferta principal" e 30 para "oferta simples".

- “condição pagamento”: este objeto somente faz parte de “oferta principal” e nem sempre são definidos, existindo 9 objetos.
- “comentário”: também existe somente para “oferta principal” e são em 10.

A página que trata de livros contém os seguintes objetos:

- “ofertas”: É único, pois refere-se a toda página.
- “tipo oferta”: São 8 tipos de ofertas diferentes.
- “oferta principal”: Como existe apenas uma ”oferta principal” para cada “tipo”, também são 8 objetos.
- “oferta simples”: São 20 objetos “oferta simples”, pois normalmente são definidas 3 para cada tipo, sendo que 2 tipos possuem apenas 1 “oferta simples”.
- “produto”: Existem 8 objetos “produto” para “oferta principal” e 20 para “oferta simples”.
- “preço”: Nem todos os preços de “oferta principal” são apresentados. Dessa forma existe apenas 1 objeto “preço” para “oferta principal” e 20 para “oferta simples”.
- “condição pagamento”: Este objeto somente faz parte de “oferta principal” e nem sempre são definidos, existindo apenas 1 objeto.
- “comentário”: Também existe somente para “oferta principal” e são em 8.

Conforme a TABELA 4.1, pode-se observar que o extrator buscou 100% dos objetos definidos. Além disso, tratou com correção os objetos não obrigatórios e a composição da estrutura de aninhamento. Com a mesma gramática ou mesmo extrator, pode-se extrair informações de diversas páginas diferentes. Os objetos extraídos geraram o documento XML que possui uma parte representada na FIGURA 4.6.

```
<?xml version=”1.0”?>
<!doctype advent system esquema>
<oferta-venda>
        <ofertas>
                <tipo>DESKTOPS</tipo>
                <ofprincipal>
                        <produto> 5000LA Celeron 766 MHz Compaq: </produto>
                        <preço>1.799,00 </preço>
                        <cond.pagto.>ou em 6x sem juros de R$299,83 no cartão</cond.pagto>
                        <comentario>64 MB de memória expansível, Processador Intel 766
                                Mhz, 20 MB de disco rígido, CD-ROM de 48x e
                                Windows Millenium Edition instalado</comentario>
                </ofprincipal>
                <ofsimples>
                        <produto>Presario 5WV253 AMD Duron 700MHz</produto>
                        <preco>2.799,00</preco>
                </ofsimples>
                <ofsimples>
                        <produto>5WV293 AMD Athlon 950 MHz Compaq</produto>
                        <preco>4.199,00</preco>
                </ofsimples>
                <ofsimples>
                        <produto>Toshiba Lince 9170 CDT</produto>
                        <preco>2.007,00</preco>
                </ofsimples>
        </ofertas>
        <ofertas>
                <tipo>PALMTOPS    </tipo>
                ...
        </ofertas>
</ofertasvenda>
```

FIGURA 4.6 - Resultado XML da Página da Submarino - Informática

O sucesso do processo de extração reside no fato de que a gramática utilizada pela ferramenta é capaz de representar as várias possibilidades de representação da informação. Ela permite estruturar objetos complexos com vários níveis de aninhamento, assegurando flexibilidade, uma vez que permite formas de extração diferentes para o mesmo objeto. Proporciona robustez ao extrator, uma vez que não se perde na falta de um determinado objeto.

TABELA 4.1 – Resumo Objetos Extraídos

Página	Objeto	Quantidade		% Extraído
		Documento	Extraída	
ACM	Obras	1	1	100
	Ano de edição	16	16	100
	Publicação	34	34	100
	Autor	34	34	100
	Nome	78	78	100
	Titulo	34	34	100
	Editora	21	21	100
Submarino Equipamentos Informática	Oferta	10	10	100
	Tipo	10	10	100
	Oferta Principal	10	10	100
	Oferta Simples	30	30	100
	Produto	40	40	100
	Preço	39	39	100
	Cond. Pagto.	9	9	100
	Comentário	10	10	100
Submarino Livros	Oferta	8	8	100
	Tipo	8	8	100
	Oferta Principal	8	8	100
	Oferta Simples	20	20	100
	Produto	28	28	100
	Preço	21	21	100
	Cond. Pagto.	1	1	100
	Comentário	8	8	100

5 Interação com o Usuário Através de Ferramentas Visuais

Esta Seção apresenta um esboço do funcionamento da fase de montagem do primeiro exemplo, a qual exige uma interação direta com o usuário. Como já foi mencionado na Seção anterior, o protótipo desenvolvido abrangeu as fases de extração automática e geração de resultados. A proposta dessa Seção é abrir caminho para a continuação do desenvolvimento do protótipo, completando a fase de interação com o usuário.

Como essa fase exige um processo de interação com o usuário, é necessário oferecer uma interface amigável. Uma interface amigável é aquela que se aproxima o máximo possível da realidade e das experiências de quem a opera, devendo ser intuitiva. O foco de atenção do usuário deve ser o processo proposto pela ferramenta, e não a forma de interagir com ela [SHN99, LIN99].

Tal ferramenta exige que muitas informações fiquem disponíveis em tela, ou seja, tanto o texto do documento, quanto à ontologia e as regras de extração. Para exibir as informações, é necessário um estudo aprofundado em técnicas de interfaces visuais [COR97]. Para aproximar as interfaces das ferramentas do mundo real do usuário, muitas técnicas oriundas da área de computação gráfica realizam simulação de três dimensões [YOU99].

É preciso que a ferramenta mantenha disponível em tela quatro informações necessárias para a construção das regras de extração. Cada uma dessas informações dispostas em um quadro a parte, como pode ser visto na FIGURA 5.1:

- *Documento*: Exibe o texto do documento a ser exemplificado. Dependendo do tamanho do documento, não é possível sua visualização total na tela.
- *Ontologia*: Para que o usuário associe as porções do texto aos objetos da ontologia e, dessa forma, possa se valer do conhecimento já embutido nela, é preciso exibir a ontologia.
- *Tipos de Dados*: Todos os tipos de dados disponíveis para o objeto em questão. A ferramenta somente disponibiliza para o usuário os tipos de dados que estão de acordo com as restrições de cardinalidade dos relacionamentos da ontologia e sugere o melhor tipo de dado baseado no conhecimento já adquirido em outros processos de extração envolvendo a ontologia.
- *Regras de Extração*: Em um último quadro são exibidas as regras de extração. A distribuição das regras no quadro segue a hierarquia dos objetos associados ao documento. Está utilizando uma estrutura tabular para representar este aninhamento das informações, iniciando dos objetos mais complexos, e a cada nível de aninhamento deixa uma margem maior em relação ao item imediatamente de nível superior. Essas regras, internamente, compõem a gramática, que é o guia do processo de extração. Porém gramáticas não são de uso comum, o que leva a propor uma representação alternativa ao usuário.

O usuário pode optar pela seqüência de objetos a ser definido nessa etapa. Algumas regras impostas pela ferramenta precisam ser seguidas conforme explicado na Seção 3.3.1. No exemplo utilizado, o usuário segue a definição mais intuitiva: do todo

para o detalhe, da esquerda para a direita: “obras”, “anoedição”, “ publicação”, “autor”, “nome”, “titulo”, “editora”.

A FIGURA 5.1 espelha uma imagem inicial do processo de montagem do exemplo, no momento em que o usuário não realizou ainda nenhuma intervenção.

Pode-se adotar como raciocínio básico para compreender o processo: o usuário contorna porções do texto e as associa a objetos da ontologia. Conforme o objeto, a ferramenta disponibiliza os tipos de dados compatíveis. O usuário escolhe o tipo de dado que melhor representa o objeto, informa os delimitadores, e estabelece os parâmetros de validação. O processo sempre deve partir da informação mais completa para a mais detalhada, ou seja, do dado complexo para o atômico.

A ferramenta deve consistir alguns critérios para garantir a correção do processo de montagem do exemplo:

a) A ontologia, através da cardinalidade de seus relacionamentos, obriga que o usuário monte o exemplo utilizando sempre os objetos não léxicos antes dos objetos léxicos relacionados a ele. No exemplo da FIGURA 3.2, o usuário pode optar em não identificar o objeto não léxico “publicação”, por não ser necessário na representação daquela classe de documentos, e exemplificar apenas os objetos “editora” e “anoedição”. Entretanto, se quiser exemplificar essa relação, é obrigado a identificar o objeto “publicação” antes dos objetos léxicos “editora” e “anoedicao”.
b) A ferramenta somente disponibiliza tipos complexos quando o relacionamento associado ao objeto da ontologia tiver cardinalidade “n” no sentido do outro objeto.
c) A ferramenta sugere tipos de dados conforme o conhecimento aprendido em outras modelagens.

Utilizando o mesmo exemplo da FIGURA 3.3 ao longo deste texto, demonstram-se as interações que o usuário deve fazer para montar o primeiro exemplo da classe de documentos relacionada à ontologia de obras publicadas:

a) Primeira Interação – Definição da Lista “obras” (FIGURA 5.2)

O usuário contorna a porção de interesse mais ampla do texto, que contém todas as publicações (contorno em linha pontilhada no texto da FIGURA). Como já dito anteriormente, a primeira marcação não tem associação a um objeto da ontologia, e sim ao domínio que ela representa. Dessa forma, não associa a porção do texto a nenhum objeto da ontologia, mas dá um nome a ela, sendo utilizado para o exemplo “obras”. Por se tratar da porção mais ampla do texto, a ferramenta disponibiliza para escolha tanto os tipos complexos como atômicos. O usuário seleciona o tipo lista e a opção de obrigatoriedade de, no mínimo, um elemento, clicando sobre eles (opções em negrito no quadro “Tipos de Dados” da FIGURA). Depois disso, demonstra ao sistema os delimitadores à esquerda e à direita da lista. Para os delimitadores, o usuário pode optar em usar um delimitador exato ou se deseja associá-lo ao caracter coringa (“*”). Nesse caso, o usuário clica em “delimitador à esquerda” e “exato” (em negrito no quadro “Tipos de Dados” da FIGURA), e, imediatamente, contorna no texto a seqüência de caracteres que o representa “<hr>” (em negrito no texto). Procede da mesma maneira para o delimitador à direita. Na FIGURA, somente pode ser visto o delimitador à esquerda, pois à direita não aparece no texto de exemplo. Como se trata de uma lista, precisa identificar o primeiro elemento da mesma, clicando nessa opção e, em seguida, contornando a área que representa o primeiro elemento da lista (contorno em linha

pontilhada dentro do contorno maior). Essa lista não possui delimitadores entre elementos, mas, se existisse, ele deveria ser indicado logo em seguida marcando a cadeia de caracteres. O resultado da montagem deste dado pode ser visto no quarto quadro da FIGURA “regras de extração”.

b) Segunda Interação – Definição do Dado Atômico “anoedição” (FIGURA 5.3)

O dado mais complexo do documento está definido como uma lista e já foi indicado o primeiro elemento dela. As informações devem continuar a ser decompostas, sendo que, o próximo passo é identificar a composição dos elementos da lista “Obras”. O usuário pode optar em definir “anoedição” ou “publicação”. A ordem não importa, pois a ferramenta monta a seqüência dos dados conforme a ordem em que aparece no texto. No exemplo, o usuário opta por definir “anoedição”, contorna a porção do texto que contém um exemplo desse dado (contorno em linha pontilhada no texto da FIGURA), e clica no objeto da ontologia que identifica “anoedição” (retângulo com linhas mais largas que os demais objetos no quadro ontologia). Não existe nenhum relacionamento associado a “anoedição” com cardinalidade “n” para este objeto. Desta forma, a ferramenta somente disponibiliza tipos de dados atômicos para o usuário escolher. O usuário escolhe o tipo de dado atômico inteiro (opção em negrito no quadro “Tipos de Dados” da FIGURA). A ferramenta permite que o usuário informe ainda os delimitadores e os parâmetros. O usuário clica em “delimitadores à esquerda” e “exato” e contorna no texto a cadeia de caracteres “<h2>”. Clica em “delimitadores à direita” e “exato” e contorna no texto “</h2>”. Os delimitadores estão em negrito no texto da FIGURA. Para restringir mais o filtro de busca da informação, o usuário clica em “tamanho mínimo” e informa o numeral quatro. Segue o mesmo raciocínio para os demais parâmetros: tamanho máximo com valor 4, valor mínimo com conteúdo 1500 e valor máximo com conteúdo 2999. Essa nova regra criada pode ser vista no quadro de “Regras de Extração”.

c) Terceira Interação – Definição da Lista “publicação” (FIGURA 5.4)

O próximo dado a ser definido ainda faz parte da composição da lista “Obras”. O usuário contorna, dentro do primeiro elemento a seqüência de caracteres que compõe o dado “publicação” (contorno em linha pontilhada no texto). Em seguida, clica no objeto da ontologia que o representa (retângulo com linhas mais largas no quadro ontologia). A ferramenta identifica que esse objeto tem quatro relacionamentos com cardinalidade “n”: um “anoedição” pode ter várias “publicação”, um “autor” pode publicar várias “publicação”, uma “editora” pode registrar várias “publicação”, e um “preço” pode estar associado a várias “publicação”. Por possuir esse tipo de cardinalidade, a ferramenta deixa opcional tanto os dados atômicos como os complexos. Neste exemplo, o usuário marca para “publicação” um dado complexo, do tipo lista, com obrigatoriedade de, no mínimo, um elemento (em negrito no quadro “Tipos de Dados”). Depois disso, demonstra ao sistema os delimitadores à esquerda e à direita da lista. Clica em “delimitador à esquerda” e “exato” (em negrito no quadro tipos de dados da FIGURA), e, imediatamente, contorna no texto a seqüência de caracteres que o representa “
” (em negrito no texto). Escolhe “delimitador à direita”, “coringa à esquerda” e “coringa à direita” (em negrito nos tipos de dados), e contorna no texto a seqüência “” (em negrito no texto). Como a opção foi de delimitador à direita com coringa à esquerda e à direita, o delimitador transforma-se de “” para “*
*” (quadro de “Regras de Extração” da FIGURA). Esta lista não possui delimitadores entre elementos, então

deve-se pular para a demonstração do primeiro elemento da lista contornando a porção do texto (contorno em linha pontilhada dentro do contorno da lista “publicação”).

d) Quarta Interação – Definição da Lista “autor” (FIGURA 5.5)

Como “publicação” é um dado complexo, é preciso decompô-lo até chegar no nível de dados atômicos, mesmo que a lista “obras” fosse composta por outros dados. O usuário contorna, dentro do dado publicação, a seqüência de caracteres que compõe o dado “autor” (contorno em linha pontilhada no texto). Em seguida, clica no objeto da ontologia que o representa (retângulo com linhas mais largas no quadro “Ontologia”). A ferramenta identifica que o objeto tem um relacionamento com “publicação” com cardinalidade “n”, ou seja, uma publicação pode ser escrita por vários autores. Por possuir este tipo de cardinalidade, a ferramenta deixa opcional tanto os dados atômicos como os complexos. Nesse exemplo, o usuário marca para “autor” um dado complexo, do tipo lista, com obrigatoriedade de, no mínimo, um elemento (em negrito no quadro “Tipos de Dados”). Depois disso, demonstra ao sistema os delimitadores à esquerda e à direita da lista. Clica em “delimitador à esquerda” e “coringa à esquerda” (em negrito no quadro tipos de dados da FIGURA) e, imediatamente, contorna no texto a seqüência de caracteres que o representa “<li>” (em negrito no texto). Como a opção foi de delimitador à esquerda com coringa, o delimitador transforma-se de “<li>” para “*<li>” (quadro de “Regras de Extração” da FIGURA). Para delimitador à direita existem três alternativas. Desta vez, deve clicar a primeira vez em “delimitador à direita” e “exato” e marcar no texto a primeira opção “ (Ed.):”. Como são delimitadores alternativos, ou seja, valida um ou outro delimitador, o usuário deve clicar em “alternativo” e depois fazer a segunda combinação. Deve proceder da mesma maneira para a terceira possibilidade de delimitador. Essa demonstração não está ilustrada na FIGURA, pois a parcial do texto utilizada não contém os delimitadores alternativos. Como pode ser visto no quadro “Regras de Extração”, os delimitadores são “(Ed.):” ou “ (Eds.):” ou “ (Hrsg.):”. A lista “autor” possui separador de elementos da lista, sendo que o usuário deve clicar nessa opção em negrito no quadro tipos de dados e, em seguida, marcar no texto este delimitador (em negrito no texto) que é uma vírgula (“,”). Obedecendo às exigências da ferramenta, o usuário demonstra o primeiro elemento da lista “autor” contornando a região (contorno em linha pontilhada dentro do contorno da lista).

e) Quinta Interação – Definição do Dado Atômico “nome” (FIGURA 5.6)

Como “autor” é um dado complexo, é preciso decompô-lo até chegar no nível de dados atômicos, mesmo que a lista “publicação” não tenha sido toda trabalhada. O usuário contorna, dentro do dado “autor”, a seqüência de caracteres que compõe o dado “nome” (contorno em linha pontilhada no texto). Em seguida, clica no objeto da ontologia que o representa (retângulo com linhas mais largas no quadro ontologia). Não existe nenhum relacionamento associado a “nome” com cardinalidade “n” para esse objeto. Assim, a ferramenta somente disponibiliza tipos de dados atômicos para o usuário escolher. O usuário escolhe o tipo de dado atômico conjunto de nomes próprios (opção em negrito no quadro “Tipos de Dados” da FIGURA). A ferramenta permite que o usuário informe ainda os delimitadores e os parâmetros. O usuário clica em “delimitadores à esquerda” e “exato” e contorna no texto a cadeia de caracteres “<a”. Este delimitador é composto por várias seqüências de caracteres intercaladas por coringas. Quando essa necessidade ocorrer, o usuário deve utilizar a possibilidade de união de vários delimitadores. Dessa forma, deve clicar em “união” e na seqüência, em

“coringa à esquerda” e marcar no texto a porção correspondente “href=”. Repete esse processo, clicando em “união”, “coringa à esquerda” e marca no texto a seqüência “>”. Essa união de combinações resulta no delimitador “<a*href=*>”. Clica em “delimitadores à direita” e exato e contorna no texto “”. Os delimitadores estão em negrito no texto da FIGURA. Para restringir mais o filtro de busca da informação, o usuário clica em “tamanho mínimo” e informa o numeral 1 e, em “tamanho máximo”, o numeral 999. Isso significa que o tamanho máximo de um nome pode ser de 999 caracteres e no mínimo deve possuir um caracter. O usuário possui um cadastro de nomes próprios e, para restringir o filtro, indica que um nome para ser válido deve estar previamente cadastrado no arquivo “arq_autores”.

f) Sexta Interação – Definição do Dado Atômico “Titulo” (FIGURA 5.7)

Como foi concluída a definição do dado complexo “autor”, o usuário volta a definir o dado complexo hierarquicamente maior “publicação”. O usuário contorna, dentro do dado “publicação” a seqüência de caracteres que compõe o dado “titulo” (contorno em linha pontilhada no texto). Em seguida, clica no objeto da ontologia que o representa (retângulo com linhas mais largas no quadro ontologia). Não existe nenhum relacionamento associado a “titulo” com cardinalidade “n” para esse objeto. Dessa forma, a ferramenta somente disponibiliza tipos de dados atômicos para o usuário escolher. O usuário escolhe tipo de dado atômico conjunto de nomes próprios (opção em negrito no quadro “Tipos de Dados” da FIGURA). A ferramenta permite que o usuário informe ainda os delimitadores e os parâmetros. Como não existe delimitador à esquerda, o usuário clica em “delimitadores à direita” e “exato” e contorna no texto “.”.Os delimitadores estão em negrito no texto da FIGURA. Para restringir mais o filtro de busca da informação, o usuário clica em “tamanho mínimo” e informa o numeral 1 e em “tamanho máximo” o numeral 999. Isso significa que o tamanho máximo de um título pode ser de 999 caracteres e, no mínimo, deve possuir um caracter.

g) Sétima Interação – Definição do Dado Atômico “editora” (FIGURA 5.8)

O usuário contorna, dentro do dado “publicação”, a seqüência de caracteres que compõe o dado “editora” (contorno em linha pontilhada no texto). Em seguida, clica no objeto da ontologia que o representa (retângulo com linhas mais largas no quadro ontologia). Não existe nenhum relacionamento associado a “editora” com cardinalidade “n” para este objeto. Dessa forma, a ferramenta somente disponibiliza tipos de dados atômicos para o usuário escolher. O usuário escolhe tipo de dado atômico conjunto de nomes próprios (opção em negrito no quadro “Tipos de Dados” da FIGURA). A ferramenta permite que o usuário informe ainda os delimitadores e os parâmetros. Como não existe delimitador à esquerda, o usuário clica em “delimitadores à direita” e “exato” e contorna no texto “,”. Entretanto, existem outros delimitadores alternativos à direita do dado “editora”, que não aparecem na FIGURA. Assim, o usuário deve clicar em “alternativo”, novamente em “exato” e realizar a nova marcação no texto na seqüência de caracteres “.”. Os delimitadores estão em negrito no texto da FIGURA. Para restringir mais o filtro de busca da informação, o usuário clica em “tamanho mínimo” e informa o numeral 1 e em “tamanho máximo” o numeral 999. Isso significa que o tamanho máximo do nome de uma editora pode ser de 999 caracteres e, no mínimo, deve possuir um caracter.

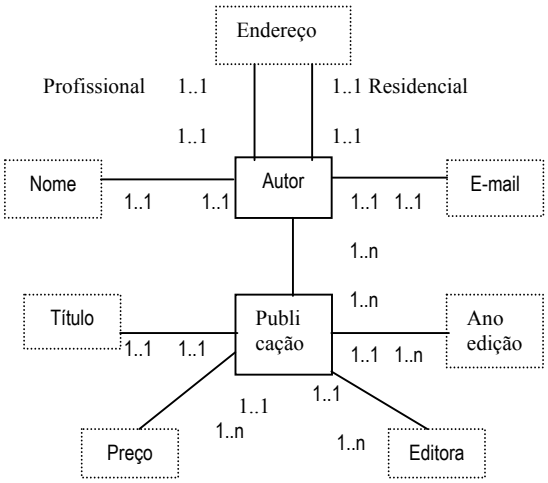
O exemplo simulou uma interação com o usuário descrevendo exatamente os passos a serem seguidos. Isso não significa que deve ser exatamente na ordem demonstrada, por isso são importantes algumas considerações:

- O usuário, ao decompor um dos elementos da lista, não precisa fazê-lo utilizando obrigatoriamente o primeiro elemento da lista. Ele pode utilizar qualquer um dos elementos. Em alguns casos, ao ocorrerem mudanças nas regras de extração, torna-se necessário demonstrar dados de vários elementos de uma lista.
- O usuário não precisa ter a preocupação de definir os dados dentro de uma determinada seqüência. A única ordem que deve seguir é a hierarquia do documento, ou seja, dos dados mais complexos para os atômicos. Dentro de um dado complexo, por exemplo, não precisa definir sua composição da esquerda para a direita. A ferramenta conhece a seqüência dos dados por sua posição física no texto.
- A ferramenta sempre disponibiliza somente os tipos de dados que estão de acordo com as restrições de cardinalidade do objeto na ontologia. No exemplo demonstrado, a ontologia está sendo utilizada pela primeira vez. Ao utilizá-la novamente em outros processos de exemplificação, o conhecimento aprendido sobre os objetos é repassado ao usuário, através das restrições e/ou sugestões dos dados.
- Todos os delimitadores foram demonstrados, contornando-os no texto. A ferramenta abre uma opção para o usuário digitar o delimitador.
- Para usuários experientes, a ferramenta disponibiliza a ação de criação de tipos de dados atômicos. No quadro tipos de dados, a ferramenta disponibiliza a opção “Criação de Novos Tipos de Dados”.

Texto do Documento

```
html><head><title>Collections</title></head><body
bgcolor="#ffffff" text="#000000" link="#000000">
<a href="../../index.html"><img alt="dblp.uni-trier.de"
src="../../Logo.gif" border=0 height=60 width=170></a>
<h1>Collections</h1><hr>

<h2>2001</h2>
<ul>
<li><a href="../../indices/a-tree/d/Dittrich:Klaus_R=.html">Klaus
R. Dittrich</a>,
<a href="../../indices/a-tree/g/Geppert:Andreas.html">Andreas
Geppert</a>  (Eds.):
Component Database Systems. Morgan Kaufmann Publishers,
ISBN 1-55860-642-4 and dpunkt.verlag,
ISBN 3-932588-75-4,
2001 <br>
<a href="DittrichG01.html">Contents</a>
</ul>

<h2>2000</h2>
<ul>
<li><a href="../../indices/a-
tree/k/Kuper:Gabriel_M=.html">Gabriel M. Kuper</a>,
<a href="../../indices/a-tree/l/Libkin:Leonid.html">Leonid
Libkin</a>,
<a href="../../indices/a-tree/p/Paredaens:Jan.html">Jan
Paredaens</a>  (Eds.):
Constraint Databases.
Springer Verlag, 2000, 428pp., ISBN 3-540-66151-4<br>
<a href="../../conf/cdb/cdb2000.html">Contents</a>
</ul>

<h2>1999</h2>
<ul>
<li><a href="../../indices/a-
tree/p/Paton:Norman_W=.html">Norman W. Paton</a>  (Ed.):
Active Rules in Database Systems.
Springer-Verlag, New York, 1999,
ISBN 0-387-98529-8<br>
<a href="patton99.html">Contents</a>
<li><a href="../../indices/a-tree/d/Dogac:Asuman.html">Asuman
Dogac</a>,
<a href="../../indices/a-tree/=/=Ouml=zsu:M=_Tamer.html">M.
Tamer &Ouml;zsu</a>,
<a href="../../indices/a-
tree/u/Ulusoy:=Ouml=zg=uuml=r.html">&Ouml;zg&uuml;r
Ulusoy</a>  (Eds.):
Current Trends in Data Management Technology.
Idea Group Publishing, January 1999,
ISBN 1-878289-51-9<br>
```

Ontologia



Tipos de dados

Atômico

Palavra
Conjunto de Palavras
Nome Próprio
Conjunto Nomes

Próprios

Decimal
Inteiro

Tamanho Mínimo: _____
Tamanho Máximo: _____

Valor Mínimo: _____
Valor Máximo: _____

Valida Arquivo: _____

Complexo

Lista !! **V**azia !! **O**brigatoriedade de um Elemento
Tupla !! **O**rdenada !! **D**esordenada!!

1) Para todos os tipos informe delimitadores
   _À esquerda _À direita
   _União _Alternativo
   _Exato _ Coringa à esquerda _Coringa à direita
2) Para todos os tipos informe
   _Obrigatório _Não Obrigatório
3) Para tipos atômicos informe
   _Parâmetros de Consistência
4) Para tipos Lista Demonstre
   _Separador de Elementos _Primeiro Elemento
5) Para tipos tupla demonstre
   _Possíveis Variações

Regras de Extração

FIGURA 5.1 - Instante Inicial Interação para Montagem do Primeiro Exemplo

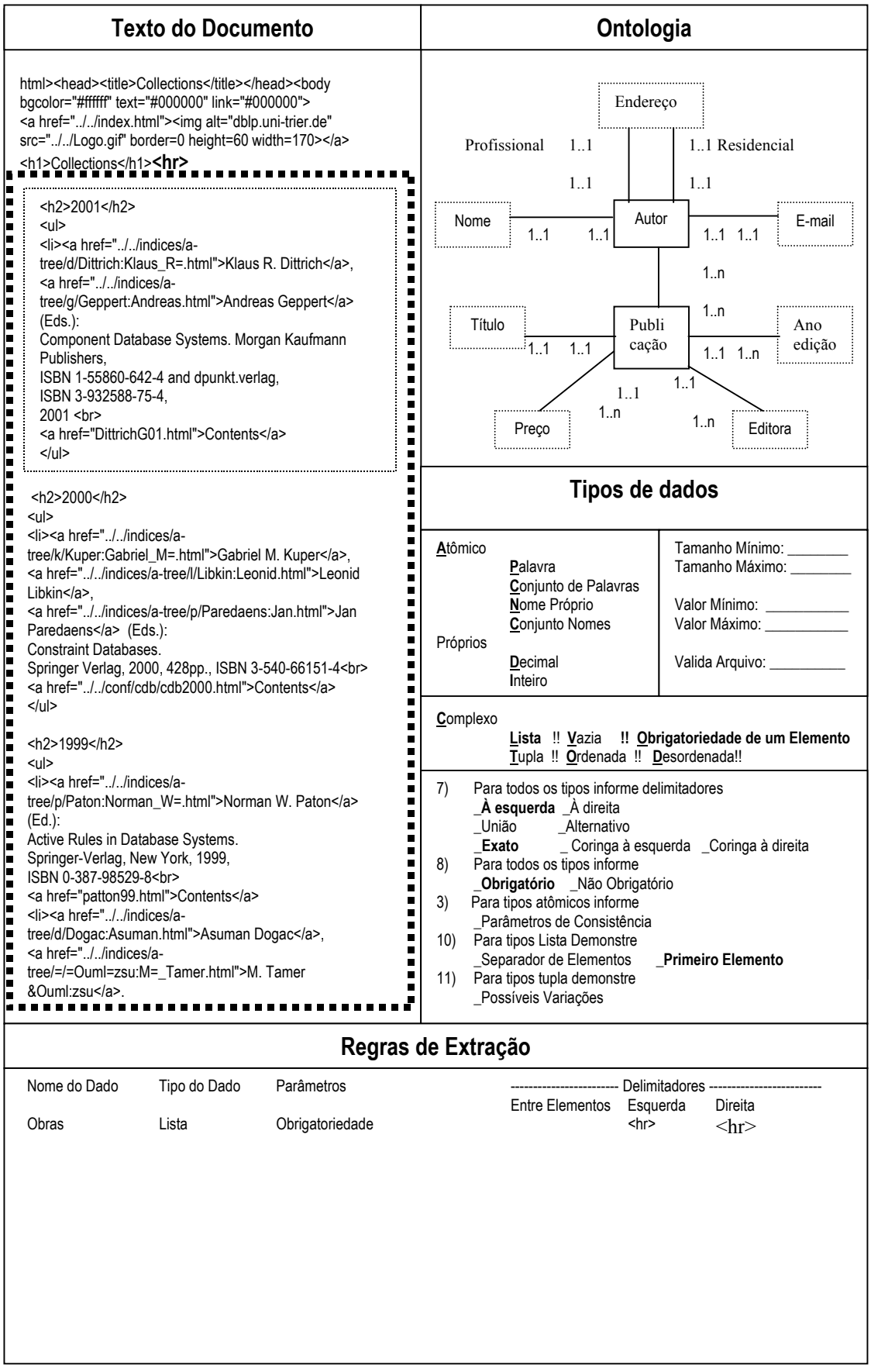
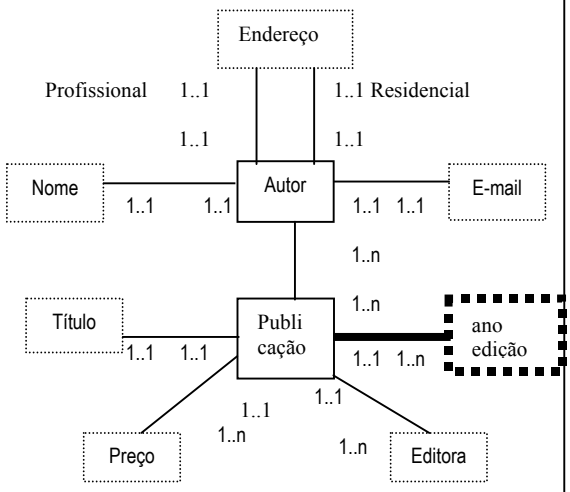


FIGURA 5.2 - Primeira Interação para Montagem do Primeiro Exemplo

Texto do Documento

```
html><head><title>Collections</title></head><body
bgcolor="#ffffff" text="#000000" link="#000000">
<a href="../../index.html"><img alt="dblp.uni-trier.de"
src="../../Logo.gif" border=0 height=60 width=170></a>
<h1>Collections</h1><hr>

<h2>  2001  </h2>
<ul>
<li><a href="../../indices/a-
tree/d/Dittrich:Klaus_R=.html">Klaus R. Dittrich</a>,
<a href="../../indices/a-
tree/g/Geppert:Andreas.html">Andreas Geppert</a>
(Eds.):
Component Database Systems. Morgan Kaufmann
Publishers,
ISBN 1-55860-642-4 and dpunkt.verlag,
ISBN 3-932588-75-4,
2001 <br>
<a href="DittrichG01.html">Contents</a>

 <h2>2000</h2>
<ul>
<li><a href="../../indices/a-
tree/k/Kuper:Gabriel_M=.html">Gabriel M. Kuper</a>,
<a href="../../indices/a-tree/l/Libkin:Leonid.html">Leonid
Libkin</a>,
<a href="../../indices/a-tree/p/Paredaens:Jan.html">Jan
Paredaens</a>  (Eds.):
Constraint Databases.
Springer Verlag, 2000, 428pp., ISBN 3-540-66151-4<br>
<a href="../../conf/cdb/cdb2000.html">Contents</a>
</ul>

<h2>1999</h2>
<ul>
<li><a href="../../indices/a-
tree/p/Paton:Norman_W=.html">Norman W. Paton</a>
(Ed.):
Active Rules in Database Systems.
Springer-Verlag, New York, 1999,
ISBN 0-387-98529-8<br>
<a href="patton99.html">Contents</a>
<li><a href="../../indices/a-
tree/d/Dogac:Asuman.html">Asuman Dogac</a>,
<a href="../../indices/a-tree/=/=Ouml=zsu:M=_Tamer.html">M.
Tamer &Ouml;zsu</a>,
<a href="../../indices/a-
```

Ontologia



Tipos de dados

Atômico
- **P**alavra
- **C**onjunto de Palavras
- **N**ome Próprio
- **C**onjunto Nomes

Próprios
- **D**ecimal
- **Inteiro**

Tamanho Mínimo: _4_____
Tamanho Máximo: _4_____

Valor Mínimo: _1500_____
Valor Máximo: _2999_____

Valida Arquivo: _____

Complexo
- L̲ista !! V̲azia !! O̲brigatoriedade de um Elemento
- **T**upla !! **O**rdenada !! **D**esordenada!!

9) Para todos os tipos informe delimitadores
_À esquerda _À direita
_União _Alternativo
_Exato _ Coringa à esquerda _Coringa à direita

10) Para todos os tipos informe
_Obrigatório _Não Obrigatório

3) Para tipos atômicos informe
_Parâmetros de Consistência

12) Para tipos Lista Demonstre
_Separador de Elementos _Primeiro Elemento

13) Para tipos tupla demonstre
_Possíveis Variações

Regras de Extração

Nome do Dado	Tipo do Dado	Parâmetros	Delimitadores: Entre Elementos	Esquerda	Direita
Obras	Lista	Obrigatoriedade		<hr>	<hr>
Anoedição	Inteiro	Obrig. / Tmi=4 / Tma=4 / Vmi=1500 / Vma= 2999		<h2>	</h2>

FIGURA 5.3 - Segunda Interação para Montagem do Primeiro Exemplo

Texto do Documento

html><head><title>Collections</title></head><body
bgcolor="#ffffff" text="#000000" link="#000000">
<img alt="dblp.uni-trier.de"
src="../../Logo.gif" border=0 height=60 width=170>
<h1>Collections</h1><hr>

<h2>2001</h2>

<ul>

Klaus R. Dittrich,
Andreas Geppert (Eds.):
Component Database Systems. Morgan Kaufmann Publishers,
ISBN 1-55860-642-4 and dpunkt.verlag,
ISBN 3-932588-75-4,
2001
Contents

**
**

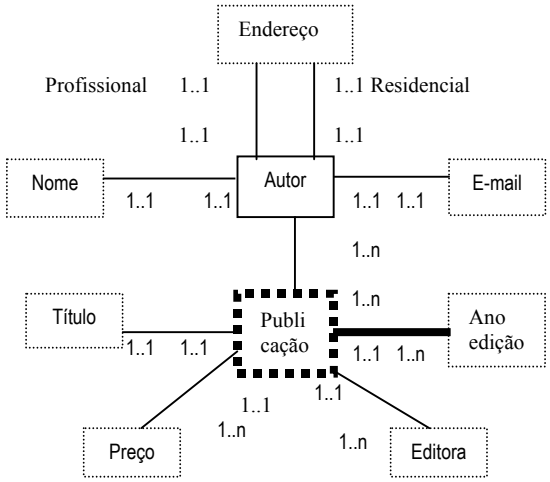
<h2>2000</h2>

Gabriel M. Kuper,
Leonid Libkin,
Jan Paredaens (Eds.):
Constraint Databases.
Springer Verlag, 2000, 428pp., ISBN 3-540-66151-4
Contents

<h2>1999</h2>

Norman W. Paton (Ed.):
Active Rules in Database Systems.
Springer-Verlag, New York, 1999,
ISBN 0-387-98529-8

Ontologia



Tipos de dados

Atômico
- **P**alavra
- **C**onjunto de Palavras
- **N**ome Próprio
- **C**onjunto Nomes

Próprios
- **D**ecimal
- Inteiro

Tamanho Mínimo: _____
Tamanho Máximo: _____

Valor Mínimo: _____
Valor Máximo: _____

Valida Arquivo: _____

Complexo
- **L**ista !! **V**azia **!! Obrigatoriedade de um Elemento**
- **T**upla !! **O**rdenada !! **D**esordenada!!

11) Para todos os tipos informe delimitadores
_À esquerda _À direita
_União _Alternativo
_Exato _ Coringa à esquerda _Coringa à direita
12) Para todos os tipos informe
_Obrigatório _Não Obrigatório
3) Para tipos atômicos informe
_Parâmetros de Consistência
14) Para tipos Lista Demonstre
_Separador de Elementos **_Primeiro Elemento**
15) Para tipos tupla demonstre
_Possíveis Variações

Regras de Extração

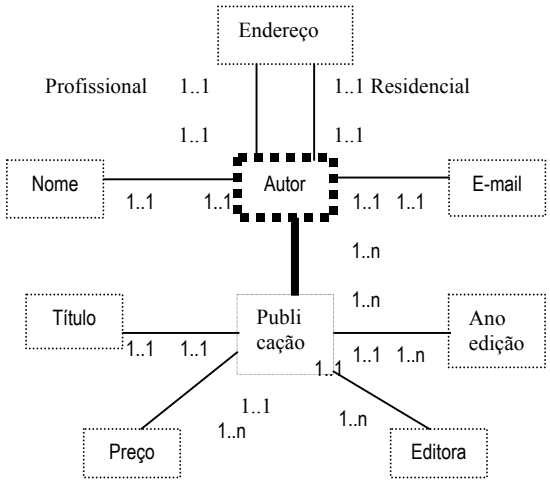
Nome do Dado	Tipo do Dado	Parâmetros	Delimitadores: Entre Elementos	Esquerda	Direita
Obras	Lista	Obrigatoriedade		<hr>	<hr>
Anoedição	Inteiro	Obrig. / Tmi=4 / Tma=4 / Vmi=1500 / Vma= 2999		<h2>	</h2>
Publicação	Lista	Obrigatoriedade			* *

FIGURA 5.4 - Terceira Interação para Montagem do Primeiro Exemplo

Texto do Documento

```
html><head><title>Collections</title></head><body
bgcolor="#ffffff" text="#000000" link="#000000">
<a href="../../index.html"><img alt="dblp.uni-trier.de"
src="../../Logo.gif" border=0 height=60 width=170></a>
<h1>Collections</h1><hr>

<h2>2001</h2>
<ul>
<li>
<a href="../../indices/a-
tree/d/Dittrich:Klaus_R=.html">Klaus R.
Dittrich</a>

,<a href="../../indices/a-
tree/d/Dittrich:Klaus_R=.html">Klaus R.
Dittrich</a>,

(Eds.):
Component Database Systems. Morgan
Kaufmann Publishers,
ISBN 1-55860-642-4 and dpunkt.verlag,
ISBN 3-932588-75-4,
2001 <br>
<a href="DittrichG01.html">Contents</a>

</ul>

<h2>2000</h2>
<ul>
<li><a href="../../indices/a-
tree/k/Kuper:Gabriel_M=.html">Gabriel M. Kuper</a>,
<a href="../../indices/a-tree/l/Libkin:Leonid.html">Leonid
Libkin</a>,
<a href="../../indices/a-tree/p/Paredaens:Jan.html">Jan
Paredaens</a> (Eds.):
Constraint Databases.
Springer Verlag, 2000, 428pp., ISBN 3-540-66151-4<br>
<a href="../../conf/cdb/cdb2000.html">Contents</a>
</ul>

<h2>1999</h2>
<ul>
```

Ontologia



Tipos de dados

Atômico
- **P**alavra
- **C**onjunto de Palavras
- **N**ome Próprio
- **C**onjunto Nomes

Próprios
- **D**ecimal
- **I**nteiro

Tamanho Mínimo: _____
Tamanho Máximo: _____

Valor Mínimo: _____
Valor Máximo: _____

Valida Arquivo: _____

Complexo
Lista !! **V**azia !! **O**brigatoriedade de um Elemento
Tupla !! **O**rdenada !! **D**esordenada!!

3) Para todos os tipos informe delimitadores
_À esquerda _À direita
_União **_Alternativo**
Exato ** Coringa à esquerda** _Coringa à direita
4) Para todos os tipos informe
_Obrigatório _Não Obrigatório
3) Para tipos atômicos informe
_Parâmetros de Consistência
6) Para tipos Lista Demonstre
_Separador de Elementos **_Primeiro Elemento**
7) Para tipos tupla demonstre
_Possíveis Variações

Regras de Extração

Nome do Dado	Tipo do Dado	Parâmetros	Delimitadores: Entre Elementos	Delimitadores: Esquerda	Delimitadores: Direita
Obras	Lista	Obrigatoriedade		<hr>	<hr>
Anoedição	Inteiro	Obrig. / Tmi=4 / Tma=4 / Vmi=1500 / Vma= 2999		<h2>	</h2>
Publicação	Lista	Obrigatoriedade			* *
Autor	Lista	Obrigatoriedade	,	*	(Eds.): ou (Ed.) ou (Hrsg.):

FIGURA 5.5 - Quarta Interação para Montagem do Primeiro Exemplo

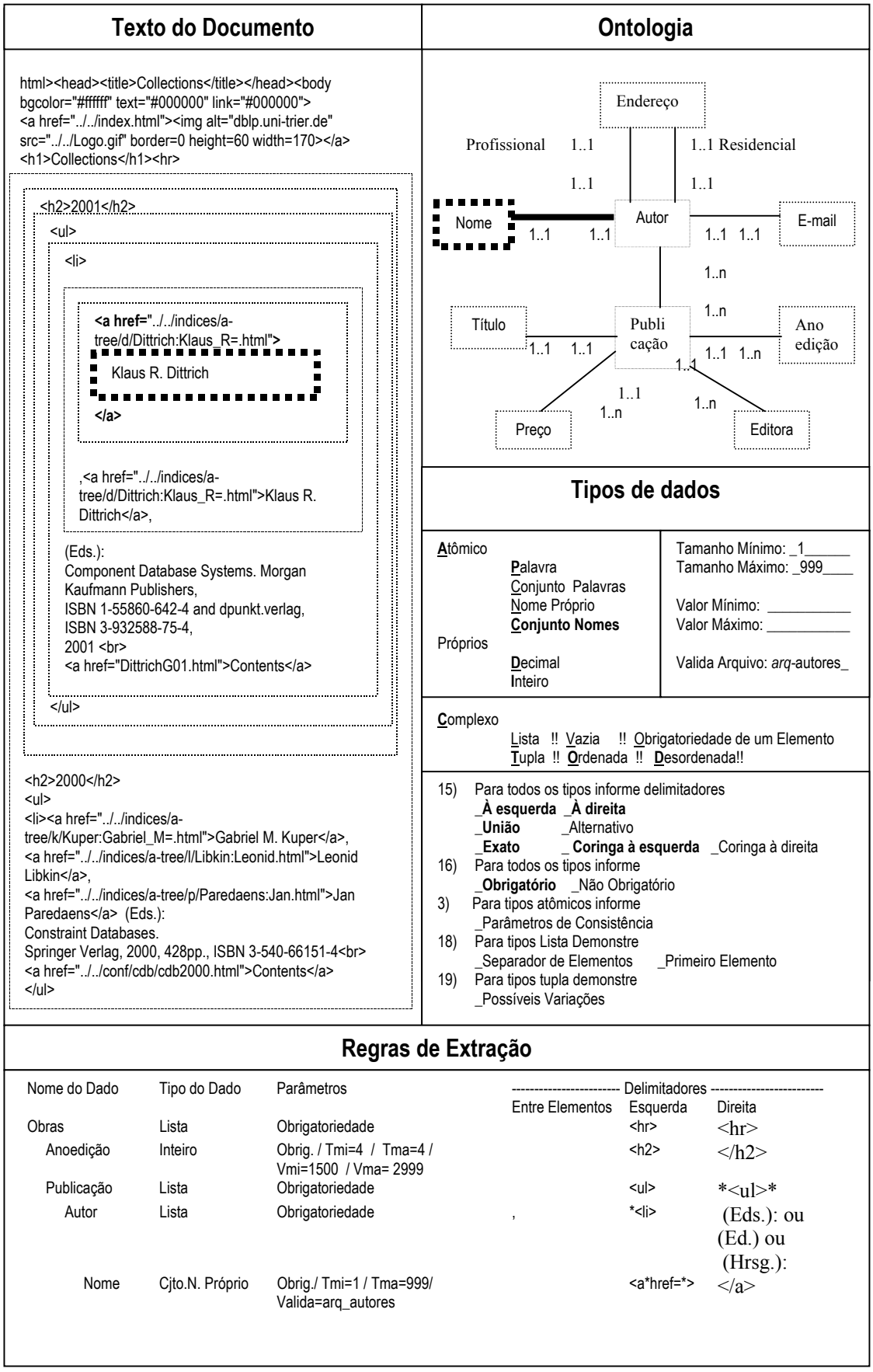


FIGURA 5.6 - Quinta Interação para Montagem do Primeiro Exemplo

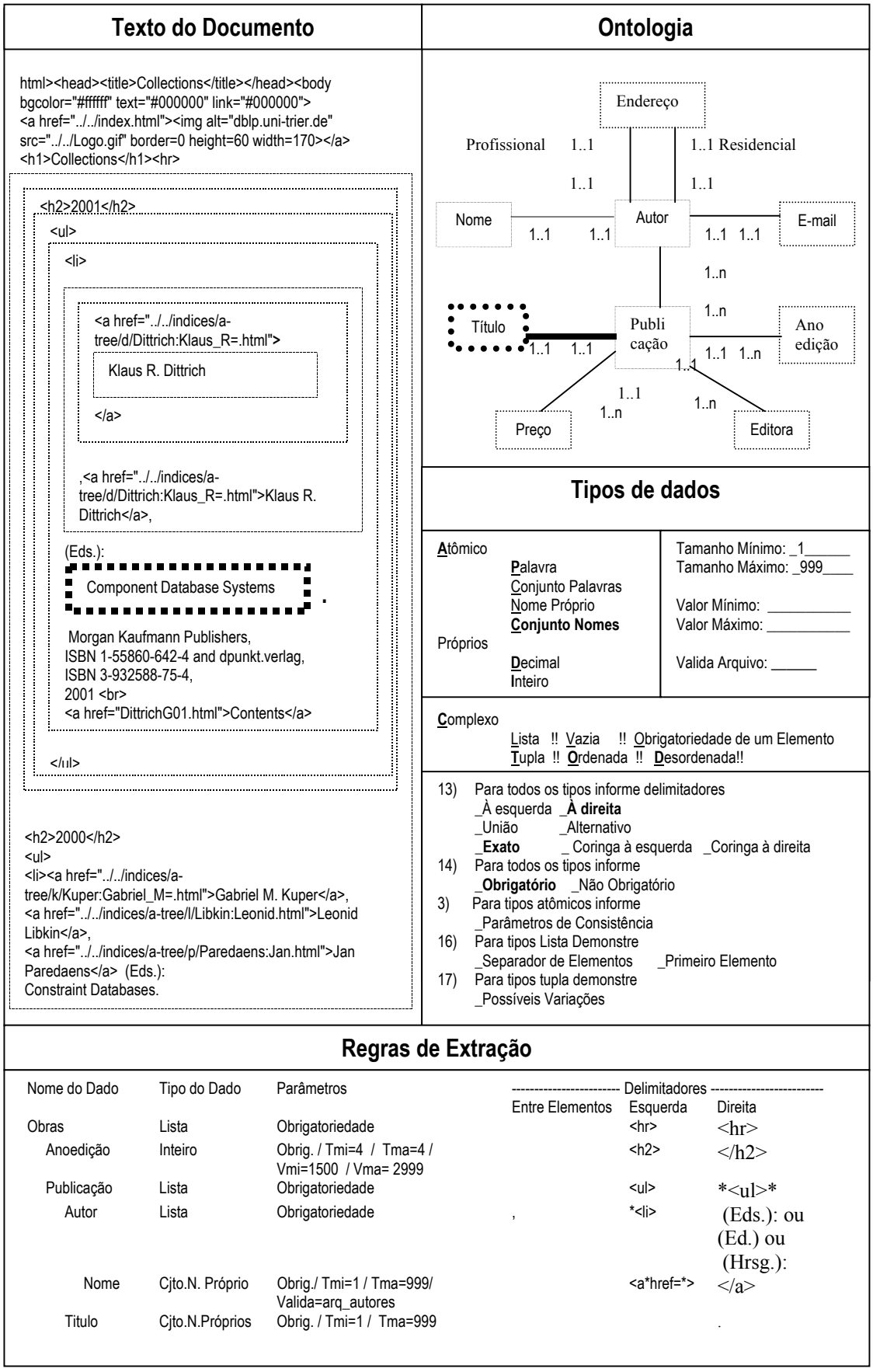


FIGURA 5.7 - Sexta Interação para Montagem do Primeiro Exemplo

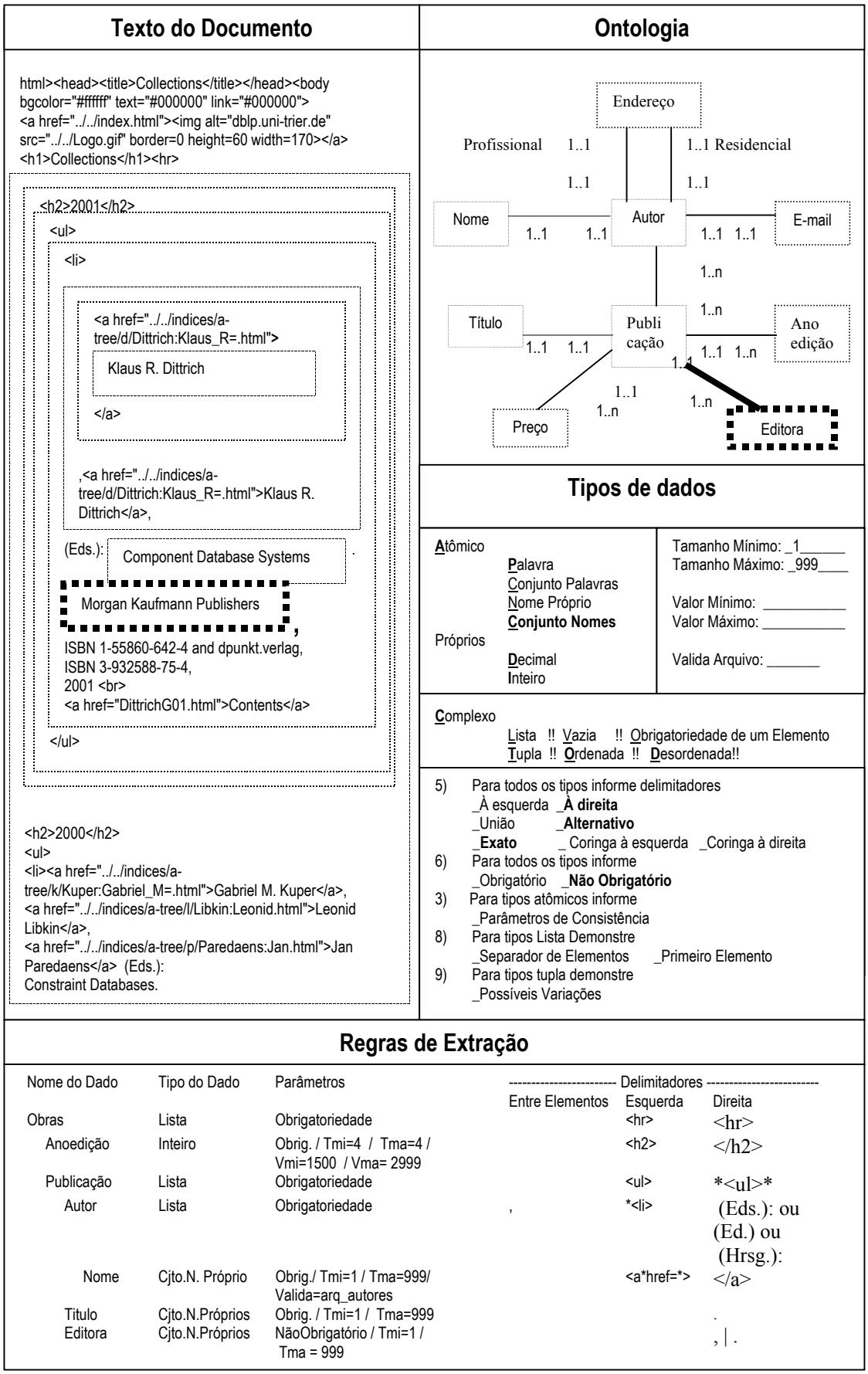


FIGURA 5.8 - Sétima Interação para Montagem do Primeiro Exemplo

6 Conclusões e Trabalhos Futuros

A principal contribuição desse trabalho é o desenvolvimento de um extrator semi-automático baseado em ontologias e na montagem de exemplos. A utilização de ontologias é um ponto em comum dessa proposta com o método de Embley [EMB99a]. Entretanto, o grande diferencial é a forma de recuperação da estrutura dos documentos, a qual utiliza a abordagem de exemplos. Enquanto Embley manipula documentos sem estrutura e sem delimitadores, o método apresentado trabalha sobre documentos com indicação de estrutura hierárquica. A existência de alguma estrutura implícita permite ao extrator aprender com algum exemplo e extrair as informações de forma automática de outros documentos semelhantes.

A ontologia é constituída pelo conhecimento inserido na modelagem dos objetos e por informações que são aprendidas durante o processo de extração. O modelo semântico identifica os objetos, seus relacionamentos, as restrições de integridade e os tipos de dados que a informação pode assumir. Com a evolução do processo de extração, a ontologia enriquece esse conhecimento com as estatísticas armazenadas.

A ontologia é o guia mestre do usuário na montagem do primeiro exemplo da classe de documentos. Essa etapa de construção das regras é a que exige maior cuidado, pois erros cometidos nesse momento vão refletir no processo automático de extração. Para que não se torne um processo manual, exaustivo e muito propenso a erros, a proposta é uma ferramenta visual guiada pela ontologia conforme esboçado na Seção 5.

Outro diferencial desse método é que as regras de extração montadas pelo usuário compõem uma gramática. A gramática é o resultado da etapa de montagem do primeiro exemplo e a principal entrada para o processo automático. Ela funciona como um espelho do documento, identificando a composição dos objetos, os delimitadores da informação, os quais funcionam como filtros, e as combinações possíveis de caracteres que a informação pode assumir. No início dessa pesquisa foram analisadas outras formas de representação, como expressões regulares. Mas foi somente com a utilização de gramáticas que se conseguiu o poder de expressão necessário para representar objetos aninhados e regras de extração alternativas para o mesmo objeto. As gramáticas livres de contexto podem apresentar problemas de processamento quando encontram produções ambíguas ou recursivas, mas essas questões foram tratadas pela ferramenta conforme explicado na Seção 3.

A partir da gramática, o extrator passa a operar de forma automática sobre a classe de documentos, gerando os resultados obtidos em linguagem XML, formato estruturado e institucionalizado como um padrão. Desta forma, a ferramenta permite uma interação com outras ferramentas de manipulação de dados em XML. Esta linguagem é amplamente difundida e possui várias linhas de pesquisa e ferramentas já concebidas para o manuseio de suas informações.

O método exige que os documentos pertencentes a uma mesma classe sejam semelhantes. As diferenças entre um documento e outro precisam ser exemplificadas pelo usuário. Assim, quando as diferenças forem expressivas, o usuário precisa demonstrar mais de um exemplo. Para que a montagem das regras fique restrita a um único exemplo, os documentos precisam ser bem estruturados, e possuir a mesma

estrutura hierárquica de objetos. Os objetos, por sua vez, precisam ser formados pela mesma combinação de caracteres e estarem acompanhados dos mesmos delimitadores.

Apesar do método ter sido baseado em documentos HTML, é importante salientar que ele não restringe os delimitadores às marcas de formatação (*tags),* como é o caso das ferramentas W4F [AZA99] e TSIMMIS [HAM98]. Conforme a estrutura do documento, poderiam ter sido utilizados como delimitadores outros conjuntos de caracteres.

Foi desenvolvido um protótipo e realizados vários experimentos. Os experimentos foram bem sucedidos, conseguindo extrair sempre 100% das informações desejadas, conforme demonstrado na Seção 4. Foi possível mostrar que a ferramenta reconhece não só dados atômicos, como também recompõem os dados complexos através da gramática. As experiências realizadas demonstraram que a mesma informação pode possuir diversas regras de extração, ou seja, possuir comportamentos e posições diferentes nos documentos da mesma classe. Também comprovaram que a mesma gramática pode ser utilizada por vários documentos diferentes, desde que pertençam à mesma classe de documentos.

Ao experimentar a ferramenta, foram utilizados vários documentos. Os tipos de dados oferecidos pela ferramenta atenderam todas as necessidades para definição dos tipos de dados das informações. Os tipos complexos lista e tupla foram suficientes para representar dados complexos, pois um caracteriza o conjunto de dados iguais e o outro conjunto de dados diferentes. Já os tipos atômicos possuem um mecanismo para que o usuário crie seus próprios tipos. Para isso, é necessário um conhecimento básico de expressões regulares. Tão importante quanto a especificação clara dos tipos de dados é o fato da ferramenta trabalhar com dados aninhados.

Ao desenvolver o protótipo foi utilizada a estratégia *top-down* para extração das informações. Ela foi escolhida entre muitos algoritmos para o tratamento de gramáticas, pela sua facilidade de compreensão, sem ser analisado o critério de performance.

Essa área de pesquisa está em uma etapa inicial. Muito se tem a estudar em extração de dados semi-estruturados. A proposta desse trabalho pode ter outros desdobramentos e ter continuidade em vários aspectos.

É preciso dar seqüência ao desenvolvimento do protótipo, implementando a fase de montagem do primeiro exemplo. Como é uma fase de interação com o usuário, é necessário um estudo aprofundado em técnicas de interfaces homem-computador. Com esse protótipo será possível validar a proposta demonstrada na Seção 5 desta dissertação e medir a facilidade de interação da ferramenta, principalmente na identificação dos delimitadores da informação.

Além desses aspectos, é preciso definir técnicas de ferramentas visuais capazes de exibir ontologias maiores. As ontologias podem utilizar thesaurus e dicionário de línguas para facilitar o processo de interação com o usuário.

Vários métodos de extração foram estudados antes da concepção deste (Seção 2). Poucos experimentos foram feitos com eles. Um trabalho futuro seria disponibilizar

todas as ferramentas e aplicá-las sobre os mesmos documentos para comparar resultados.

Alguns incrementos adicionais poderiam ser feitos à ferramenta, uma vez que opera sobre documentos da WWW. Ela pode ser enriquecida tratando também de aspectos como a busca de páginas na WWW e a extração indireta em páginas encadeadas (*links)* da página principal.

Diante da disseminação de dados semi-estruturados, é imprescindível que continuem as pesquisas sobre extração e manipulação desse tipo de dado. A extração de dados semi-estruturados é um assunto que interessa tanto as grandes corporações como indivíduos em particular.

Bibliografia

[ABI97] ABITEOUL, Serge. **Querying Semistructured Data**. In: INTERNATIONAL CONFERENCE ON DATABASE THEORY, DELPHI, 1997, Greece. Disponível em: <http://www-rocq.inria.fr/~abiteoul/pub/index.html>. Acesso em: março 2001.

[ABI97a] ABITEOUL, S.; GOLDMAN, R. et al. **Views for Semistructured Data**. In: INTERNATIONAL WORKSHOP ON MANAGEMENT OF SEMISTRUCTURED DATA, TUCSON, 1997. Disponível em: <http:/www.rocq.inria.fr/~abiteoul/pub/index.html>. Acesso em: março 2001.

[ABI2000] ABITEOUL, S.; BUNEMAN, P.; SUCIU, D. **Data on the Web**: From Relations to Semistructured Data and XML. [S.l.]: Morgan Kaufmann, 2000.

[ADE98] ADELBERG, B. NoDoSe – A Tool for Semi-Automatically Extracting Structured and Semistructured Data From Text Documents. **SIGMOD Record,** [S.l.], 1998. Disponível em: <http://www.cs.nwu.edu/~adelberg>. Acesso em: março 2001.

[ADE99] ADELBERG, B. **Building Robust Wrappers for Text Sources.** 1999. Technical Report. Disponível em: <http://www.cs.nwu.edu/~adelberg>. Acesso em: março 2001.

[AHO95] AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey . **Compiladores – Princípios, Técnicas e Ferramentas**. Rio de Janeiro: LTC Livros Técnicos e Científicos, 1995.

[ATZ98] ATZENI, Paolo; MECCA, Giansalvatore. Cut and Paste. **Journal of Computing and System Sciences**, 1998. Disponível em: <http://www.dia.uniroma3.it/Araneus>. Acesso em: março 2001.

[ATZ98a] ATZENI, Paolo; MECCA, Giansalvatore et al. **From Database to Web-Bases**: The Araneus Experience. [S.l.: s.n.], 1998. Technical Report. Disponível em: <http://www.dia.uniroma3.it/Araneus>. Acesso em: março 2001.

[ATZ98b] ATZENI, Paolo; MECCA, Giansalvatore et al. **The Araneus Web-Base Management System**. [S.l.: s.n.], 1998. Technical Report. Disponível em: <http://www.dia.uniroma3.it/Araneus>. Acesso em: março 2001.

[ATZ98c] ATZENI, Paolo; MECCA, Giansalvatore et al. **To Weave the Web**. [S.l.: s.n.], 1998. Technical Report. Disponível em: <http://www.poincare.inf.uniroma3.it>. Acesso em: março 2001.

[AZA99] AZAVANT, Fabien. W4F: A WysiWyg Web Wrapper Factory for Minute-Made Wrappers. In: INTERNATIONAL CONFERENCE ON COOPERATIVE INFORMATIVE SYSTEMS, COOPIS, 1999. Disponível em: <http:/db.cis/upenn.edu>. Acesso em: março 2001.

[AZA99a] AZAVANT, Fabien; SAHUGUET, Arnaud. Looking at the Web Through XML Glasses. In: INTERNATIONAL CONFERENCE ON COOPERATIVE INFORMATION SYSTEMS, COOPIS, 1999. Disponível em: <http:/db.cis/upenn.edu>. Acesso em: março 2001.

[AZA99b] AZAVANT, Fabien; SAHUGUET, Arnaud. **WysiWyg Web Wrapper Factory (W4F)**. [S.l.: s.n.], 1999. Technical Reports. Disponível em: <http:/db.cis/upenn.edu>. Acesso em: março 2001.

[BAE99] BAEZA, Ricardo; RIBEIRO, Berthier. **Modern Information Retrieval**.

[S.l.]: Addison Wesley, 1999. p. 257-323.

[BER99] BERGAMASCHI, S.; CASTANO, S.; VINCINI, M. et al. **Intelligent Techniques for the Extraction and Integration of Heterogeneous Information.** [S.l.: s.n], 1999. Technical Report. Disponível em: <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS>. Acesso em: março 2001.

[BUN97] BUNEMAN, Peter et al. Adding Structure to Unstructured Data. In: INTERNATIONAL CONFERENCE ON DATABASE THEORY, ICDT, 1997. **Database Theory**: proceedings. Berlin: Springer-Verlag, 1997.

[BUN97a] BUNEMAN, Peter. **Semistructured Data**. [S.l.: s.n.]: 1997. Technical Report.

[CHA94] CHAWATE, S.; GARCIA-MOLINA, H. et al. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In: THEN ANNIVERSARY MEETING OF THE INFORMATION PROCESSING SOCIETY OF JAPAN, 1994. **Proceedings...** [S.l.: s.n.], 1994.

[COR97] CORTES, Leslie. **Designing a Graphical User Interface**. In: MEDICAL COMPUTING TODAY, 1997. Disponível em: <http://www.medicalcomputingtoday.com>. Acesso em: março 2001.

[CRA99] CRANEFIELD, Stephen; PURVIS, Marin. **UML as an Ontology Modelling Language**. In: WORKSHOP ON INTELLIGENT INFORMATION INTEGRATION, IJCAI, 1999. Disponível em: <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18>. Acesso em: março 2001.

[CRE98] CRESCENZI, Valter; MECCA, Giansalvatore. Grammars Have Exceptions. **Information System**, 1998. Disponível em: <http://www.difa.unibas.it/users/gmecca>. Acesso em: março 2001.

[EMB92] EMBLEY, David W. et al. **Object-oriented System Analysis**: A Model-Driven Approach. Englewood Cliffs, New Jersey: Prentice Hall, 1992. Disponível em: <http://www.deg.byu.edu/papers>. Acesso em: março 2001.

[EMB98] EMBLEY, David et al. A Conceptual-Modelling Approach to Extracting Data from the Web. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, ER,17.,1998. **Conceptual Modeling**: proceedings. Berlin: Springer-Verlag, 1998.

[EMB98a] EMBLEY, David et al. **Ontology-Based Extraction and Structuring on Information from Data-Rich Unstructured Documents**. [S.l.: s.n.], 1998. Technical Report. Disponível em: <http://www.deg.byu.edu/papers>. Acesso em: março 2001.

[EMB99] EMBLEY, David W. et al. **Ontology Suitability for Uncertain Extraction of Information from Multi-Record Web Documents**. [S.l.: s.n.], 1999. Technical Report. Disponível em: <http://www.deg.byu.edu/papers>. Acesso em: março 2001.

[EMB99a] EMBLEY, David W. et al. Conceptual Model-based Data Extraction from Multi-Record Web Pages. **Data & Knowledge Engineering**, Amsterdan, v.31, 1999.

[EMB99b] EMBLEY, David W. et al. Record Boundary Discovery in Web Documents. **SIGMOD Record**, New York, v. 28, n. 2, June 1999. Trabalho apresentado na ACM SIGMOD International Conference on Management of Data, SIGMOD, 1999, Philadelphia.

[ERD99] ERDMANN, M.; STUDER, R. Ontologies as Conceptual Models for XML Documents. In: WORKSHOP ON KNOWLEDGE ACQUISITION, MODELING AND MANAGEMENT, KAW, 1999, Alberta, Canada. **Proceedings...** [S.l.: s.n.], 1999. Disponível em: <http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html>. Acesso em: março 2001.

[FLO99] FLORESCU, D. et al. XML-QL: A Query Language for XML. In: INTERNATIONAL WWW CONFERENCE, 1999, Toronto, Canadá. **Proceedings...** [S.l.: s.n.], 1999.

[GAR97] GARCIA-MOLINA, H. et al. The TSIMMIS Approach to Mediation: Data Models and Languages. **Journal of Intelligent Information System**, [S.l.], 1997. Disponível em: <http://www.db-stanford.edu/tsimmis/publications.html>. Acesso em: março 2001.

[GOL97] GOLDMAN, R.; WIDOM, J. Dataguides: Enabling Query Formulation and Optimization in Semistructured Databases. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES, 1997, Athens, Greece. **Proceedings...** [S.l.: s.n.], 1997. Disponível em <http://www-db.stanford.edu/lore/pubs>. Acesso em: março 2001.

[GOL98] GOLDFARB, Charles. **The XML Handbook**. [S.l.]: Prentice-Hall, 1998.

[GOL99] GOLDMAN, Roy; WIDOM, Jennifer. **Approximate DataGuides**. [S.l.: s.n.], 1999. Technical Report. Disponível em: <http://www.rodin.inria.fr/external/ssd99/workshop.html>. Acesso em: março 2001.

[GRU2000] GRUBBER, Tom. **Stanford KSL Ontology Editor**. [S.l.: s.n.], 2000. Technical Report. Disponível em: <http:/www-ksl-svc.stanford.edu>. Acesso em: março 2001.

[GUA94] GUARINO, Nicola. The Ontological Level. In: WITTGENSTEIN SYMPOSIUM, 4., 1993, Kirchberg, Austria. **Philosophy and the Cognitive Sciences:** proceedings. Vienna: Hölder-Pichler-Tempsky 1994. Disponível em: <http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html>. Acesso em: março 2001.

[GUA95] GUARINO, Nicola. Formal Ontology, Conceptual Analysis and Knowledge Representation. **Journal of Human and Computer Studies**, [S.l.], v. 43, n. 5/6, 1995. Disponível em: <http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html>. Acesso em: março 2001.

[GUA97] GUARINO, Nicola. Understanding, Building and Using Ontologies. **International Journal of Human and Computer Studies**, [S.l.], v. 46, n. 2/3, 1997. Disponível em: <http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html> Acesso em: março 2001.

[GUA2000] GUARINO, Nicola. A Formal Ontology of Properties (Preliminary Version). In: WORKSHOP ON APPLICATIONS OF ONTOLOGIES AND PROBLEM-SOLVING METHODS, ECAI, 2000. **Proceedings...** [S.l.: s.n.], 2000. Disponível em: <http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html>. Acesso em: março 2001.

[HAM95] HAMMER, Joachim; GARCIA-MOLINA, Hector et al. Information

Translation, Mediation, and Mosaic-Bases Browsing in the TSIMMIS System. In: INTERNATIONAL CONFERENCE OF MANAGEMENT, 1995, San Jose, Califórnia. **Proceedings...** [S.l.: s.n.], 1995. Disponível em: <http://www-db.stanford.edu>. Acesso em: março 2001.

[HAM98] HAMMER, J.; GARCIA-MOLINA, J; ARANHA, R. et al. **Extracting Semistructured Information from the Web**. [S.l.: s.n.], 1998. Technical Report.

[HOL01] HOLZNER, Steven. **Desvendando XML**. [S.l.]: Campus, 2001.

[HUC98] HUCK, Gerald et al. Jedi: Extracting and Synthesizing Information from the Web. In: INTERNATIONAL CONFERENCE ON COOPERATIVE INFORMATION SYSTEMS, COOPIS, 1998, New York. **Proceedings...** [S.l.: s.n.], 1998.

[LAE99] LAENDER, Alberto; SILVA, Elaine; SILVA, Altigran. DEByE – Uma Ferramenta para Extração de Dados Semi-Estruturados. In: SIMPOSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 1999, Florianópolis. **Anais...** Florianópolis: UFSC, 1999.

[LAE99a] LAENDER, Alberto; SILVA, Elaine; SILVA, Altigran. **Top-Down Extraction of Semistructured Data**. Belo Horizonte: [s.n.], 1999. Technical Report.

[LAE2000] LAENDER, Alberto; SILVA, Elaine; SILVA, Altigran. **Extracting Semi-strucutred Data Through Examples**. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 2000, João Pessoa. **Anais...** João Pessoa: PUCRS, 2000.

[LIN99] LIN, Xia et al. **Visual Interactions with Web Database Content**. [S.l.: s.n.], 1999. Technical Report. Disponível em: <http://www.medicalcomputingtoday.com>. Acesso em: março 2001.

[LOH99] LOH, Stanley; WIVES, Leandro; FRAINER, Antônio. Recuperação Semântica de Documentos Textuais na Internet. In: CONFERÊNCIA LATINOAMERICANA DE INFORMATICA, 1999, Assuncion, Paraguay. **Memórias...** Assuncion: Universidad Autonoma de Assuncion, 1999.

[LOP99] LÓPEZ, Fernández. Overview of Methodologies for Bulding Ontologies. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1999. **Proceedings...** [S.l.: s.n.], 1999.Disponível em: <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/vol-18>. Acesso em: março 2001.

[MCH98] McHUGH, Jason; ABITEOUL, Serge et al. Lore: A Databse Management System for Semistrucutred Data. **SIGMOD Record**, New York, v. 26, n. 3, p. 54-66, Sept. 1997. Disponível em: <http://www-db.stanford.edu/lore/pubs/index.html>. Acesso em: março 2001.

[MEC99] GRUMBACH, Stephane; MECCA, Giansalvatore. **In Search of the Lost Schema**. [S.l.: s.n.], 1999. Technical Report. Disponível em: <http://www.dia.uniroma3.it/araneus>. Acesso em: março 2001.

[MEC99a] ATZENI, Paolo; MECCA, Giansalvatore; MERIALDO, Paolo. **To Weave the Web**. [S.l.: s.n.], 1999. Technical Report. Disponível em: <http://www.dia.uniroma3.it/araneus>. Acesso em: março 2001.

[MEN2000] MENEZES, Paulo Blauth . **Linguagens Formais e Autômatos**. 3. ed. Porto Alegre: Instituto de Informática, 2000.

[NOD99] HOME Page do Projeto NoDoSe. Disponível em: <http://www.cs.nwu.edu/~adelberg>. Acesso em: março 2001.

[PAP95] PAPAKONSTANTINOU, Y.; GARCIA-MOLINA, Hector; WIDOM, Jennifer. Object Exchange Across Heteregoneus Information Sources. In: INTERNATIONAL CONFERENCE ON DATA ENGINE, 1995. **Proceedings...** [S.l.: s.n.], 1995.

[PRI2000] PRICE, Ana Maria de Alencar; TOSCANI, Simão Sirineo. **Implementação de Linguagens de Programação**: Compiladores. Porto Alegre: Sagra Luzzato, 2000.

[SCH97] SCHEWARTZ, Randal; CHRISTIANSEN, Tom. **Learning Perl**. [S.l.]: O´Reilly & Associates, 1997.

[SHN99] SHNEIDERMAN, Bem. **Designing the User Interface:** Strategies for Effective Human-computer Interaction. Reading, MA: Addison-Wesley, 1997.

[SIL2001] SILVEIRA, I. C.; HEUSER, C.A. Extração Semântica de Dados Semi-estruturados Através de Exemplos e Ferramentas Visuais. In: CONFERÊNCIA LATINOAMERICANA DE INFORMÁTICA, 2001, Mérida, Venezuela. **Memórias...** [S.l.: s.n.], 2001.

[VER82] VERHEIJEN, G; VAN BEKKUM, J. NIAM : An Information Analysis Method. In: CONFERENCE ON COMPARATIVE REVIEW OF INFORMATION SYSTEM DESIGN METHOLOGIES, 1982, Noordwijkerhout. **Proceedings...** [S.l.: s.n.], 1982.

[W3C2000] HOME Page W3C, 2001. Disponível em: <http://www.w3.org. >. Acesso em: março 2001.

[WIE92] WIEDERHOLD, Gio. Mediator in the Architeture of Future Information System. **IEEE Computer Magazine**, New York, Mar. 1992.

[YOU99] YOUNG, Peter. **Three Dimensional Information Visualization.** [S.l.: s.n.], 1999. Technical Report. Disponível em: <http://www.dur.ac.uk/~dcs3py>. Acesso em: março 2001.