Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Mestrado em Ciência da Computação

Dissertação de Mestrado

Extração Resiliente de Dados RDF a partir de Fontes Dinâmicas em Linguagem de Marcação

Por
Eduardo Edson Félix Mendonça
eduardo@lia.ufc.br

Fortaleza-CE. 2003

Orientador: Dr. Javam de Castro Machado.

RESUMO

A Web foi originalmente concebida como meio de comunicação voltado para o uso direto por pessoas, que acessam os documentos desejados, que são então visualizados para leitura através de um cliente padrão. Pouco à pouco, têm surgido outros usos para a infra-estrutura de comunicação que se criou. O aparecimento de XML motivou a utilização da internet e do protocolo HTTP para troca de informações entre aplicações. Ultimamente, tem se buscado uma participação mais efetiva de agentes de software no cenário da Web, auxiliando as pessoas na realização de tarefas. Hoje, a ajuda dada por programas já é considerável. Ferramentas de busca, de importância fundamental na utilização da Web, são um bom exemplo. No entanto, o papel desempenhado por programas poderia ser muito maior.

Dados presentes na Web encontram-se, muitas vezes, em formatos que dificultam sua utilização de forma mais efetiva. Técnicas de extração de dados podem ser usadas para localizar e obter dados presentes em documentos HTML e XML obtidos na Web, e então disponibilizá-los em formato conveniente para manipulação posterior. Um fator complicador para processos de extração de dados a partir de fontes Web são as mudanças de formato que eventualmente ocorrem nas fontes, quando um site muda seu *lay-out*, ou o documento disponibilizado por uma fonte XML tem sua estrutura alterada. Para enfrentar esta dificuldade, são necessários técnicas de extração resilientes, isto é, que sejam capazes de se reconfigurar no caso de mudanças no formato disponibilizado pela fonte alvo da extração.

Soluções de extração de dados são necessárias quando os dados originais encontram-se em um formato que os programas que desejam utilizá-los não são capazes de entender diretamente, como é o caso da maior parte dos dados disponíveis na Web hoje. O formato RDF surgiu com o objetivo de tornar informações disponíveis de uma forma que aplicações em geral possam utilizar. RDF permite expressar dados que podem ser relacionados a conceitos expressos com o auxílio de outras tecnologias, como RDF *Schema* e OWL. Desta forma,

RDF expressa os fatos que são a base sobre a qual se constrói o que se tem chamado de Web Semântica.

Na arquitetura proposta para a Web Semântica, a participação de uma fonte de informação depende da disponibilização, pela mesma, dos seus dados em formato RDF. Esta exigência pode representar, em muitos casos, um obstáculo à participação de algumas fontes no conjunto da Web Semântica, seja pelo esforço exigido, pelo eventual impacto da mudança sobre funcionalidades existentes, ou pela fraca motivação decorrente da falta de benefícios diretos para o provedor da informação.

Este trabalho propõe o uso de técnicas de extração de dados para a disponibilização de dados RDF a partir de fontes da Web tradicional. Extração de dados é apresentada como opção fácil e de baixo impacto para geração de dados RDF a partir de conteúdo existente. Além disso, a abordagem apresentada traz resiliência para a tarefa de extração de dados, proporcionando uma real independência com relação aos formatos disponibilizados pelas fontes e possíveis mudanças que estes possam sofrer. Resiliência é obtida através da técnica proposta das reconsultas, que, em caso de falha, reconstrói a configuração que guia o processo de extração, com base no histórico de dados extraídos, e no novo formato exportado pela fonte. A partir da independência sintática atingida, e da geração de dados em formato RDF, é possível construir a interoperabilidade prometida pelas tecnologias da Web Semântica.

A abordagem apresentada é validada pela implementação de um par extrator e mantenedor, em linguagem Java. O desenvolvimento foi organizado na forma de um *framework* para construção de extratores e mantenedores, que pode ser reutilizado em outros trabalhos.

ABSTRACT

The Web was originally conceived as a communication medium for direct use by people, who access the desired documents, which are randerized for reading by a standard client. Little by little, other uses for the infrastructure have arised. The creation of XML motivated the use of the internet and the HTTP protocol for the exchange of information among applications. Recently, a more effective participation of computer programs in the Web scene has been pursued. The help given by programs today is already significant. Search engines, which are of fundamental importance in the use of the Web, are a good example. Nevertheless, the role of programs could be greater.

Web data is found, many times, in formats that are difficult to use in an effective manner. Data extraction techniques can be used to locate and obtain data found in textual sources, such as HTML and XML documents obtained from the Web, and then make them available in a convenient format for future manipulation. A complicating aspect for Web data extraction processes are the format changes that eventually occur in Web data sources, when a site changes its lay-out, or the document format exported by an XML source changes its structure. In order to face this difficulty, there is a need for resilient extraction techniques, which are able to reconfigure themselves in case of format changes in the target sources.

Data extraction solutions are necessary when the original data is found in a format which programs cannot directly understand, which is the case with Web data available today. The RDF format emerged with the objective of making information available in a way that applications in general can use. RDF allows the expression of data that can be related to concepts expressed with the help of other technologies, such as RDF *Schema* e OWL. This way, RDF expresses the facts that are the basis on top of which the so-called Semantic Web is built.

In the architecture proposed by the Semantic Web, the participation of a data source depends on the disposal, by the source, of its data in RDF format. This requirement may represent, in many cases, a drawback to the participation of

some sources in the Semantic Web, either because of the required effort, the eventual impact of change over existing functionality, or the weak motivation, due to the lack of direct benefits for the information provider.

This work proposes the use of data extraction techniques for the disposal of RDF data from sources in the traditional Web. Data extraction is presented as a easy and low-impact option for existing information sources. Beyond this, the approach presents resilience for the task of data extraction, providing a real independence from the formats exported by the sources, and eventual chances that these might present. Resilience is obtained through the proposed technique of requeries, which rebuilds the configuration that guides the extraction process, based on the history of extracted data, and the new format exported by the source. On top of the syntactic independence obtained, and the generation of RDF data, the interoperability promised by the Semantic Web technologies can be built.

The approach presented is validated by the implementation of a pair extractor and maintainer, in Java. The development was organized in the form of a framework for the construction of extractors and maintainers, which can be reused in other initiatives.

Agradecimentos

Ao Deus de todos, e ao Deus de cada um.

Aos meus pais, minha tia e meus irmãos, por todo o apoio em todas as horas, e pelo ambiente propício para o crescimento intelectual que conseguimos estabelecer em nossa casa.

À minha esposa, pela compreensão e incentivo nesse período de estudo, pela inspiração e alegria, e pela ajuda em detalhes de editoração do texto e na revisão da bibliografia.

Aos meus sogros, pela segunda família que são.

Ao prof. Javam, pela confiança depositada, pelas opiniões e revisões do trabalho, e, acima de tudo, pelo relacionamento de amizade que vai além das funções de professor e orientador.

A todos os meus amigos, por estarem lá, sem importar onde estejam, nem a distância.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 A Web tradicional

Temos cada vez mais informação disponível. Mais do que o fato das instituições isoladamente produzirem e armazenarem uma quantidade cada vez maior de dados, é a interligação via redes de comunicação o fator multiplicador desse crescimento. Não é de hoje que cresce rapidamente o número de fontes de informação on-line, como previsão do tempo, cotações das bolsas de valores, resultados de competições esportivas, listas telefônicas, e muitas outras. No entanto, só recentemente o acesso a esses recursos foi universalizado de forma consistente, principalmente pela adoção de padrões abertos de comunicação, sobre os quais se ergueu a internet.

A Web foi, como se sabe, a grande responsável pela disseminação da internet. Grande parte do sucesso da Web como espaço universal de informações se deve à simplicidade de sua arquitetura, baseada em três pilares de construção: o conceito de URLs (*Universal Resource Locators*), o protocolo HTTP, e a linguagem de marcação HTML. Nesta arquitetura simples, URLs provêem o mecanismo para endereçamento de recursos, enquanto o protocolo HTTP padroniza a forma de comunicação entre diferentes computadores. Fechando a estrutura original, HTML proporciona uma sintaxe simples para construção de documentos de hipertexto, que podem ser obtidos via HTTP a partir de suas origens endereçadas por URLs, e podem conter apontamentos para outros documentos, na forma de links, que, novamente, usam URLs para endereçamento. Ao clicar em um link, obtém-se o documento apontado pelo link. A simplicidade destes princípios de construção proporciona grande facilidade de publicação e acesso às informações, fundamental para a disseminação e

crescimento da Web, hoje estimada em mais de 3 bilhões de documentos [GOOG03].

Depois da Web ganhar massa crítica e importância como meio de difusão de informações entre seres humanos, o desafio passou a ser utilizar a estrutura que se criou para outras formas de comunicação. Dentro deste contexto, XML surgiu, inicialmente como um projeto destinado a endereçar as limitações de HTML para expressar informação mais estruturada. Extrapolando o objetivo inicial, XML rapidamente se firmou como solução para troca de dados eletrônicos estruturados entre aplicações. A fração de documentos da Web em formato XML é ainda muito pequena. No entanto, esse número tende a crescer bastante, principalmente devido à padronização de tecnologias como WebServices [BHM+03].

Os formatos HTML e XML são os mais comuns entre os documentos Web. Entretanto, a resposta de uma chamada HTTP a uma URL pode ter virtualmente qualquer formato, desde uma outra linguagem de marcação, como SGML, a um formato texto, como ASCII, TeX ou RTF, a formatos binários como PDF, e até formatos desconectados do conceito de texto, como imagens em formato GIF ou JPEG. Apesar do número de possibilidades, a maior parte da Web está na forma de documentos em linguagem de marcação, objeto de interesse deste trabalho.

1.2 Obtendo dados da Web

A facilidade de publicação e acesso a informações do modelo Web gerou o círculo virtuoso que a retroalimenta, promovendo seu crescimento. No entanto, este cenário trouxe consigo dificuldades para o uso efetivo das informações.

1.2.1 Encontrando um documento

O primeiro aspecto que dificulta o uso de informação da Web é o problema de encontrar informação relevante em meio a tudo que está disponível. Esta dificuldade é conseqüência natural do modelo de publicação de informações da Web, onde quem publica não garante a qualidade da informação e nem é obrigado a fornecer metadados sobre o documento publicado. Este modelo

desburocratizado torna fácil o processo de publicação, gerando assim grande volume de documentos publicados que, devido à ausência de medidores de qualidade e metadados que possibilitem classificação, se torna uma massa desorganizada que ofusca a informação realmente relevante.

Com o objetivo de minimizar este problema, surgiram as ferramentas de busca na Web, entre as quais se destaca o Google [BP98]. Estas tentam compensar as limitações do modelo de publicação através de funcionalidades de busca, disponibilizadas para usuários por meio de uma interface Web. Nesta interface, o usuário caracteristicamente informa palavras-chave que ele julga possuir alta probabilidade de estarem presentes em documentos relevantes para sua necessidade. A busca, obviamente, não é feita diretamente à Web, e sim a catálogos internos de informações previamente obtidas da Web e posteriormente preparadas, de modo a garantir a qualidade da busca e minimizar o tempo de resposta. Para tal, uma ferramenta de busca precisa realizar outras duas tarefas que dão suporte à funcionalidade de busca. A primeira é o rastreamento de documentos na Web (*webcrawling*), realizada por um programa que vasculha a Web[1], através da malha de links existente, armazenando, em um repositório local, representações dos documentos em formato conveniente. A outra tarefa é a indexação e ranqueamento da informação obtida. Os índices gerados permitem que as consultas por palavras-chave tenham boa performance de execução, enquanto que a informação de ranque traz precisão para as buscas, possibilitando que resultados mais relevantes ocupem as primeiras posições das listas de resposta.

A descoberta de documentos relevantes na Web e as ferramentas de busca despertaram o interesse natural da comunidade da área de recuperação de informações (*information retrieval*), como um problema dentro de seu contexto de trabalho [Gol97, PAD98, CRB98]. O objetivo primordial dos trabalhos da

[1] Tais programas, chamados comumente de *crawlers*, são agentes de software que percorrem a Web, coletando páginas novas ou atualizadas, para que estas sejam posteriormente indexadas pelo programa de indexação. *Crawlers* são chamados também de robôs, *spiders*, andarilhos, entre outras denominações. Apesar do nome, *crawlers* na verdade não se movem de uma máquina para outra, rodando localmente no servidor do mecanismo de busca, coletando documentos remotos via HTTP [BR99].

área é a recuperação de todos os documentos relevantes para uma consulta do usuário, juntamente com o menor número possível de documentos irrelevantes. Neste contexto, a menor unidade de informação com a qual se trabalha é o documento.

1.2.2 Usando informações contidas no documento

Este trabalho tem seu foco voltado para informações que estão contidas dentro de um documento, que fazem parte de seu conteúdo. Uma quantidade considerável de documentos da Web trazem dados sobre uma os várias entidades lógicas, como pessoas, cidades, lojas, livros, CDs, ações da bolsa, etc. Podemos dizer que em um documento deste tipo encontram-se um ou vários objetos – ou tuplas – e seus respectivos atributos. Classificamos estes casos como **documentos ricos em dados** [ECJ+99]. Um arquivo XML é, por característica, um documento rico em dados, da mesma forma que uma grande quantidade de documentos HTML presentes na Web.

Para fazer uso efetivo das informações, seria interessante obter, de forma caracterizada, estas unidades de informação presentes dentro de documentos ricos em dados. Para tal, é necessário utilizar técnicas para manipulação de documentos nos formatos apresentados. No caso de XML, devido ao maior grau de estruturação, existem diferentes técnicas para manipulação de alto nível, que facilitam o trabalho, minimizando o esforço e padronizando as soluções. Neste âmbito, podemos citar a manipulação através de *parsers* DOM [HHW+00] ou SAX [SAX03], o uso de transformações XSLT [Cla99] e consultas XQuery [CCF+03]. No caso de documentos HTML, não existem soluções padronizadas, uma vez que a obtenção dos dados não pode ser generalizada, exigindo normalmente a construção de uma solução específica para cada caso. As técnicas de obtenção de unidades de informação presentes em documentos textuais, sejam estas soluções específicas ou genéricas, estão dentro do conceito denominado extração de informações ou, mais comumente, **extração de dados** [FLM98]. De forma geral, processos de extração de dados realizam a localização e obtenção de unidades de dados a partir de entradas em formatos textuais, disponibilizando

esses dados como saída, em um formato especificado. Existem muitas linguagens e sistemas que auxiliam a extração de dados a partir de texto, incluindo Perl[2] e o largo campo dos *scanners* léxicos e geradores de parsers, como lex e yacc [LS01]. Em muitas das soluções, o uso de expressões regulares desempenha papel fundamental.

Técnicas de extração de dados possibilitam a obtenção de dados em formatos tradicionais, como, por exemplo, tuplas do modelo relacional, a partir de fontes Web. Um importante fator complicador são as mudanças nos formatos de publicação das fontes, que caracteristicamente ocorrem com elevada freqüência na Web. Para se ter idéia da relevância do problema, já foi estimado que 40% da Web mude todo mês [BR99].

1.3 A Web Semântica

Hoje em dia, o auxílio prestado por programas e máquinas no uso da Web já é considerável. Esta afirmação não leva em consideração as funcionalidades que provêem a infra-estrutura básica necessária para o funcionamento da Web, como o correto funcionamento de redes de comunicação, servidores Web e outros serviços, além do *browser* e infra-estrutura de conectividade local do usuário. Fora tudo isso, ainda sobram funcionalidades como a fornecida pelas ferramentas de busca. Apesar de não fazerem parte da infra-estrutura essencial da Web, é fácil reconhecer o enorme benefício que elas proporcionam, e fica difícil imaginar esse mundo virtual sem a presença delas.

Entretanto, a ajuda mecânica[3] que recebemos hoje é ínfima com relação à potencialidade dessa área. Computadores e programas poderiam auxiliar pessoas de forma mais efetiva, e e até mesmo automatizar a realização de algumas tarefas. No entanto, limitações no modelo atual da Web inviabilizam uma contribuição mais expressiva por parte de processos mecânicos, para estas finalidades.

[2] No acronismo original que formou o nome da linguagem, a letra “e” correspondia à palavra *extraction*.
[3] Neste trabalho, a palavra mecânico(a) será utilizada no sentido de algo auxiliado, realizado ou automatizado pelo uso de programas e computadores

Um fator limitante é a ausência quase completa de metadados acompanhando as informações disponíveis na Web. O caso mais grave é o das páginas HTML, que não trazem um modelo forte para o uso de metadados. O que existe são soluções paleativas, como a inclusão de *tags* META nos documentos. Estas podem ser usadas para incluir informações como assunto, autor, palavras-chave, entre outras, que podem ser utilizadas por ferramentas de busca, como dados para o algoritmo de indexação. Fora esse uso, a utilidade dessas informações é restrita. As *tags* META contemplam somente o nível do documento como um todo, e têm expressividade restrita a um conjunto fechado de conceitos.

XML trouxe uma evolução para este quadro, uma vez que, neste formato, cada unidade de informação é acompanhada de uma *tag* que a qualifica. Como as *tags* podem ser livremente criadas, estas tendem a ter nomes bastante representativos da informação que acompanham. Representativo, neste contexto, diz respeito somente à facilidade de inferência do significado por um ser humano. Para processos mecânicos, como ferramentas de busca, estas informações não são mais representativas do que palavras soltas dentro do texto. Nomes de *tags* não podem ser associados a conceitos por um processo mecânico, devido ao problema de diferenças de vocabulário. Na modelagem de arquivos XML, autores diferentes podem usar nomes de *tags* diferentes para unidades de informação correspondentes ao mesmo conceito. Desta forma, a evolução trazida por XML é limitada, fato que é muitas vezes não compreendido, o que faz com que XML seja considerado erroneamente por muitos uma solução definitiva para todos os problemas de interoperabilidade semântica.

A iniciativa de inclusão de metadados em documentos da Web não é recente. O primeiro movimento neste sentido foi o desenvolvimento da PICS (*Platform for Internet Content Selection*) [RM96], uma reação do W3C [W3C03] à ameaça de criação de controles de censura na Web, através de um ato, votado no congresso americano em 1996, que regulava o conteúdo da internet. Graças à mobilização da comunidade internet, e da disponibilização rápida da PICS, o ato foi derrubado como inconstitucional. A plataforma traz a possibilidade de seleção de conteúdo pelo usuário, com base em rótulos fornecidos pelo provedor original

da informação, ou por terceiros, que expressam afirmações acerca das qualidades do recurso em questão, como endosso, recomendação, termos e condições de uso, entre outros.

Posteriormente, a concepção do rótulo PICS foi generalizada para um modelo de informação como grafos direcionais rotulados[4], batizado de modelo RDF (*Resource Description Framework*) [LS99]. O objetivo foi o de ampliar o modelo para a definição de meta-dados conceituais, que se destinam a tornar informações disponíveis para aplicações diferentes daquelas para as quais estas informações foram originalmente criadas. A base do modelo RDF é a definição de propriedades sobre recursos e valores para estas propriedades. Um recurso é qualquer coisa que possa ser apontada por uma URL. O valor de uma propriedade pode ser um outro recurso, caso em que a propriedade conecta dois recursos. Uma propriedade pode ser ela própria um recurso, sobre o qual podem ser atribuídas outras propriedades. Seguindo a filosofia da internet, qualquer um pode dizer qualquer coisa sobre qualquer coisa, isto é, qualquer um pode definir propriedades sobre recursos acessíveis na Web, seus ou de terceiros.

Para facilitar a definição de metadados, RDF suporta um sistema de classes, no estilo de classes do modelo orientado a objetos, que são organizadas em coleções chamadas esquemas. A definição de classes é a forma de estabelecer conceitos no modelo RDF, que podem ser utilizados para classificar recursos. Através de mecanismo similar à herança, é possível especializar conceitos existentes. Um esquema pode ser definido em RDF, mas pode ser interessante utilizar um vocabulário de definição mais rico, como RDF *Schema* [BG03], que traz um conjunto de classes e propriedades para descrição de um esquema RDF. Com a expressividade de RDF *Schema* pode-se, por exemplo, restringir as classes às quais uma propriedade se aplica, e que tipos são aceitos como valor, entre tipos literais e classes.

Apesar de simples, RDF *Schema* é uma linguagem de definição de ontologias. Uma ontologia é uma conceitualização de um domínio, sendo portanto composta de conceitos e seus interrelacionamentos. As chamadas linguagens de ontologia

voltadas para a Web são, em sua maioria, extensões de RDF *Schema* com primitivas de modelagem mais ricas. Recentemente, as duas principais frentes de desenvolvimento de linguagens para definição de ontologias, no caso a frente européia OIL [FH99] e a americana DAML [DAML00] uniram esforços, criando o que se chamou DAML+OIL [HHPS01]. Mais recentemente ainda, a linguagem DAML+OIL foi tomada como base para um esforço de padronização de uma linguagem de definição de ontologias dirigido pelo W3C, que se chama OWL (*Web Ontology Language*) [PHH03]. Estes quatro exemplos de linguagens de definição de ontologias são todos derivados de RDF, agregando maior expressidade àquela já existente. Usando OWL, é possível criar relacionamentos entre classes mais especializados do que a relação de subclasse, como equivalência e disjunção. Quanto às propriedades, pode-se expressar características como simetria, e relacionamentos entre propriedades, como o conceito de propriedades inversas.

Com toda essa infra-estrutura de meta-dados que estará disponível quando a Web Semântica ganhar massa crítica, processos mecânicos terão ao seu dispor informações, disponibilizadas de forma padrão, que possibilitarão a inferência de novos relacionamentos entre dados, que podem ser usados em cenários de integração de informações e automação de serviços, entre outros.

O modelo da Web Semântica tem como alicerce uma massa de dados em formato RDF disponibilizada pelos provedores de informação. No entanto, a disponibilização de dados em formato RDF por parte dos provedores das informações, em muitos cenários, não é simples, e pode, em alguns casos, nunca acontecer.

1.4 Contextualização, Objetivos e Motivação

A disponibilização de dados em formato RDF, pelas diversas fontes de informação, é a base na qual espera-se que a nova geração da Web seja, pouco à

[4] Grafos direcionais rotulados são grafos cujos arcos são pares ordenados, isto é, possuem um sentido associado, e no qual rótulos sáo associados aos vértices.

pouco, construída. Segundo Tim Berners Lee[5], o primeiro passo para a Web Semântica é "...colocar dados na Web em uma forma que máquinas possam entender naturalmente, ou **convertê-la para essa forma**" [BLF00].

Nesta visão compartilhada pelo W3C, a disponibilização de dados RDF será facilitada por "software de prateleira para a escrita de páginas da Web Semântica" [BLHL01], que incluirão dados RDF automaticamente, durante o processo de construção de documentos. Quando tais ferramentas existirem, é possível que boa parte do conteúdo novo que seja incluído na Web já traga consigo dados RDF de qualidade. Outra coisa que deve acontecer no futuro é a adoção em larga escala do formato RDF para troca de dados eletrônicos estruturados entre aplicações, cenário hoje parcialmente atendido por XML, não necessariamente compatível com RDF. À medida que a massa crítica da Web semântica for divulgando o formato, novas aplicações tenderão a usar RDF para suas necessidades de interação com outras aplicações.

No entanto, a Web Semântica não será construída somente por novas aplicações. Obviamente, o conteúdo existente na Web hoje deve, de alguma forma, se tornar disponível para as aplicações da próxima geração. Afinal, a Web Semântica deve ser, como foi dito, uma "extensão da Web atual" [BLHL01]. Assim sendo, o conteúdo da Web tradicional deve estar presente de forma representativa. Uma possível abordagem seria a conversão de fontes da Web tradicional. No caso de páginas HTML, a disponibilização dos dados se dá via anotações incluídas no conteúdo. Para fontes XML, o caminho indicado seria a reestruturação para um formato compatível com a especificação RDF. Em ambos os casos, um esforço considerável é necessário, enquanto, em muitos cenários, o benefício direto para o provedor da informação é duvidoso, não gerando motivação suficiente. Com isso, corre-se o risco de que muitas fontes de informação ricas em dados fiquem fora da chamada Web Semântica.

Há necessidade, portanto, de soluções para geração de dados RDF a partir de conteúdo existente, na forma em que ele se encontra. Este trabalho apresenta uma

[5] Tim Berners Lee é o inventor da Web, tendo escrito o primeiro cliente Web (que era, além de *browser*, editor), o primeiro servidor Web, juntamente com a maior parte do software de comunicação, definindo URLs, HTTP e HTML.

solução para obtenção de dados RDF a partir de fontes textuais dinâmicas de linguagem de marcação, como páginas HTML e documentos XML. A solução é baseada em técnicas de extração de dados, através das quais os dados RDF serão obtidos a partir do conteúdo original das fontes, que não necessitam alterar seus formatos de publicação. O processo de extração funciona, portanto, como forma de vencer a barreira sintática para entrada de fontes existentes na Web Semântica. O uso de técnicas de extração desfaz a dependência da disponibilização explícita de dados RDF por parte das fontes, permitindo, inclusive, que dados de uma fonte sejam disponibilizados à sua revelia, por terceiros. Desta forma, a extração de dados tem grande contribuição a dar no processo de transição para a nova geração da Web, ajudando a gerar a massa crítica de dados necessária para viabilizar a Web Semântica.

Para participar de forma efetiva na infra-estrutura da Web Semântica, novos desafios são colocados para os processos de extração. Uma necessidade das mais importantes é a resiliência no processo de extração. Resiliência é a capacidade de se recuperar de mudanças [Mor91]. No caso da Web, mudanças nos formatos de publicação são uma característica do meio. Desta forma, é de grande interesse que processos de extração de dados de fontes Web sejam resilientes. Somente o uso de extração resiliente traz verdadeira independência sintática, sobre a qual se pode construir interoperabilidade semântica.

1.5 Organização da Dissertação

No capítulo 2, são apresentados os conceitos de fonte de informação textual, formatos de dados semi-estruturados, e extração de dados a partir de fontes semi-estruturadas, incluindo as dificuldades inerentes à tarefa. Em seguida é feito um apanhado de trabalhos que tratam de extração de dados a partir de fontes HTML e XML. Finalmente, é apresentada uma formalização do problema da extração de dados, para uso no restante do trabalho.

O capítulo 3 contextualiza as técnicas de extração de dados no âmbito da Web Semântica. Para isso, inicialmente é apresentado o conjunto de tecnologias, padronizadas pelo W3C, relacionadas à questão da interoperabilidade semântica

na Web. Em seguida, é discutido o papel de técnicas de extração de dados, em particular daquelas que tratam a questão da resiliência na extração, dentro deste contexto. Finalmente, são discutidos resumidamente, e comparados entre si, outros trabalhos que trazem funcionalidades voltadas à resiliência na extração.

No capítulo 4, é apresentada a abordagem para extração de dados resiliente proposta neste trabalho. O capítulo começa trazendo uma tentativa de formalização para os conceitos de extração de dados, e em seguida para os problemas específicos relacionados com a resiliência na extração. Em seguida, a técnica proposta pelo trabalho para extração resiliente é discutida em detalhes, explanando-se a forma utilizada para a resolução dos problemas específicos da verificação e manutenção da extração.

O capítulo 5 discute a implementação da abordagem proposta. É apresentado um *framework* para desenvolvimento de extratores e mantenedores em linguagem Java. O *framework* proposto é utilizado para a construção de uma solução específica completa, que utiliza expressões regulares. São apresentados algorimos para solução dos principais problemas relacionados à extração e manutenção da extração. Finalmente, é apresentada uma análise de desempenho da técnica utilizada.

Capítulo 2

Extração de Dados de Fontes Textuais Dinâmicas

2.1 Introdução: Fontes de Informação Textuais

Fontes de informação são sistemas que respondem a consultas, retornando uma resposta apropriada para cada consulta submetida. No caso de fontes de informação textuais, a resposta é montada em um formato textual arbitrário, normalmente a partir dos dados de alguma fonte de dados interna. No entanto, iremos nos abstrair da arquitetura das fontes. Veremos as fontes textuais como caixas-pretas, que recebem consultas e retornam respostas em um formato textual qualquer.

São exemplos de fontes de informação em formato textual aquelas acessíveis via Web, que podem ser manipuladas seguindo o paradigma *request-response*, implementado pelo protocolo HTTP. Como formato textual, estas fontes utilizam normalmente formato HTML [RHJ99] ou XML [BPSM00]. Apesar de um *response* HTTP poder carregar qualquer texto, formatado ou não, nosso interesse se concentra em fontes que retornam documentos nesses dois formatos mais comuns.

HTML e XML são linguagens de marcação. Uma linguagem de marcação é um mecanismo para explicitar e identificar estruturas em um documento. A especificação XML define uma forma padrão de adicionar marcação a documentos. Em outras palavras, o autor de um documento XML pode ele mesmo definir os marcadores que utilizará em seus documentos, não estando preso a um conjunto fechado, como o conjunto de tags HTML. Em um documento XML, as *tags*, e a hierarquia estrutural entre elas, podem representar

o significado das informações. No entanto, quem determina estes significados é o autor do documento, e este só será conhecido por alguém que saiba que o documento respeita este determinado formato pré-estabelecido. No caso dos documentos HTML, as *tags* trazem somente informação de formatação estética do documento.

Documentos HTML, e principalmente XML, apresentam um nível de estruturação bastante superior, se comparados a outros formatos textuais. No entanto, essa estrutura não é rígida como em tabelas do modelo relacional, permitindo certo grau de variação estrutural. Devido a essa característica, HTML e XML apresentam facilidades para representação de dados semi-estruturados.

Para estabelecer o conceito de fonte de informação semi-estruturada, é preciso primeiramente dizer o que seria considerado uma fonte de informação totalmente estruturada. Primeiramente, tal fonte teria que retornar dados que apresentam sempre a mesma estrutura. Em outras palavras, os dados atômicos (atributos) que formam estruturas compostas (tuplas) vêm sempre na mesma ordem, nunca estão ausentes (podem trazer valor vazio, mas estão sempre presentes na estrutura) nem presentes em apenas algumas das estruturas.

Seguindo o conceito, uma fonte de dados semi-estruturada é aquela que apresenta, ou pode potencialmente apresentar, insconstância na estrutura das tuplas. Fontes de informação que retornam documentos HTML ou XML são, potencialmente, fontes de informação semi-estruturada. Neste trabalho, fontes de informação que retormam documentos em linguagem de marcação, como resposta a requisições, são denominadas **fontes de dados textuais dinâmicas em linguagem de marcação**. No caso particular em que o esquema de requisições e respostas é realizado via protocolo HTTP, usamos a demoninação **fonte Web**[6]. O interesse deste trabalho concentra-se especificamente nas fontes Web e os dados contidos em documentos retornados pelas mesmas.

Para fazermos uso mais efetivo das informações contidas nos documentos retornados por fontes Web, normalmente torna-se necessário isolar essas informações do resto do documento, e reorganizá-la de uma forma que possibilite

seu uso de forma flexível, como por exemplo populando um banco de dados relacional com as informações obtidas. Necessitamos então de técnicas para localizar, coletar e organizar dados de interesse presentes em documentos semi-estruturados, descartando informações irrelevantes e outros fragmentos de texto. Em outras palavras, necessitamos de técnicas de **extração de dados**.

Técnicas de extração de dados de fontes textuais viabilizam novas utilizações para as informações obtidas, que não eram possíveis a partir das fontes diretamente. Após a extração, a massa de dados obtida pode permitir a realização de consultas e cruzamento de dados que não eram possíveis através das interfaces de consulta pré-estabelecidas das fontes. A extração de dados é uma tecnologia capacitadora, isto é, que abre caminho para a utilização de outras tecnologias, como integração de dados e agentes inteligentes.

Quando se fala em obter informações na Web, existe um outro contexto distinto que precisa ser diferenciado, para evitar confusão. Pode-se estar falando na procura e localização de documentos relevantes a partir de uma consulta do usuário, que são retornados, de preferência, ordenados por relevância. Este é o contexto das ferramentas de busca na Web[7]. Um dos componentes destes sistemas é um programa, usualmente chamado de *webcrawler*, que automaticamente percorre a Web, catalogando e classificando URLs (que normalmente apontam para documentos HTML), guardando as informações obtidas em um repositório local. Um outro componente responde a consultas do usuário, utilizando informações do repositório para selecionar URLs provavelmente relevantes, a partir da entrada textual do usuário, que normalmente toma a forma de um conjunto de palavras-chave. Apesar de extremamente importante e interessante, a área de busca na Web tem relação

[6] Por simplicidade, este trabalho utiliza os termos **fonte de dados textuais dinâmicas em linguagem de marcação** e **fonte Web** como sinônimos.

[7] Bastante exploradas comercialmente, as técnicas para busca na Web não foram, inicialmente, publicadas como trabalhos científicos nem eram de domínio público. Isto só mudou com o desenvolvimento do Google, que também revolucionou no uso da função PageRank, baseada na quantidade de links na web que apontam para uma certa página, como informação auxiliar na determinação da relevância de um documento como resposta a uma dada consulta [BP98].

apenas periférica com o foco de interesse deste trabalho, que é a extração de dados.

2.2 Extração de Dados Semi-Estruturados

Uma consulta realizada a uma fontes de informação acessível na Web retornará, caracteristicamente, um documento HTML ou XML que, em meio a outras informações irrelevantes ou diretivas de apresentação visual, trará uma conjunto de uma ou mais tuplas[8] de interesse, compostas cada uma por atributos. O processo de extração tem por objetivo obter os dados de interesse a partir de um documento retornado, desprezando toda a informação irrelevante. Para fazer isso, o processo de extração tem que localizar os atributos e isolá-los do resto do texto irrelevante circunvizinho, e também relacionar os atributos pertencentes à mesma tupla.

O processo de extração de dados de uma fonte textual semi-estruturada é implementado por um programa de extração. A abordagem tradicional para extração de dados de fontes Web baseadas em HTML é a escrita de programas específicos para cada fonte, que ficaram conhecidos pelo termo ***wrapper*** [FLM98].

[8] No contexto deste trabalho, uma tupla é um conjunto de dados sobre uma determinada entidade. Desta forma, tupla é um conceito equivalente ao conceito de registro (ou tupla) do modelo de dados relacional, e ao conceito de objeto no modelo de dados orientado a objetos. O capítulo 4 traz uma definição formal para o conceito de tupla.

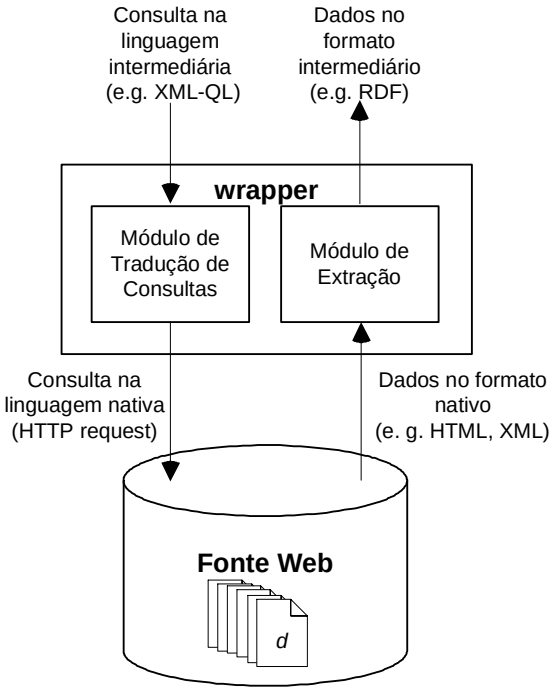


Figura 2.1 – Módulos de um *wrapper*

As tarefas de um *wrapper* são duas: traduzir consultas entregues pela aplicação cliente em requisições que possam ser feitas diretamente à fonte encapsulada; e converter resultados obtidos como resposta do formato entregue pela fonte encapsulada para um formato que a aplicação cliente possa tratar.

Neste trabalho, nosso interesse é a extração de dados presentes nas respostas obtidas das fontes, o que corresponde à tarefa de tradução de resultados. Esta é realizada pelo que chamamos de módulo de extração do *wrapper*.

O processo de extração de dados pode ser descrito como a seguir. Ao ser consultada, uma fonte de informação retorna um documento resposta (ou simplesmente resposta[9]). Um documento resposta contém dados de interesse, que correspondem a certos fragmentos de texto do documento resposta. Fora os dados de interesses, todo o restante do documento resposta é texto irrelevante, e será descartado durante o processo de extração. Os dados de interesse em uma página resposta podem corresponder a várias tuplas, cada uma com seus

[9] Os termos documento resposta, documento e resposta são usados alternadamente para expressar o mesmo conceito, no texto deste trabalho.

atributos. O processo de extração, além de obter os fragmentos correspondentes aos atributos, deve reconhecê-los como instâncias dos respectivos tipos de atributo, e reorganizá-los corretamente, agrupando os pertencentes à mesma tupla, em uma representação interna. Finalmente, as tuplas obtidas devem ser disponibilizadas para o cliente do processo de extração, codificadas em um formato conveniente para manipulação posterior.

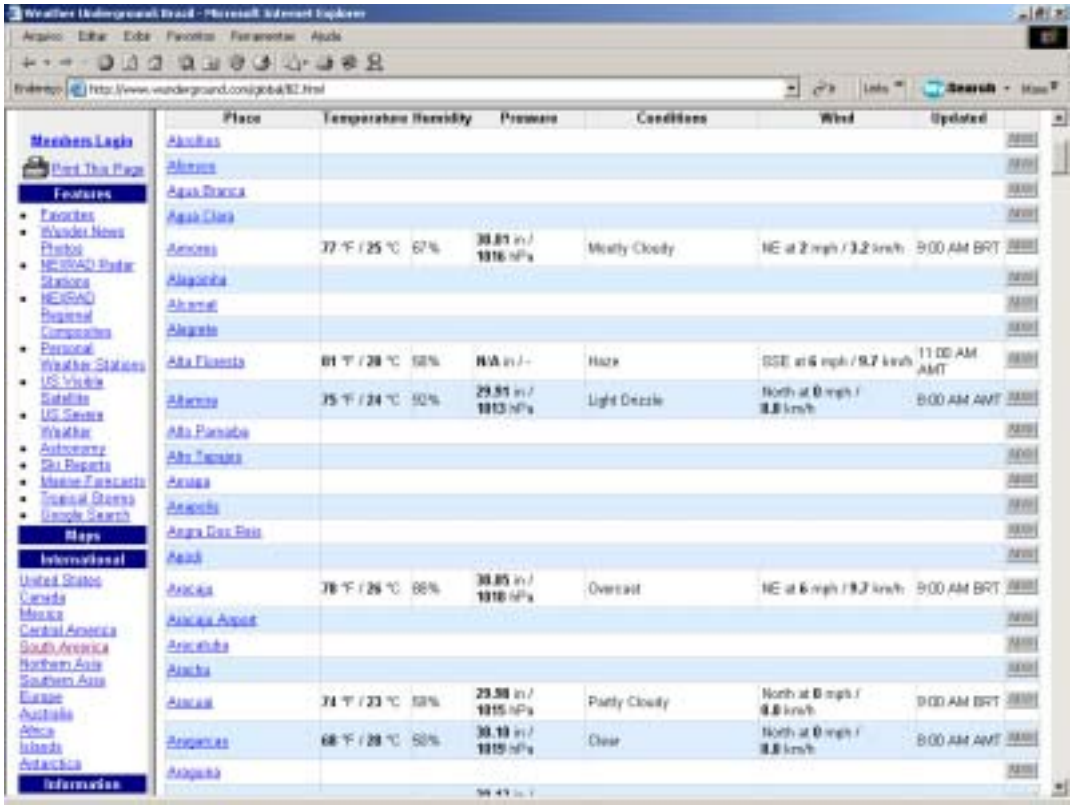


Figura 2.2 – Documento resposta HTML exibido no browser Internet Explorer

Para ilustrar, usaremos como exemplo o *site* http://www.wunderground.com, que traz informações sobre as condições do tempo em diferentes cidades do mundo. A página inicial do serviço traz um menu lateral com uma lista de países ou regiões do mundo. Ao clicarmos sobre um país, o *site* retorna uma lista com as principais cidades deste país, cada uma acompanhada das informações climáticas mais recentes de cada cidade, como temperatura, umidade relativa do ar, condição do céu, vento, etc.

Um clique sobre a o menu corresponde a uma consulta à fonte de dados. O link clicado determina o país cujas cidades aparecerão na resposta. Pode-se notar que um clique sobre um link completa a url base chamada com informação adicional que identifica o país. Assim, se a url for completada com /global/BZ.html, obteremos uma lista com informações de hora contendo cidades do Brasil. O complemento /global/BZ.html pode ser passado tanto clicando-se no link "Brazil", como escrevendo a url com o complemento diretamente no browser. Esta url contendo o complemento corresponde a uma consulta à fonte de dados em questão.

A resposta à nossa consulta é o documento HTML retornado pela chamada da url. Para o usuário comum, a resposta é aquilo que ele vê no browser: uma lista das cidades, com suas respectivas informações meteorológicas, dispostas uma em cada linha de uma tabela de fundo alternando entre o azul e o branco a cada linha, com os nomes das cidades na forma de links (normalmente texto sublinhado de cor azul), e as outras informações em fonte preta. Sabemos, no entanto, que a resposta é, na verdade, um documento textual codificado em uma linguagem de marcação, no caso HTML, a partir da qual o browser é capaz de gerar uma representação gráfica. De todo o conteúdo textual do documento resposta, estamos interessados somente na lista de cidades e suas informações. Este trecho de interesse é formado por uma seqüência de blocos de estrutura similar, como o que está ilustrado abaixo.

```
<TR bgColor=#ddeeff>
<TD><A
href="http://www.wunderground.com/global/stations/82398.html">Fortaleza</A>
                </TD>
                <TD><B>26</B> °C </TD>
                <TD>78%</TD>
                <TD><B>1015</B> hPa </TD>
                <TD>Scattered Clouds</TD>
                <TD vAlign=center>SSW at <B>3.2</B> km/h </TD>
                <TD>10:00 PM BRT</TD>
<TD><A
href="http://www.wunderground.com/php/editfavs.php?addfav=00000.82398"><IMG
                  height=16 src="Weather Underground Brazil_arquivos/ADD.gif"
                  width=30 border=0></A> </TD></TR>
```

Figura 2.3 – Trecho HTML contendo informações de interesse

É fácil notar a existência de uma tupla de dados presente no trecho acima, em meio a *tags* de marcação. Estamos interessados nos fragmentos de texto que correspondem aos dados atômicos que compõem as tuplas. Precisamos extraí-los, ou seja, isolá-los do resto da informação irrelevante e exportá-los, já novamente agrupados em tuplas, em um formato conveniente para os usos posteriores que estes dados terão.

O processo de extração terá que localizar esses dados atômicos de interesse, para poder extraí-los. Para tal, o programa poderá se guiar, por exemplo, por informações que determinam a posição dos dados atômicos na estrutura do documento. Outra possibilidade é utilizar informações sobre a vizinhança imediata de cada dado, isto é, guiar-se pelos fragmentos de texto que vêm antes e depois do dado de interesse.

```
<TD><B>26</B> °C </TD>
<TD>78%</TD>
<TD><B>1015</B> hPa </TD>
<TD>Scattered Clouds</TD>
```

Figura 2.4 – Vizinhanças dos dados, possíveis guias do processo de extração

Apesar da maior parte das informações disponíveis hoje através da Web estarem no formato HTML, processos de extração podem ter como fonte dados em outros formatos. Aos poucos, mais e mais informação vai sendo disponibilizada via HTTP em formato XML. Vejamos então um exemplo.

O site da bolsa de valores eletrônica NASDAQ disponibiliza uma URL que retorna a cotação de qualquer ação cujo código for passado como parâmetro componente da URL. Por exemplo, chamando-se a URL

http://quotes.nasdaq.com/quote.dll?mode=stock&page=xml&symbol=SUNW, obtém-se a cotação das ações da Sun Microsystems, Inc, conforme figura abaixo.

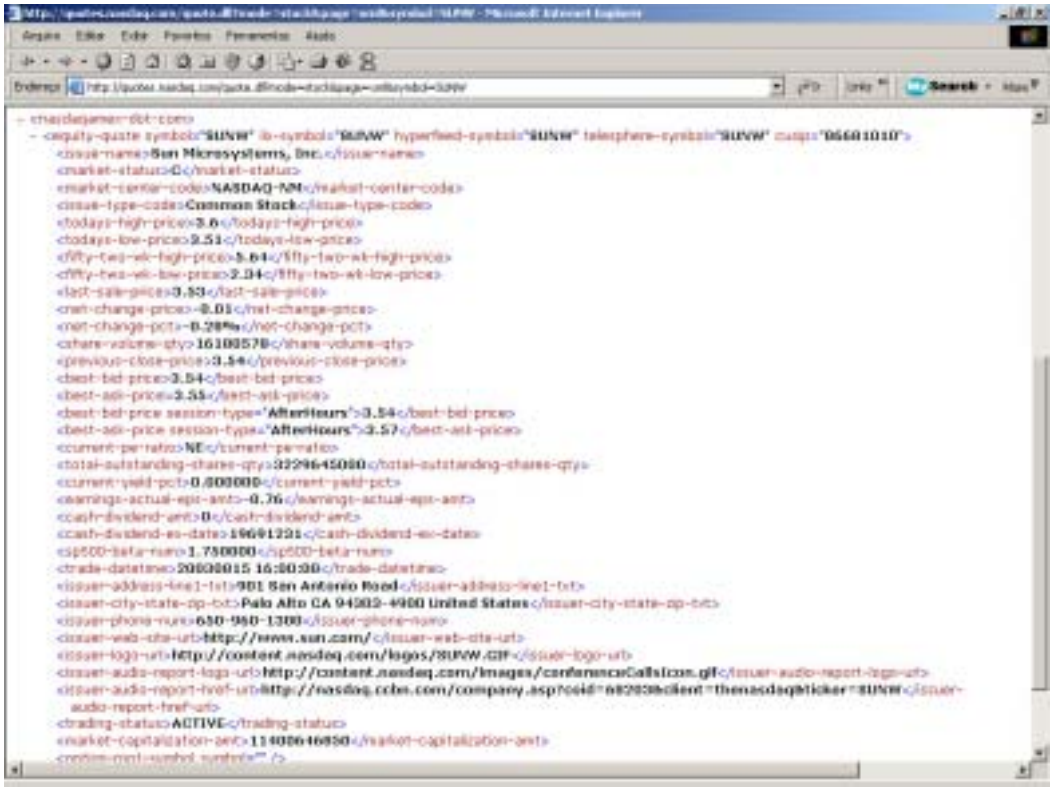


Figura 2.5 – Documento resposta XML exibido no browser Internet Explorer

Diferente do HTML, XML não destina-se à visualização direta. Desta forma, a imagem do documento construída pelo *browser* é simples, aproximando-se muito do código conteúdo do documento.

No caso de XML, vizinhanças que serão utilizadas pelo processo de extração corresponderão, na maioria das vezes, às próprias *tags* que acompanham os dados de interesse.

```
<equity-quote symbol="SUNW" ilx-symbol="... cusip="86681010">
<issue-name>Sun Microsystems, Inc.</issue-name>
<market-status>C</market-status>
<market-center-code>NASDAQ-NM</market-center-code>
<issue-type-code>Common Stock</issue-type-code>
<todays-high-price>3.6</todays-high-price>
```

```
<todays-low-price>3.51</todays-low-price>
<fifty-two-wk-high-price>5.64</fifty-two-wk-high-price>
<fifty-two-wk-low-price>2.34</fifty-two-wk-low-price>
<last-sale-price>3.53</last-sale-price>
<net-change-price>-0.01</net-change-price>
<net-change-pct>-0.28%</net-change-pct>
<share-volume-qty>16180578</share-volume-qty>
```

Figura 2.6 – Vizinhanças dos dados de interesse no documento XML

Em primeira análise, processos de extração podem parecer desnecessários para documentos XML. Isto se deve à falsa impressão de que um documento XML carrega consigo semântica associada aos dados. No entanto, o fato de, como seres humanos, conseguirmos prever que a *tag* `<todays-high-price>` deve acompanhar o dado associado ao conceito do maior preço da ação no dia, não diz muito para um processo automático que precise obter esse dado, assim como não diria muito a alguém que não soubesse inglês. A discussão acerca do papel limitado de XML no contexto de interoperabilidade semântica será aprofundado no capítulo 3.

2.3 Dificuldades da Extração de Dados Semi-Estruturados

Para poder ir além dos casos mais simples, o processo de extração deve trazer estratégias para lidar com uma série de fatores complicadores que podem surgir quando se lida com dados semi-estruturados.

Atributos ausentes. Documentos mais complexos podem trazer algumas tuplas com certos atributos nulos ou mesmo completamente ausentes, isto é, faltando até mesmo seus marcadores e delimitadores. Por exemplo, em uma consulta que agrega informações de endereço, tuplas podem trazer o atributo país somente quando o endereço for fora do Brasil.

Atributos multi-valorados. No caso mais comum e mais simples, um documento trará uma ou mais tuplas, cada uma composta por atributos, sendo que cada atributo traz um único valor. Diferente do modelo relacional, a representação de dados em documentos semi-estruturados não traz consigo

recomendações para modelagem normalizada dos dados, portanto estruturas com atributos multi-valorados podem ocorrer. Por exemplo, uma cadeia de lojas, ao figurar em um documento, pode listar todos os endereços onde está presente.

Diferentes ordenações dos atributos. Apesar de não ser tão comum em casos práticos, documentos semi-estruturados podem trazer tuplas com seus atributos em ordens diferentes.

Delimitadores disjuntivos. Nos casos mais simples, consideramos que cada tipo de atributo poderia ser reconhecido pela presença de um determinado delimitador. Em páginas HTML, no entanto, delimitadores são somente informação idiossincrática, situação oposta ao que acontece em documentos XML, nos quais o delimitador funciona como um identificador seguro daquele determinado tipo de atributo. Em documentos HTML, portanto, podemos ter mais de um possível delimitador para um mesmo tipo de atributo. Por exemplo, um *site* de comércio eletrônico pode trazer preços em negrito, com exceção de preços em promoção, que são mostrados em vermelho. Da mesma forma, pode haver mais de um tipo de atributo compartilhando o mesmo delimitador.

Erros tipográficos e exceções. Como sabemos, documentos HTML podem ser mal-formados, contendo erros de diferentes tipos. Diferente de XML, HTML não possui um formalismo de validação estrutural do documento, como DTD ou XML *Schema*. O problema é ainda agravado pelo fato que a maioria dos browsers HTML tende a ser permissiva, resolvendo e mostrando, sem maiores problemas, documentos mal formados, o que leva a uma proliferação dos mesmos pela Web. Se ocorre um erro tipográfico nos elementos de formatação que direcionam a extração, como os delimitadores, um processo de extração simples pode falhar. No entanto, no caso característico, documentos HTML ricos em dados, e portando de maior interesse no contexto de extrações, são gerados automaticamente por programas, normalmente a partir de templates. Se o template contiver um erro de formatação, este estará presente em todos os documentos gerados, e não provocará falha no processo de extração.

2.4 Extração de dados de fontes HTML

HTML é uma linguagem de marcação voltada para apresentação. Desta forma, informações que poderiam servir como guia para localização do conteúdo aparecem misturadas a diretivas de formatação dos dados para visualização. Além disso, HTML não traz, de forma explícita, nenhuma informação estrutural, descrevendo como a informação mais complexa é formada a partir de dados atômicos, ou que possa ter algum uso semântico, como nome de campos e tipos de dados. O conjunto de *tags* HTML é fixo, não sendo possível criar novas *tags* para uso específico. HTML também não traz nenhum mecanismo para validação estrutural, e o que se vê na prática é uma grande proliferação de documentos HTML mal formados na Web, uma vez que os mesmos são normalmente exibidos sem maiores problemas pelos principais *browsers*[10].

Devido a estas características, o uso direto de informação presentes em documentos HTML para um fim outro além da visualização e leitura humanas, é inviável. Um processo de extração de dados se faz necessário para viabilizar uma utilização mais completa das informações.

A seguir, será apresentado um breve levantamento de técnicas utilizadas para extração de dados a partir de fontes HTML.

2.4.1 *Wrappers específicos*

As primeiras propostas de *wrappers* para fontes HTML que surgiram [GMPQ+97, CCS00] traziam as regras de extração embutidas no código ("*hard-coded*") o que dificultava sua manutenção e reuso. Desta forma, para realizar extrações em um novo *site*, era necessário reescrever o programa.

Para evitar isso, surgiram *wrappers* que usavam arquivos de configuração, baseados em gramáticas especializadas, constituídos de regras escritas de forma declarativa [HGMC+98, GRVB98], que o preparavam para a tarefa de *wrapping* de uma determinada fonte. Desta forma, a mesma implementação de um *wrapper* podia ser utilizada para diferentes fontes HTML, através da edição manual dos

[10] Duas iniciativas do W3C tentam minimizar o problema da proliferação de documentos mal-formados na Web. A primeira delas é HTMLTidy [Rag03], um programa capaz de consertar problemas em arquivos HTML mal-formados. A segunda é a especificação XHTML [Pem+02], uma reformulação de HTML em sintaxe XML, de forma a permitir o uso dos mecanismos de validação de XML.

arquivos de configuração, que permitem especificar como os dados estão dispostos na página HTML em questão.

2.4.2 Construção semi-automatizada de *wrappers*

Uma das dificuldades na construção de *wrappers* para fontes Web são as mudanças freqüentes no formato das páginas HTML que hospedam os dados, o que dificulta a manutenção dos *wrappers*. Ter as regras de extração representadas de forma declarativa editável pelos usuário não soluciona o problema, mas é uma forma de amenizá-lo.

Uma evolução que surgiu foi a criação de interfaces WYSIWYG para geração das regras de extração. Em [SA01], uma ferramenta chamada *extraction-wizard* é disponibilizada para o usuário, na qual este visualiza o documento da mesma forma que faria em um browser HTML. Nesta interface, o usuário marca informações de interesse, e a ferramenta automaticamente gera regras de extração para cada informação. As regras geradas não apresentam refinamento suficiente para serem utilizadas diretamente, necessitando passar por revisão manual, servindo apenas como um facilitador na construção de regras.

Outra estratégia para facilitar a construção de *wrappers* baseia-se na disponibilização, pelo usuário, de exemplos de objetos complexos (tuplas) que devem estar presentes no resultado da extração sobre um determinado documento. Em [RNLS99], alguns poucos destes objetos exemplo e o documento alvo da extração são submetidos a um programa capaz de gerar as regras de extração para a fonte em questão.

2.4.3 Técnicas de representação do conhecimento

Outras técnicas baseadas em exemplos, porém em maior número, objetivam não a geração direta de regras de extração declarativas, e sim informação representando conhecimento sobre os documentos, as informações de interesse, e como estas estão presentes naqueles.

Os exemplos, neste caso, não são de objetos já montados, mas o que chamamos documentos anotados, que são documentos acompanhados da

anotação de quais fragmentos devem ser extraídos. O sistema de aprendizado, então, generaliza a partir desses exemplos, de modo a produzir uma forma de conhecimento ou regras capazes de guiar a extração de conteúdo similar a partir de outros documentos. Em [Kus97], Kushmerick faz a primeira formulação do problema da geração de um *wrapper* Web como um problema de aprendizado indutivo, além de identificar seis classes de *wrappers* simples, com seus respectivos algoritmos de aprendizado. Um passo inicial no processo de indução identifica e classifica a estrutura do tipo de documento alvo da extração, direcionando o processo para a geração de um *wrapper* da classe correspondente. Por exemplo, se o tipo de documento apresenta uma estrutura HLRT – composta de cabeçalho (*head*), um corpo contendo tuplas de estrutura plana, que podem ser reconhecidas e extraídas percorrendo-se o corpo da esquerda (*left*) para a direita (*right*), e finalizada por um rodapé (*trail*) – um *wrapper* da classe HLRT será gerado. O trabalho desenvolvido em [Kus97] deu origem à ferramenta de indução de *wrappers* WIEN [Kus00b].

Outra ferramenta que merece destaque é STALKER [MMK99], que gera wrappers capazes de lidar com dados hierárquicos. Para tal, o programa de indução recebe como entrada, além dos exemplos de treinamento, uma descrição da estrutura hierárquica das entidades presentes no tipo de documento, na forma de uma árvore, chamada ECT (*Embedded Catalog Tree*).

O problema com as técnicas de aprendizado é a necessidade de prover os exemplos de treinamento. Muitas vezes, para que a gramática fique precisa, é necessário um grande número de exemplos. Mesmo assim, sistemas de extração de informações baseados nessas técnicas partem da idéia de que é mais fácil gerar exemplos anotados do que escrever regras de extração, que normalmente requer habilidades similares àquelas utilizadas em programação. No entanto, se o número de exemplos exigidos não for pequeno, tal idéia pode se mostrar equivocada.

2.5 Extração de dados de fontes XML

Assim como HTML, XML é uma linguagem de marcação derivada de SGML [ISO86]. No entanto, XML não foi projetado para descrever documentos que deveriam ser mostrados diretamente para o usuário final. Um documento XML é apenas precariamente legível, uma vez que não traz em si diretivas de diagramação e formatação. Nos casos em que os dados serão utilizados para visualização por pessoas, o documento XML normalmente recebe tratamento adicional[11]. A separação entre dados e apresentação, imposta por XML, é uma de suas principais vantagens.

Ao contrário do que acontece com HTML, a marcação utilizada em um documento XML é indicadora tanto de sua estrutura quanto de um vocabulário particular associado aos dados. Desta forma, XML pode ser usado para representar, de uma forma mais explícida, dados no modelo relacional ou orientado a objetos. Ao contrário de HTML, em XML não existe um conjunto fechado de *tags*, nem um sentido global associado a uma *tag*. O conjunto de *tags* a ser usado, assim como o sentido ou interpretação de cada uma, é de responsabilidades das partes envolvidas no cenário que usa determinado fomato XML.

Na verdade, XML foi projetado para ser manipulado por programas, em uma infinidade de situações. Para este fim, a grande vantagem de XML é a sua extensibilidade, que permite que ele seja usado para modelar e representar virtualmente qualquer tipo de dado, para praticamente qualquer finalidade.

Esta grande vantagem, no entanto, representa um grande problema. Por ser tão flexível, muitas vezes não é possível prever qual formato específico terá o documento retornado por uma fonte XML, uma vez que o responsável pela fonte de dados pode livremente escolher e criar as *tags* que irá utilizar, e organizá-las arbitrariamente para formar a estrutura do documento de retorno, de modo que esta se adapte ao conteúdo que se quer transmitir. Conjuntos diferentes de *tags*, nomes diferentes para elementos e atributos, diferentes formatações para valores numéricos e datas, ou diferentes estruturas de documentos dificultam a tarefa de fazer uso eficaz das informações. Muitas vezes, para que as informações possam

[11] Seguindo as recomendações do W3C, transformações de XML para fins de apresentação

ser utilizadas de forma mais ampla, é necessário que esta seja trazida para um formato mais conveniente, seja através de seleção parcial, reorganização, mudanças no vocabulário de *tags*, e outras operações. Em outras palavras, um passo de extração de dados se faz necessário.

Existem esforços de padronização dos formatos dos documentos XML utilizados em várias áreas, o que minimizaria a necessidade de se usar soluções de interoperabilidade envolvendo extração de dados. No entanto, devido à grande quantidade de diferentes "padrões" que se desenvolveram em algumas áreas, como comércio eletrônico, é comum o caso em que as partes envolvidas usam padrões distintos, o que exige tradução de documentos de um formato para outro, processo que envolve técnicas de extração de dados.

2.5.1 XSLT

Como foi dito, XML não traz diretivas para apresentação visual do documento, sendo responsável apenas pelo conteúdo e sua estrutura. Desta forma, se os dados contidos em um documento XML precisarem ser publicados em algum formato visual, o documento XML deverá passar por uma formatação.

Seguindo as recomendações do W3C, essa formatação deve ser especificada através de folhas de estilo, ou *stylesheets*. A linguagem para especificação de *stylesheets* não é única. Na verdade, existem duas recomendações W3C para linguagens de *stylesheets*: CSS e XSL. A primeira e ser criada foi CSS (*Cascading Stylesheets*) [LB99], com o objetivo de tentar isolar dos documentos HTML parte das diretivas de formatação. Após a introdução de XML, o W3C intensificou os trabalhos na especificação XSL [ABC+01] (*Extensible Stylesheet Language*), inicialmente com o objetivo de padronizar a geração de apresentação a partir de conteúdo XML. Desde o início de seu desenvolvimento, foi estabelecido que o processo deveria ser composto por dois estágios distintos: transformação e formatação. A especificação de transformação foi finalizada primeiro, e não demorou muito para que se notasse que esta tinha utilizações que iam além do seu propósito original, o que culminou no estabelecimento da

---

devem ser realizadas através de *stylesheets* XSL.

especificação XSLT (*XSL Transformations*) como um sistema de transformação XML de uso geral, apesar de não ser oficialmente, segundo o W3C, de propósito geral. A especificação de formatação continua em desenvolvimento, e é hoje conhecida como XSL:FO (*XSL Formatting Objects*).

XSLT pode ser usado para transformar XML em virtualmente qualquer formato textual arbitrário de saída, incluindo *flatfiles* ou Tex, apesar deste não ser um uso considerado padrão e normalmente exigir esforço adicional. O mais comum é que XSLT seja usado para transformar um documento XML em outro documento XML de formato distinto. Estes casos incluem a transformação para formatos XML destinados à apresentação, como HTML (na verdade, XHTML [Pem+02]) e XSL:FO [ABC+01]. No entanto, nem todo formato de saída de uma transformação se destina à exibição. XSLT é também a solução padrão para transformação entre formatos XML em cenários onde o documento gerado se destina a sistemas computacionais, como troca eletrônica de documentos (EDI) e comércio eletrônico baseados em XML.

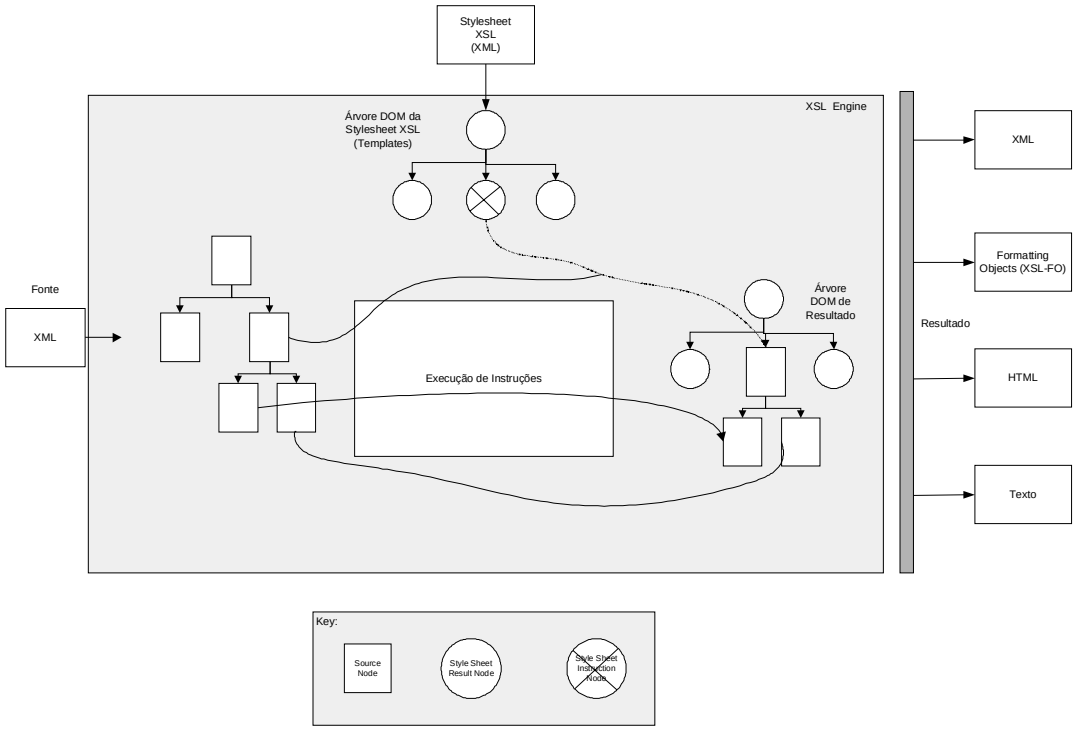


Figura 2.7 – Funcionamento de um processador XSLT

Uma transformação XSLT parte da árvore XML correspondente ao documento de entrada e, seguindo as instruções contidas na *stylesheet* XSLT, gera a árvore correspondente ao documento de saída.

Uma *stylesheet* XSLT é formado basicamente por regras *template*. Uma regra *template* associa padrões (*patterns*) que podem ocorrer na entrada, com a saída que deve ser produzida, na forma do *template* trazido na regra. O exemplo abaixo traz uma *stylesheet* que obtém dados a partir de documentos XML no formato retornado pelo site da NASDAQ.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
     <xsl:output method="xml"/>

     <xsl:template match="/nasdaqamex-dot-com">
         <xsl:element name="Cotacoes">
             <xsl:apply-templates select="equity-quote"/>
         </xsl:element>
     </xsl:template>

     <xsl:template match="equity-quote">
         <xsl:element name="Acao">
             <xsl:element name="Simbolo">
                 <xsl:value-of-select="@symbol"/>
             </xsl:element>
             <xsl:element name="Nome">
                 <xsl:value-of-select="issue-name"/>
             </xsl:element>
             <xsl:element name="PrecoMaxHoje">
                 <xsl:value-of-select="todays-high-price"/>
             </xsl:element>
             <xsl:element name="PrecoMinHoje">
                 <xsl:value-of-select="last-sale-price"/>
             </xsl:element>
             <xsl:element name="VolumeNegociado">
                 <xsl:value-of-select="share-volume-qty"/>
             </xsl:element>
         </xsl:element>
     </xsl:template>

</xsl:stylesheet>
```

Figura 2.8 – *Stylesheet* de transformação XSL

Um documento gerado pode conter total ou parcialmente os dados presentes no documento de origem, que podem ser compostos com informações adicionais, não presentes no documento original, incluídas pelo *template* no processo de transformação. É fácil perceber que o papel desempenhado por diretivas como `match`, `select`, `value-of-select` e outras é claramente de extração de dados do documento de origem, que são usados como matéria-prima na geração do documento de saída. Assim, a tarefa de transformação desempenhada por XSLT pode ser vista como uma tarefa de extração de dados, na qual a etapa de disponibilização dos resultados é flexível, sendo possível configurar o formato de saída. Quando o interesse está somente em parte da informação contida no documento de origem, torna-se clara a idéia intuitiva de extração de dados.

No entanto, XSLT tem suas limitações. A principal é o fato de que o documento de entrada tem que ser um documento XML bem formado, para que a árvore de origem possa ser montada. Desta forma, XSLT não pode ser usado de forma geral para realizar transformações a partir de HTML, pois, como discutido anteriormente, documentos HTML podem ser potencialmente mal formados.

2.5.2 XQuery

XML foi concebido inicialmente para publicação eletrônica em larga escala. Não demorou muito, no entanto, para que seu escopo fosse gradualmente avançando para utilizações envolvendo representação e troca de dados, e consequentemente atraindo a atenção fora da área de documentos, em particular em meio à comunidade de bancos de dados. Concluiu-se que, à medida que XML fosse ficando mais difundido, mais fontes de dados tenderiam a prover uma visão externa em formato XML, independente da tecnologia de armazenamento de dados usada internamente. Assim, ficou clara a necessidade por formas de consultar tais fontes, ou seja, de se ter uma linguagem de consulta XML. Começaram então a surgir propostas e implementações de linguagens de consulta XML, como XML-QL [DFF+98], XQL[RLS98] e XML-GL[CCD+98], dentre outras.

Em dezembro de 1998, o W3C promoveu o *workshop* QL'98, que contou com a participação dos principais fabricantes de bancos de dados, especialistas convidados e membros do consórcio. O QL'98 foi o marco inicial da união de esforços rumo a uma recomendação oficial do W3C para a necessidade de consultas a dados XML. Um dos principais produtos do *workshop* foi um resumo de requisitos para a tecnologia de consultas XML [CM98], dentre os quais foram apontados os seguintes:

- linguaguem não-procedural
- sintaxe XML
- alinhamento com outros padrões XML
- suporte a funções básicas de consulta
- suporte a joins
- suporte a operações de atualização dos dados
- suporte a operações em vários documentos
- facilidades de construção de XML como resposta

Em 1999, foi criado no W3C o grupo de trabalho *XML Query*, com a incumbência de conduzir o processo e prover as soluções na forma de recomendações do W3C. O trabalho do grupo consiste em produzir um modelo de dados para documentos XML, um conjunto de operadores sobre esse modelo de dados, e uma linguagem de consulta baseada nesses operadores, que recebeu o nome de XQuery [CCF+03].O modelo de dados para XQuery é, na verdade, um novo modelo de dados comum para XQuery, XPath[CD99] e XSLT, o que explicita o esforço do W3C no alinhamento destas tecnologias claramente interrelacionadas. Este modelo de dados define as informações manipuláveis por um processador de XSLT ou XQuery, assim como os possíveis valores que podem ser assumidos por expressões em todas as fases do processamento. O modelo de dados é baseado no XML Information Set [CT01], acrescido de suporte ao sistema de tipos de XML *Schema* [TBM+01]. Além desses, é possível representar coleções de documentos e de valores complexos.

A linguagem XQuery opera em instâncias do modelo de dados, gerando instâncias do modelo de dados. Adicionalmente, a linguagem define a transformação de uma instância do modelo de dados para um XML Infoset. XQuery é uma linguagem funcional, ou seja, formada por expressões que retornam valores e não têm efeitos colaterais. Expressões são normalmente compostas de expressões de mais baixo nível, combinadas por operadores ou palavras chaves. Literais, variáveis e chamadas de função são exemplos de expressões.

Outro tipo de expressão que merece destaque são as expressões de caminho, que usam a mesma sintaxe de XPath e XSLT para referenciar e selecionar nós através de seus caminhos na hierarquia dos documentos. Outra forma de selecionar nós é através de predicados, compostos por uma expressão entre colchetes que determina critérios para escolher nós, baseado em suas características.

Além de selecionar nós existentes, a linguagem é capaz de criar novos nós, assim como conteúdo para os mesmos, através de expressões de construção de elementos.

Finalmente, XQuery é munida de um poderoso esquema de controle de fluxo e repetição, através de expressões FLWR. Expressões "*flower*" (como são pronunciadas) podem ter várias cláusulas `for`, várias cláusulas `let`, opcionalmente uma cláusula `where`, e uma cláusula `return`. Para mostrar um exemplo de uma consulta XQuery característica, que faz uso de uma cláusula FLWR, será considerado o documento de dados XML abaixo.

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price> 65.95</price>
    </book>

    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
```

```
    </book>

    <book year="2000">
        <title>Data on the Web</title>
        <author><last>Abiteboul</last><first>Serge</first></author>
        <author><last>Buneman</last><first>Peter</first></author>
        <author><last>Suciu</last><first>Dan</first></author>
        <publisher>Morgan Kaufmann Publishers</publisher>
        <price>39.95</price>
    </book>

    <book year="1999">
        <title>The Economics of Technology and Content for Digital TV</title>
        <editor>
               <last>Gerbarg</last><first>Darcy</first>
                <affiliation>CITI</affiliation>
        </editor>
            <publisher>Kluwer Academic Publishers</publisher>
        <price>129.95</price>
    </book>

</bib>
```

Figura 2.9 – Documento de dados XML alvo da consulta XQuery [TBM+01]

A consulta abaixo retorna o nome de todos os autores presente no documento de entrada, cada um acompanhado dos títulos de todos os livros do mesmo, construindo o resultado dentro de um elemento “result”.

```
<results>
  {
    let $a := doc("http://www.bn.com/bib/bib.xml")//author
    for $last in distinct-values($a/last),
        $first in distinct-values($a[last=$last]/first)
    return
        <result>
            { $last, $first }
            {
                for $b in doc("http://www.bn.com/bib.xml")/bib/book
                where some $ba in $b/author
                      satisfies ($ba/last = $last and $ba/first=$first)
                return $b/title
            }
        </result>
  }
</results>
```

Figura 2.10 – Exemplo de consulta XQuery [TBM+01]

A cláusula `where`, através de sua condição expressa como uma expressão XPath ou um predicado, realiza a seleção de informações a partir de um ou vários documentos origem, ou seja, realiza a etapa de **extração** de dados. A cláusula `for` realiza o controle do fluxo de repetição sobre as seqüências extraídas, dentro do qual se realiza a **construção** da informação de saída, através da cláusula `return`.

Além das funcionalidades de extração e construção, existentes também em XSLT, XQuery possui facilidades de **combinação** de informações extraídas, que podem, como foi dito, serem originárias de um mesmo documento, ou de vários documentos distintos. Facilidades de combinação podem ser obtidas pelo uso de operações sobre seqüências, como `union`, `intersect` e `except`.

Assim como XSLT, XQuery não é capaz de atuar sobre documentos não-XML, exigindo documentos de entrada bem formados. No entanto, dentro do contexto deste trabalho, a principal limitação das técnicas de manipulação de documentos XML que envolvem extração de dados, é a dificuldade que estaria associada a um esquema qualquer de geração ou atualização automática de uma *stylesheet* ou consulta. Como será discutido mais a frente, a geração automática da configuração da extração é um requisito importante dentro do contexto deste trabalho.

2.6 Generalização dos Conceitos de Extração de Dados

Soluções para extração de dados de HTML ou XML, padronizadas pelo W3C ou frutos de desenvolvimentos próprios, seguem os mesmos conceitos básicos.

Esta sessão traz uma tentativa de generalização desses conceitos, de forma a proporcionar a discussão das questões em um grau de abstração mais elevado, independente das tecnologias específidas. Mais à frente neste trabalhos, esses conceitos serão utilizados como base para as soluções genéricas propostas.

2.6.1 Descritor de Extração

Como foi visto, um processo que envolve extração de dados segue um conjunto de regras para obter os dados contidos em um documento. O conjunto

destas regras, que podem estar embutidas no próprio programa, ou escritas em separado em um formato declarativo qualquer, formam o que chamamos de **descritor de extração**.

A base de um descritor de extração são informações que possibilitam localizar dados atômicos dentro do documento, aos quais chamamos genericamente de **padrões de extração de atributo (PEA)**. Um padrão de extração de atributo pode ser baseado em referências posicionais com relação à estrutura do documento, como as expressões em formato proprietário utilizadas em [HGMC+98], ou as expressões XPath utilizadas em *stylesheets* XSLT e consultas XQuery. Outra possibilidade são padrões de extração de atributo baseados em contextos locais, ou seja, no texto circunvizinho às ocorrências do atributo.

Padrões de extração de atributo baseados em referências posicionais descrevem o caminho navegacional na estrutura do documento que leva à localização do dado de interesse, através do que chamamos de **expressão de caminho**. Chamamos de **descritor de extração posicional** aqueles formados por padrões de extração desse tipo. *Stylesheets* XSLT são exemplos de descritores de extração posicionais, escritos utilizando XPath para as expressões de caminho no documento XML alvo da extração. O descritor de extração apresentado em [HGMC+98] é um exemplo de descritor de extração posicional para fontes HTML, que apresenta uma sintaxe própria para expressões de caminho. O grande problema com os modelos baseados em caminhos estruturais é a existência de documentos mal formados, o que é comum em alguns cenários, como é o caso dos documentos HTML. Pelo fato dos browsers serem capazes de exibir documentos HTML mal formados, os mesmos se proliferaram em quantidade considerável na Web.

Um **descritor de extração comparativo** é formado por padrões de extração de atributo baseados nas possíveis vizinhanças das ocorrências de cada atributo que se deseja extrair. Estes padrões de extração serão utilizados por rotinas de busca baseada em padrões, que tentam encontrar, dentro de uma dada string, fragmentos que se enquadram em um dado padrão. Chamamos esse tipo de padrão de extração de **expressão de padrão**. Dentre as possibilidades para

implementação de rotinas que trabalhem com reconhecimento de padrões em strings a partir de expressões de padrão, provavelmente a mais usual seja o uso de expressões regulares.

2.6.2 Extratégia de Extração

Na execução de sua tarefa, o processo de extração deverá adotar uma **estratégia de extração** que, seguindo o modelo de extração, consiga efetivamente obter e reestruturar os dados contidos no documento resposta. De maneira geral, podemos agrupar as estratégias de extração em dois grupos: as estratégias *top-down* e as estratégias *bottom-up*.

Em uma **estratégia de extração *top-down***, o processo de extração tenta primeiramente reconhecer objetos complexos, ou seja, tuplas inteiras, isolando o fragmento de texto onde uma tupla se encontra, para então realizar a extração de todos os atributos daquela tupla. Para isso será necessária a utilização de um **padrão de extração de tupla (PET)**. No caso de um descritor de extração comparativo, o PET será uma nova expressão de padrão construída a partir dos PEAs, enquanto em um descritor de extração posicional o PET corresponderá a uma expressão de caminho da qual dependem as expressões de caminho dos PEAs.

Já um processo de extração que segue uma **estratégia de extração *bottom-up*** realiza inicialmente a extração de todos os atributos presentes em um documento, para então agrupá-los nas respectivas tuplas. Não é necessário, portanto, um PET, sendo o descritor de extração composto somente pelos PEAs. Estratégias *bottom-up* são consideravelmente mais complexas do que estratégias *top-down*, principalmente devido à tarefa de agrupamento posterior dos atributos em tuplas. No entanto, somente uma estratégia *bottom-up* possibilita tratar satisfatoriamente problemas característicos da extração de dados semi-estruturados, como atributos ausentes, multivalorados, ou em diferentes ordens. Para casos mais simples, em que as tuplas apresentam maior constância de formato, deve-se optar por uma estratégia *top-down*, devido à menor complexidade dos algoritmos.

Trabalhos existentes voltados para extração a partir de fontes HTML utilizam, na sua maioria, estratégias de extração *top-down*. Como exceção mais sofisticada, que utiliza estratégia *bottom-up*, podemos citar o DeByE [RNLS99].

As tecnologias de extração e transformação de dados XML realizam normalmente um tipo de extração que se enquadra na categoria das extrações *top-down*. Devido ao poder expressivo superior, que possibilita criar variáveis que podem acumular coleções de nós para uso porterior, XQuery possibilita combinar os dados extraídos de formas mais complexas, o que permite que algo similar a uma extração *bottom-up* seja escrito, o que não é possível com XSLT.

O principal desafio para a criação de uma estratégia de extração *bottom-up*, é a resolução do seguinte problema:

- **Montar objetos complexos a partir de objetos atômicos extraídos individualmente.** O processo deve seguir uma lógica para determinar quais atributos pertencem à mesma tupla, provavelmente levando em consideração as posições dos atributos no documento original. Se houver a possibilidade do processo de extração retornar atributos falsos, será necessário que estes sejam desprezados na montagem das tuplas.

2.7 Conclusões do Capítulo

Neste capítulo, estabelecemos nossos conceitos de fontes textuais dinâmicas e da questão da extração de dados a partir dessas fontes. Nosso interesse são as fontes textuais que entregam respostas em linguagens de marcação, caracteristicamente em formato HTML e XML, devido ao grande número delas e conseqüente importância. Discutimos a evolução dos trabalhos sobre extração de dados de fontes HTML, além de técnicas relacionadas ao conceito de extração de dados para XML, como XSLT e a linguagem de consulta XQuery.

Neste trabalho, procuramos tratar extração de dados de fontes HTML e XML como um problema único, de modo a podermos nos concentrar nos problemas que, em sua maioria, são comuns aos dois cenários. Para tal, apresentamos uma tentativa de formalização do problema e de conceitos envolvidos no contexto do

mesmo, de forma a gerar um vocabulário comum para discussão em mais alto nível, na seqüencia deste trabalho.

Capítulo 3

Extração de Dados e Interoperabilidade Semântica

Interoperabilidade semântica é a capacidade de utilizar dados em cenários para os quais estes dados não foram originalmente projetados. Para tal, o processo interessado em utilizar esses dados terá que compreender o significado dos mesmos. Saber o que um determinado dado significa é, na verdade, ser capaz de mapeá-lo para o vocabulário conhecido.

Para exemplificar o conceito, tomemos o cenário de uma comunidade de compra e venda que troca dados eletronicamente. Se tivermos interoperabilidade semântica, um novo participante pode ser admitido na comunidade, sem a necessidade que seus documentos eletrônicos obedeçam a um determinado padrão pré-estabelecido, ou que sejam escritas traduções dos seus formatos de documentos para um formato comum.

A estratégia para interoperabilidade semântica proposta atualmente para a Web, através do movimento liderado pelo W3C e denominado *Semantic Web*, é construído sobre uma base que depende da disponibilidade de dados em um formato conveniente, no caso RDF[LS99]. Só estarão disponíveis na Web Semântica dados no formato RDF, ou que possam ser convertidos para este formato.

A solução proposta para fontes já existentes, em particular para aquelas que disponibilizam documentos HTML, é a inclusão de anotações RDF ao conteúdo. Além de requerer alteração nos documentos, em muitos cenários sem agregar vantagem nenhuma ao provedor da informação, esta solução gera uma indesejada redundância de dados, que em alguns casos pode criar problemas de manutenção futuros. Devido à impossibilidade de se referenciar fragmentos de um documento HTML via URI, informações RDF incluídas na forma de anotações são sempre

referentes ao documento como um todo, e não à unidades de informações individuais contidas no mesmo.

Para fontes em formato XML, a sugestão é a alteração do formato exportado para uma sintaxe RDF, o que exige esforço, em muitos casos sem que haja fator motivador à altura, e que pode se mostrar inviável em alguns cenários, devido a dependências já criadas com relação ao formato atual.

Devido às dificuldades de alteração e falta de motivação que justifique o esforço de mudança, um grande número de fontes de informação na Web não disponibilizará informações no formato RDF, e esta barreira sintática que terá que ser vencida de outras maneiras, de forma a viabilizar a inclusão das informações dessas fontes no conjunto da Web Semântica.

Este trabalho propõe uma solução genérica e resiliente, aplicável a fontes textuais em qualquer linguagem de marcação, seja HTML, XML ou SGML, para extração de dados contidos nestas fontes em formato RDF. Resiliência, ou seja, a capacidade de um processo suportar mudanças, é característica altamente desejável em processos de extração de dados, e é discutida mais a fundo no próximo capítulo. A capacidade da extração de conviver com mudanças nas fontes origem permite um grau de independência sintática essencial para o contexto maior da interoperabilidade semântica.

Neste capítulo, apresentamos um resumo da infra-estrutura de interoperabilidade na Web proposta pelo W3C, para em seguida discutimos nossa visão do papel de técnicas de extração resiliente neste contexto da chamada *Semantic Web*.

3.1 Interoperabilidade Semântica na Web

3.1.1 XML e Semântica

O uso de XML é, sem sombra de dúvida, um passo importante na evolução da Web como meio para troca de dados. Primeiro de tudo, XML traz **poder expressivo** suficiente para expressar qualquer forma de dados, seja qual for o uso pretendido para os mesmos. Ganhamos também **interoperapilidade sintática**.

XML pode ser manipulado por virtualmente qualquer aplicação, dada a variedade de componentes de software, como parsers e APIs, disponíveis para construção de aplicações. Esses componentes resolvem as questões relativas à manipulação de dados em formato XML, de modo que esse esforço não precisa ser refeito em cada aplicação.

No entanto, é preciso mais. Necessitamos que o formato utilizado permita-nos entender os dados que estão sendo trocados. Precisamos de **interoperabilidade semântica**.

É comum algumas pessoas se referirem a XML como "linguagem de marcação semântica", e expressarem sua empolgação com a suposta capacidade de XML expressar significado com clareza através da marcação. É fácil entender o que dá origem a essa primeira impressão. Comparado com outras formas de codificação de informação, como arquivos *flat-file* posicionais ou com separadores, planilhas eletrônicas vistas no seu formato interno, ou outra forma de codificação proprietária, XML realmente parece carregar consigo mais sentido. As próprias *tags* de marcação explicam que nome recebe cada informação individual, e como estas se reúnem em estruturas mais complexas.

Marcação descritiva não foi uma novidade introduzida por XML. SGML, precursora de XML e HTML criada em 1986, já trazia essas características. Na verdade, os princípios básicos já estavam presentes em trabalhos que influenciaram a criação de SGML, como DCF GML[Gol78], de Charles Goldfarb e Scribe[RW80], de Brian Reid, do final da década de 70 e início da década de 80.

No entanto, mesmo quando nomes de *tags* e atributos são cuidadosamente selecionados por especialistas no domínio da aplicação, o uso de marcação descritiva apresenta relevância limitada como mecanismo para possibilitar interoperabilidade entre sistemas independentes. Marcação XML possibilita a um ser humano prever que informação estará presente envolta por um par de *tags* que formam um elemento, como no caso de <telefone></telefone>. Para um processador XML, <telefone>, <titulo> e <i> são todas igualmente sem sentido.

XML possibilita que o autor escolha o vocabulário que irá usar para as *tags* do seu documento, mas não provê forma de associar esse vocabulário com conceitos de mais alto nível, nem de representar associações entre esses conceitos. Desta forma, apesar da possível falsa impressão que pode emergir de uma análise superficial, XML não é capaz, por si só, de tratar da semântica das informações.

3.1.2 DTD e XML *Schema*

Como os autores de documentos XML podem criar livremente as *tags* que irão usar, o que acontece na prática é a criação de dialetos XML para usos específicos. Um grupo determina um conjunto de *tags* a ser usado, cada uma com seu significado implícito, e acorda-se também a forma de estruturar os documentos. Aqueles que quiserem participar, devem concordar com as determinações, e assim todos do grupo conseguem se entender. Desta forma, cria-se uma nova linguagem de marcação utilizando-se XML como meta-linguagem. Surge então a necessidade de especificar este formato acordado, assim como de poder verificar se um determinado documento respeita este formato.

Como a especificação XML não impõe restrições quanto ao conjunto de *tags* a serem usadas, nem como estas devem ser organizadas hierarquicamente, o máximo que se pode dizer de um documento XML em si é se o mesmo é bem-formado ou não. Para determinar se um documento é estruturalmente válido, dentro de um determinado contexto de uso, é necessário associá-lo a um outro documento de definição de estrutura.

Regras de formação para um tipo de documento podem ser determinadas através de um DTD (*Document Type Definition*) ou de um XML *Schema*. Estes definem restrições acerca de quais tags usar, e como elas devem ser aninhadas dentro de um documento. XML *Schema*, que é uma especificação mais recente do que os DTDs, oferece uma série de vantagens. A principal delas é o conceito de tipos de dados. Um XML *Schema* pode definir que um determinado elemento ou atributo só pode receber valores de um determinado tipo de dado. Além dos tipos de dados definidos pela linguagem, XML *Schema* permite a declaração de

novos tipos, que podem ser derivados de tipos existentes. Outra importante vantagem sobre DTD é que XML *Schema* é compatível com XML Namespaces, o que possibilita a referência a tipos criados em outros documentos, através de URLs.

No entanto, o papel das duas especificações é o mesmo: definir uma gramática para documentos XML, especificando combinações e aninhamento de nomes de *tags*, atributos, e assim por diante.

3.1.3 RDF e RDF *Schema*

A flexibilidade de XML para criar dialetos, que podem ser formalizados por um DTD ou XML *Schema*, leva naturalmente à explosão no número dessas gramáticas, ou "padrões" para troca de dados em formato XML em diferentes áreas de negócio. Para comércio eletrônico, por exemplo, temos os "padrões" cXML e ebXML, além dos inúmeros outros propostos em fóruns como Biztalk.org e RosettaNet.org. Na verdade, setores que já evoluíram para um pequeno número de diferentes "padrões" são os melhores casos. Em algumas áreas de negócio, não existem padrões de fato estabelecidos, de modo que formatos de troca de informações são definidos sob demanda, muitas vezes sem maiores análises quando de suas criações.

A existência de várias possibilidades para codificação de um determinado domínio traz dificuldades em situações em que se necessita entender documentos de terceiros. Considere um conjunto de parceiros que trocam documentos XML como uma determinada finalidade de negócio. Se dois parceiros utilizam formatos diferentes para seus documentos, um passo de transformação XSLT tem que ser configurado e incorporado ao processo de troca. Se um parceiro que utiliza um novo formato é introduzido, passos de transformação para todos os outros formatos existentes precisam ser configurados, o que demanda um esforço considerável. Para evitar esse tipo de situação, é necessário uma camada de abstração em cima de XML, na qual se possa representar conceitos.

Tal camada para representação de conhecimento na Web deve levar em consideração alguns requisitos essenciais. Inicialmente, seguindo o modelo da

própria Web, o conhecimento deve ser distribuído e conectável. Segundo, não se pode trabalhar com modelos de mundo fechado, isto é, não podem haver verdades absolutas. Cada aplicação deve ter liberdade para construir suas verdades, e neste processo se basear, na forma que decidir, em conceitos de outras partes, que podem em alguns casos ser inclusive antagônicos. Finalmente, deve-se prezar pela simplicidade e extensibilidade, para que possam ser derivadas soluções para diferentes comunidades de usuários.

RDF [LS99], sigla para *Resorce Description Framework*, é a recomendação W3C para a definição e uso de meta-dados. Diferentemente de XML *Schema* ou DTD, voltados para tipagem e definição estrutural, a proposta de RDF envolve meta-dados conceituais, que têm por objetivo tornar informações disponíveis para aplicações diferentes daquelas para as quais estas informações foram originalmente criadas.

O modelo de dados de RDF consiste de 3 tipos de objetos.

- Um **recurso** é qualquer coisa que possa ser referenciada por uma URI. Isto inclui páginas Web, assim como elementos individuais em um documento XML.
- Uma **propriedade** é um aspecto específico, característica, atributo ou relação usada para descrever o recurso. Uma propriedade pode e deve ser um recurso, referenciada por uma URI e podendo ter suas próprias propriedades.
- Uma **sentença** consiste da combinação de um recurso, uma propriedade e um valor. Essas partes são chamadas, respectivamente, de sujeito, predicado e objeto da sentença. O objeto pode ser um literal, ou um outro recurso.

Abaixo é mostrado um exemplo de dados em formato RDF.

```
<?xml version="1.0"?>
<rdf:Description rdf:about="http://www.china.org/geography/rivers#Yangtze"
                 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns="http://www.geodesy.org/river#">
     <rdf:type rdf:resource="http://www.geodesy.org/river#River"/>
     <length>6300 kilometers</length>
     <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
     <endingLocation>East China Sea</endingLocation>
```

```
</rdf:Description>
```

Figura 3.1 – Exemplo de dados em formato RDF [CJ03]

O exemplo acima encontra-se na forma longa de descrição de um recurso em RDF. A forma longa tem a vantagem de deixar bastante claro que se está fazendo uma descrição sobre um determinado recurso (“http://www.china.org/geography/rivers#Yangtze”) de um determinado tipo, ou classe ("http://www.geodesy.org/river#River"). As outras informações são valores das propriedades “length”, “startingLocation” e “endingLocation” para este recurso. Cada propriedade expressa uma sentença que tem o recurso como sujeito, a propriedade como predicado, e o valor da propriedade como objeto.

Além da forma longa, dados RDF podem ser expressos em forma abreviada, como mostrado abaixo.

```
<?xml version="1.0"?>
<River rdf:ID="Yangtze"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns="http://www.geodesy.org/river#"
            xml:base="http://www.china.org/geography/rivers">
     <length>6300 kilometers</length>
     <startingLocation>western China's Qinghai-Tibet Plateau</startingLocation>
     <endingLocation>East China Sea</endingLocation>
</River>
```

Figura 3.2 – Dados RDF em forma abreviada [CJ03]

A forma abreviada tem a vantagem de ser mais próxima dos formatos XML mais comuns para representação de dados, o que facilita a migração de aplicações existentes que utilizam XML.

RDF foi concebido para prover um modelo de dados básico para meta-dados. Está fora do escopo de RDF definir a forma como a modelagem conceitual deve ocorrer. Em particular, RDF não define um conjunto de termos reservados para a modelagem de dados. A criação de vocabulários específicos fica a cargo de uma outra linguagem com esta finalidade, de forma análoga ao que acontece com XML.

RDF *Schema* [BG03] permite ao desenvolvedor definir um vocabulário particular para dados RDF, e especificar os tipos de objeto para os quais estes atributos podem ser aplicados. RDF *Schema* é, na verdade, uma extensão de RDF, que define classes e propriedades que podem ser usadas para descrever classes, propriedades e outros recursos, assim como grupos de recursos e a relação entre eles. Uma definição de vocabulário RDF, ou seja, um RDF *Schema* particular, é escrito em sintaxe RDF, acrescida dos recursos definidos na especificação RDF *Schema*.

Um RDF *Schema* define um conjunto de conceitos, chamados classes, e a relação hierárquica entre eles, definida através da propriedade `subClassOf`. Desta forma, um RDF *Schema* expressa uma taxonomia, ou seja, uma hierarquia de conceitos. As figuras a seguir mostram uma taxonomia em uma representação gráfica, e expressa em RDF *Schema*.

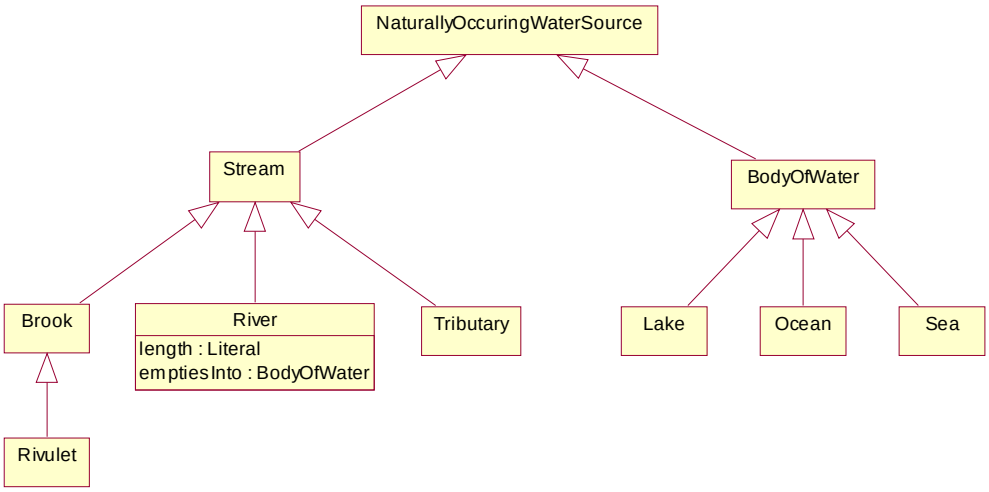


Figura 3.3 – Taxonomia representada em diagrama UML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                 xml:base="http://www.geodesy.org/water/naturally-occurring#">

    <rdfs:Class rdf:ID="NaturallyOccurringWaterSource">
    </rdfs:Class>
```

```
    <rdfs:Class rdf:ID="Stream">
        <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
    </rdfs:Class>

    <rdfs:Class rdf:ID="Brook">
        <rdfs:subClassOf rdf:resource="#Stream"/>
    </rdfs:Class>

    <rdfs:Class rdf:ID="Rivulet">
        <rdfs:subClassOf rdf:resource="#Brook"/>
    </rdfs:Class>

    <rdfs:Class rdf:ID="River">
        <rdfs:subClassOf rdf:resource="#Stream"/>
    </rdfs:Class>

    . . .

    <rdfs:Class rdf:ID="Tributary">
        <rdfs:subClassOf rdf:resource="#Stream"/>
    </rdfs:Class>

    <rdfs:Class rdf:ID="BodyOfWater">
        <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
    </rdfs:Class>

    <rdfs:Class rdf:ID="Lake">
        <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
    </rdfs:Class>

    <rdfs:Class rdf:ID="Ocean">
        <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
    </rdfs:Class>

    <rdfs:Class rdf:ID="Sea">
        <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
    </rdfs:Class>

</rdf:RDF>
```

Figura 3.4 – Taxonomia representada em RDF *Schema*

Além de classes, um RDF *Schema* traz facilidades para definição de propriedades. A sintaxe que RDF *Schema* traz propriedades a serem atribuídas a propriedades quando da sua definição. A propriedade de propriedade *domain* define uma classe à qual a propriedade pode ser aplicada, enquanto *range* define

o tipo dos valores que a propriedade pode conter. A figura abaixo traz exemplos de definições de propriedades em um RDF *Schema*.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                xml:base="http://www.geodesy.org/water/naturally-occurring#">

    . . .

    <rdfs:Class rdf:ID="River">
        <rdfs:subClassOf rdf:resource="#Stream"/>
    </rdfs:Class>

    <rdf:Property rdf:ID="emptiesInto">
        <rdfs:domain rdf:resource="#River"/>
        <rdfs:range rdf:resource="#BodyOfWater"/>
    </rdf:Property>

    <rdf:Property rdf:ID="length">
        <rdfs:domain rdf:resource="#River"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
    </rdf:Property>

    . . .

</rdf:RDF>
```

Figura 3.5 – Definição de Propriedades em RDF *Schema*

Diferente do que acontece em linguagens de programação orientadas a objetos, propriedades em um RDF *Schema* não são definidas como partes integrantes do objeto ao qual se aplica. A vantagem da separação entre classes e propriedades está na possibilidade de qualquer um, em qualquer lugar, criar uma propriedade que pode ser usada com uma dada classe, bem no estilo das tecnologias Web. Esta independência entre propriedades e a classes às quais estas se aplicam é expressa na própria forma de organização de uma arquivo RDF *Schema*. Na figura acima, podemos notar que as propriedades não são incluídas no bloco XML de definição da classe “River”. Na verdade, as propriedades associadas a esta classe não precisariam seguir imediatamente após a definição da

classe no arquivo, sendo isto uma mera questão de preferência do autor na organização do arquivo de definição de um RDF *Schema*.

Na verdade, são as características de tecnologia Web que tornam RDF e RDF *Schema* potencialmente úteis na tarefa de inferência e descobrimento de informações. RDF *Schema* provê a forma de definir e identificar unicamente conceitos na Web, através de URIs. Uma vez globalizados, estes podem ser usados na construção de novos conceitos. A partir dos relacionamentos explícitos, softwares de inferência podem derivar relacionamentos implícitos.

Apesar de promissora, a tecnologia RDF ainda deixa espaço para muitas dúvidas com relação à sua adoção em larga escala. Uma dificuldade imediata é o esforço necessário para adequar fontes Web existentes ao modelo de dados RDF, e a criação manual de representações conceituais para interoperabilidade na forma de RDF *Schemas*. Não está clara a estratégia que deve ser adotada para migração de formatos XML existentes, e também a relação entre RDF Schema e XML *Schema*.

A primeira dificuldade é o fato de que somente dados expressos dentro do padrão RDF estão disponíveis para interoperabilidade, pois somente estes podem ser associados a conceitos definidos em um RDF *Schema*. Isso implica que dados em formatos XML não compatíveis precisariam ser convertidos para o padrão RDF para fazer parte deste novo quadro. Isto poderia ser feito através de remodelagem das fontes XML para formatos RDF, de forma que cada documento já traria dados tratáveis por cenários de interoperabilidade. Outra estratégia seria a geração de dados no formato RDF a partir de documentos XML não compatíveis, através da criação de uma *stylesheet* XSLT, capaz de realizar transformação para RDF a partir de documentos XML de um determinado formato, ou seja, que seguem um determinado DTD ou XML *Schema*.

Este último cenário apresentado leva naturalmente ao questionamento sobre o relacionamento e a interação entre XML *Schema* e RDF *Schema*. Parece natural que os dois pudessem ser combinados de forma a não se necessitar de um passo de transformação XSLT no cenário apresentado. O caminho poderia ser através do relacionamento entre tipos definidos em um XML *Schema* e conceitos

definidos em um RDF *Schema*, como é proposto em [Hun01] para fontes multimídia. Isso permitiria a associação de conceitos a dados publicados em formato XML, sem necessidade de alteração do formato. Permitiria também a associação de conceitos a dados presentes em um dado tipo de documento, representado por um XML *Schema*, e não somente a associação de conceitos a dados presentes em um documento em particular, acessível através de uma determinada URI. Esta poderia ser a forma de enfrentar uma forte limitação do esquema de descrição semântica em costrução pelo W3C, que só permite a descrição semântica daquilo que é diretamente acessível via URI. Esta restrição é bastante séria, principalmente no contexto de conteúdo dinâmico, onde várias URIs distintas, porém similares, apresentando a mesma base mas cada uma contendo parâmetros distintos, acessam o mesmo tipo de documento. Apesar da necessidade apresentada, não há direcionamento claro do W3C sobre formas de integração entre XML e RDF *Schemas*.

3.1.4 Ontologias para Web

Em seu significado primário, originário da filosofia, o termo "ontologia" significa "um assunto da existência", ou seja, um registro sistemático de existência. Em Ciência da Computação, o termo foi adotado e vem sendo usado há mais de uma década, significando uma conceitualização consensual, formal e explícita do mundo como um todo, ou de um domínio em específico [G93].

Recentemente, ontologias têm sido usadas nas áreas de bancos de dados e recuperação de informações como um suporte para interoperabilidade de fontes de dados heterogêneas. Neste contexto, ontologias podem ser usadas para descrever a semântica das fontes de informação e tornar os conteúdos explícitos. Estas descrições das fontes podem então ser utilizadas para identificar e associar conceitos que possuam correspondência semântica.

Seguindo a linha da interoperabilidade de fontes de dados heterogêneas, ontologias são a peça final da infra-estrutura para interoperabilidade semântica na Web [semweb], que inclui RDF e RDF *Schema*. Como foi discutido, XML pode ser utilizado para troca de dados entre partes que concordam previamente com o

formato dos documentos, normalmente formalizado através de um DTD ou XML *Schema*. No entanto, as partes envolvidas não seriam capazes de compreender documentos XML em outro formato, que utilizasse um conjunto diferente de *tags*, mesmo que estes documentos trouxessem os dados esperados. O simples uso de outros termos, provavelmente para representar os mesmos conceitos presentes no formato XML conhecido, é suficiente para impossibilitar a interação entre as partes. Este é o problema atacado por RDF e RDF *Schema*, que permitem a associação de conceitos com identificadores. Através de RDF *Schema*, é possível definir hierarquia de conceitos, composta de classes e suas subclasses e superclasses, assim como propriedades, com suas sub-propriedades, *domain* e *range*. Na verdade, RDF *Schema* é uma linguagem de definição de ontologias, sendo, no entanto, limitada. As chamadas linguagens de ontologia Web são, em sua maioria, extensões de RDF *Schema* com primitivas de modelagem mais ricas, possibilitando desenvolver ontologias capazes de prover interoperabilidade entre um grande número de esquemas desenvolvidos e mantidos de forma autônoma.

No entanto, a aplicação de ontologias para fontes Web iniciou anteriormente à existência de RDF. As primeiras iniciativas, como SHOE [HH00] e Ontobroker [DEFS99], utilizavam um esquema de anotação ontológica nos próprios documentos HTML.

OIL (*Ontology Interchange Level*) [FH99] foi a primeira linguagem de definição de ontologias proposta como extensão de RDF *Schema*, em projeto da União Européia liderado pela Universidade de Amsterdã. Quase ao mesmo tempo, o departamento de defesa americano iniciou, através da DARPA, um projeto para criação de uma linguagem de definição de ontologias a serem exploradas por agentes inteligentes, denominada DAML (*DARPA Agent Markup Language*) [DAML00]. No final de 2000, as duas iniciativas uniram esforços, publicando uma linguagem conjunta denominada DAML+OIL [HHPS01], publicada inicialmente em dezembro de 2000, e revisada para a forma atual em março de 2001.

Em novembro de 2001, o W3C criou o grupo de trabalho *Web Ontology*, como parte da atividade *Semantic Web*. O grupo ficou incubido do desenvolvimento de uma futura recomendação W3C para linguagem de definição de ontologias, chamada OWL (*Web Ontology Language*) [PHH03], atualmente em desenvolvimento (*working draft*). DAML+OIL foi adotada como ponto de partida para a definição da nova linguagem.

Da mesma forma que suas precursoras, OWL é uma extensão de RDF *Schema*, que define primitivas de modelagem adicionais para descrição de classes e propriedades. Usando OWL, é possível expressar novas relações entre classes (disjunção, por exemplo), igualdade, cardinalidade, características de propriedades (como transitividade e simetria), entre outras.

Na verdade, OWL define três sublinguagens de crescente poder expressivo, concebidas tendo em vista diferentes cenários de uso e seus respectivos requisitos. A menos expressiva, *OWL Lite*, apesar do conjunto mais restrito de primitivas de modelagem, suporta a criação de hierarquias de classificação e restrições simples, sendo suficiente para expressar uma parcela considerável das taxonomias. *OWL DL*[12] provê a máxima expressividade possível que ainda preserva completude e decidibilidade computacionais. *OWL Full*, por outro lado, não impõem limites à expressividade e à liberdade sintática de RDF, não dando, no entanto, nenhuma garantia computacional.

Nas figuras abaixo, estendemos o exemplo de taxonomia introduzido e modelado em RDF *Schema*, através do uso de primitivas OWL.

[12] O nome OWL DL é devido à correspondência com lógica de descrições (*description logics*), campo de pesquisa que tem estudado um determinado fragmento decidível da lógica de primeira ordem

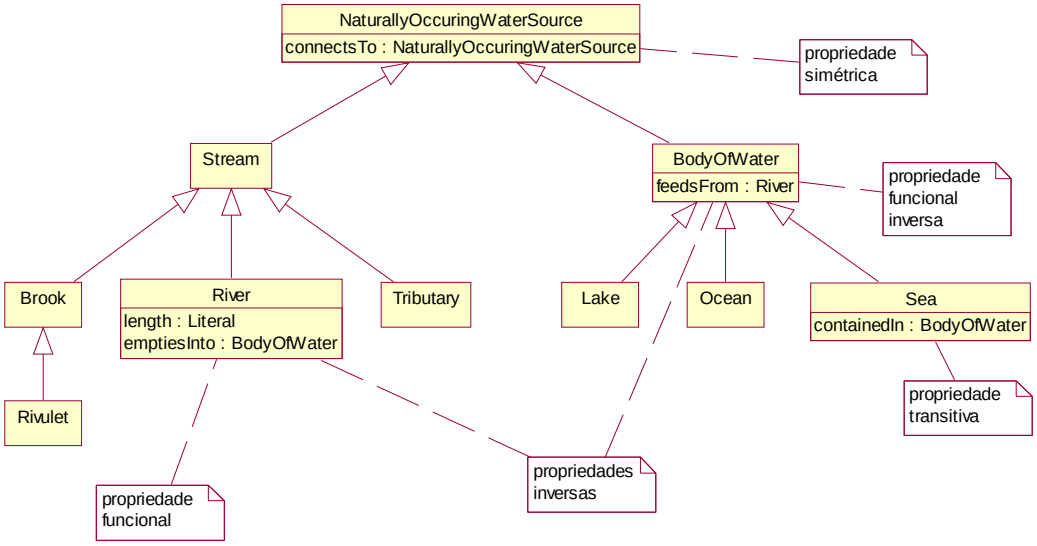


Figura 3.6 – Taxonomia representada em diagrama UML

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
       xmlns:owl="http://www.w3.org/2002/07/owl#"
       xmlns="http://www.xfront.com/owl/ontologies/water/#"
       xmlns:water="http://www.xfront.com/owl/ontologies/water/#"
       xml:base="http://www.xfront.com/owl/ontologies/water/">

   <owl:Class rdf:ID="NaturallyOccurringWaterSource">
      <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
   </owl:Class>

   <owl:Class rdf:ID="Stream">
      <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
   </owl:Class>

   <owl:Class rdf:ID="River">
      <rdfs:subClassOf rdf:resource="#Stream"/>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#connectsTo"/>
            <owl:someValuesFrom rdf:resource="#BodyOfWater"/>
         </owl:Restriction>
      </rdfs:subClassOf>
```

```
   </owl:Class>

   <owl:Class rdf:ID="Flueve">
      <rdfs:subClassOf rdf:resource="#River"/>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#emptiesInto"/>
            <owl:allValuesFrom rdf:resource="#Sea"/>
         </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class>

   <owl:Class rdf:ID="Brook">
      <rdfs:subClassOf rdf:resource="#Stream"/>
   </owl:Class>

   <owl:Class rdf:ID="Rivulet">
      <rdfs:subClassOf rdf:resource="#Brook"/>
   </owl:Class>

   <owl:Class rdf:ID="Tributary">
      <rdfs:subClassOf rdf:resource="#Stream"/>
   </owl:Class>

   <owl:Class rdf:ID="BodyOfWater">
      <owl:equivalentClass rdf:resource="http://www.other.org#WaterGeoFeature"/>
      <rdfs:subClassOf rdf:resource="#NaturallyOccurringWaterSource"/>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#maxDepth"/>
            <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardina
lity>
         </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class>

   <owl:Class rdf:ID="Lake">
      <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
   </owl:Class>

   <owl:Class rdf:ID="Ocean">
      <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
   </owl:Class>

   <owl:Class rdf:ID="Sea">
      <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
   </owl:Class>

   <owl:ObjectProperty rdf:ID="emptiesInto">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#River"/>
```

```
      <rdfs:range rdf:resource="#BodyOfWater"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:ID="feedsFrom">
      <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
      <owl:inverseOf rdf:resource="#emptiesInto"/>
      <rdfs:domain rdf:resource="#BodyOfWater"/>
      <rdfs:range rdf:resource="#River"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:ID="containedIn">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
      <rdfs:domain rdf:resource="#Sea"/>
      <rdfs:range rdf:resource="#BodyOfWater"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:ID="connectsTo">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
      <rdfs:domain rdf:resource="#NaturallyOccurringWaterSource"/>
      <rdfs:range rdf:resource="#NaturallyOccurringWaterSource"/>
   </owl:ObjectProperty>

</rdf:RDF>
```

Figura 3.7 – Taxonomia representada em OWL

É possível observar a forma mais rica que OWL traz para a definição de propriedades, que podem ser classificadas através de tipos. No exemplo, “emptiesInto” é uma propriedade funcional que relaciona entidades dos tipos “River” e “BodyOfWater”. Por sua vez, “feedsFrom” é definida como propriedade inversa de “emptiesInto”. Já “connectsTo e “containedIn” são definidas como propriedades simétrica e transitiva, respectivamente.

Fica claro o papel de OWL como linguagem de extensão de RDF *Schema*. Assim, somente dados em formato RDF podem ser associados a conceitos definidos em uma ontologia OWL. Continuamos, desta forma, com a mesma limitação para uso destas tecnologias, que sugerem que fontes de dados existentes sejam alteradas para suportar novos padrões. Na visão apresentada neste trabalho, esta dificuldade deve ser vencida não pela alteração de fontes de dados existentes, mas pelo uso de técnicas de extração de dados em um formado que possibilite o uso destas tecnologias para interoperabilidade semântica.

3.2 Extração de Dados e Interoperabilidade Semântica

Como foi mencionado, somente dados em formato RDF estarão disponíveis na Web Semântica que está por vir. O momento gerado pela indústria e a comunidade científica em torno da *Semantic Web* deve levar boa parte das novas fontes de dados construídas a fazer uso das tecnologias para interoperabilidade semântica, que têm como base fundamental o formato RDF.

Mesmo na visão utópica de que todas as novas fontes de dados que surgirem após o estabelecimento concreto da Web Semântica seguirão as novas diretrizes, temos ainda que considerar as fontes já existentes. Para estas, existem diferentes linhas de ação possíveis para se enquadrar aos novos padrões de interoperabilidade.

Fontes de dados Web preexistentes podem, por exemplo, alterar o conteúdo dos documentos para incluir informação RDF. No caso de fontes XML, a mudança necessária normalmente não é grande. No entanto, como os documentos XML podem ser utilizados por outras aplicações existentes, a mudança, mesmo pequena, pode não ser possível, pois afetaria estas aplicações clientes. No caso de fontes HTML, o processo de inclusão de dados RDF é chamado normalmente de anotação. No contexto da Web Semântica, fala-se bastante da anotação da páginas HTML com informação RDF acerca do documento como um todo, informação esta que irá viabilizar a próxima geração de ferramentas de busca na Web, mais precisas e inteligentes. Pouco se fala da disponibilização, em formato RDF, de unidades de dados individuais presentes em documentos HTML. Em qualquer um dos casos, existe o problema da redundância de dados, uma vez que informações presentes no conteúdo do documento terão que ser duplicadas nas anotações RDF. A conseqüência é o problema usual de manutenção da consistência em dados replicados.

No caso de documentos gerados automaticamente, pode-se alterar o sistema de geração dinâmica do conteúdo para que este gere, além do documento em si, dados RDF, agregados ao próprio documento ou como um documento em separado. O problema neste cenário é o elevado grau de mudança e o esforço associado.

Uma limitação comum a todas as soluções discutidas até agora é o fato delas serem dependentes de mudanças que têm que ser feitas pelo provedor da informação. Nem sempre o responsável por uma fonte de dados deseja tornar sua informação mais facilmente acessível. Pode ser que o benefício de tornar os dados disponíveis para cenários de integração não compense o esforço. Mesmo assim, pode haver o interesse de terceiros pela informação em questão. Seria então interessante ter uma estratégia de disponibilização das informações que não dependesse de alterações na fonte e conseqüente esforço por parte do provedor da informação.

Neste trabalho, defendemos soluções baseadas em extração de dados como mais adequadas para a tarefa de disponibilização de dados em formato RDF a partir de fontes Web preexistentes.

Como é possível perceber, a extração de dados funciona de forma externa e independente, possibilitando a obtenção de dados RDF sem alterar a fonte de dados em questão. Em nossa opinião, técnicas de extração de dados podem desempenhar um papel importante na construção da Web Semântica. Primeiramente, como solução de baixo impacto e fácil adoção por fontes já existentes, têm muito a contribuir na transição para a próxima geração da Web, ajudando a criar massa crítica de informações em formato RDF disponível. O efeito cascata, que leva à adoção de novas tecnologias por influência da adesão de outros, que tanto contribuiu para o crescimento da Web "tradicional", sem dúvida é um fator importante para o estabelecimento de fato da Web Semântica. À medida que aumenta a quantidade de dados RDF disponíveis na Web, surgem mais aplicações que utilizam esses dados, o que motiva outros provedores de informação a também disponibilizarem seus dados neste formato. É importante notar que é provável que muitas fontes de informação em formato HTML e XML existentes hoje nunca cheguem a realizar uma migração para os novos padrões de interoperabilidade propostos. Desta forma, técnicas de extração de dados tendem a ter papel fundamental como solução para a conexão deste futuro "legado" à nova Web Semântica.

Para assumir este papel de forma plena, algumas exigências passam a ser colocadas sobre as técnicas de extração de dados. Primordialmente, é necessário que a técnica de extração seja confiável, uma vez que outras camadas da infra-estrutura estarão dependendo dela. Isto aponta para a necessidade por técnicas de extração que sejam resistentes às mudanças de formato, comuns nas fontes Web de hoje. Em outras palavras, é reforçada a necessidade por técnicas de extração resilientes.

3.3 Exemplo de cenário de Interoperapilidade envolvendo Extração de Dados

Nesta seção, ilustraremos o uso de uma solução de extração de dados como parte de um cenário de interoperabilidade. Vale ressaltar que uma solução completa de interoperabilidade semântica está além do escopo deste trabalho. No entanto, a descrição de um cenário exemplo esclarece o papel da extração de dados na construção de soluções de interoperabilidade, uma das motivações deste trabalho.

Como exemplo motivador, imaginemos um conjunto de bibliotecas de uma determinada cidade, que desejam divulgar os livros que constam dos seus acervos. Consideremos quatro dessas bibliotecas. Uma delas possui a informação sobre seus livros em um banco de dados relacional, que possui facilidades para exportação de dados em formato XML. Outra possui um site Web em formato HTML, que traz informações sobre os livros do seu acervo. A terceira possui informação sobre seus livros em formato XML. A última, por sua vez, mantém dados sobre seu acervo em RDF, codificado no formato XML.

A figura abaixo descreve uma possível arquitetura de solução de interoperabilidade, envolvendo extração de dados, para essas quatro fontes de informação.

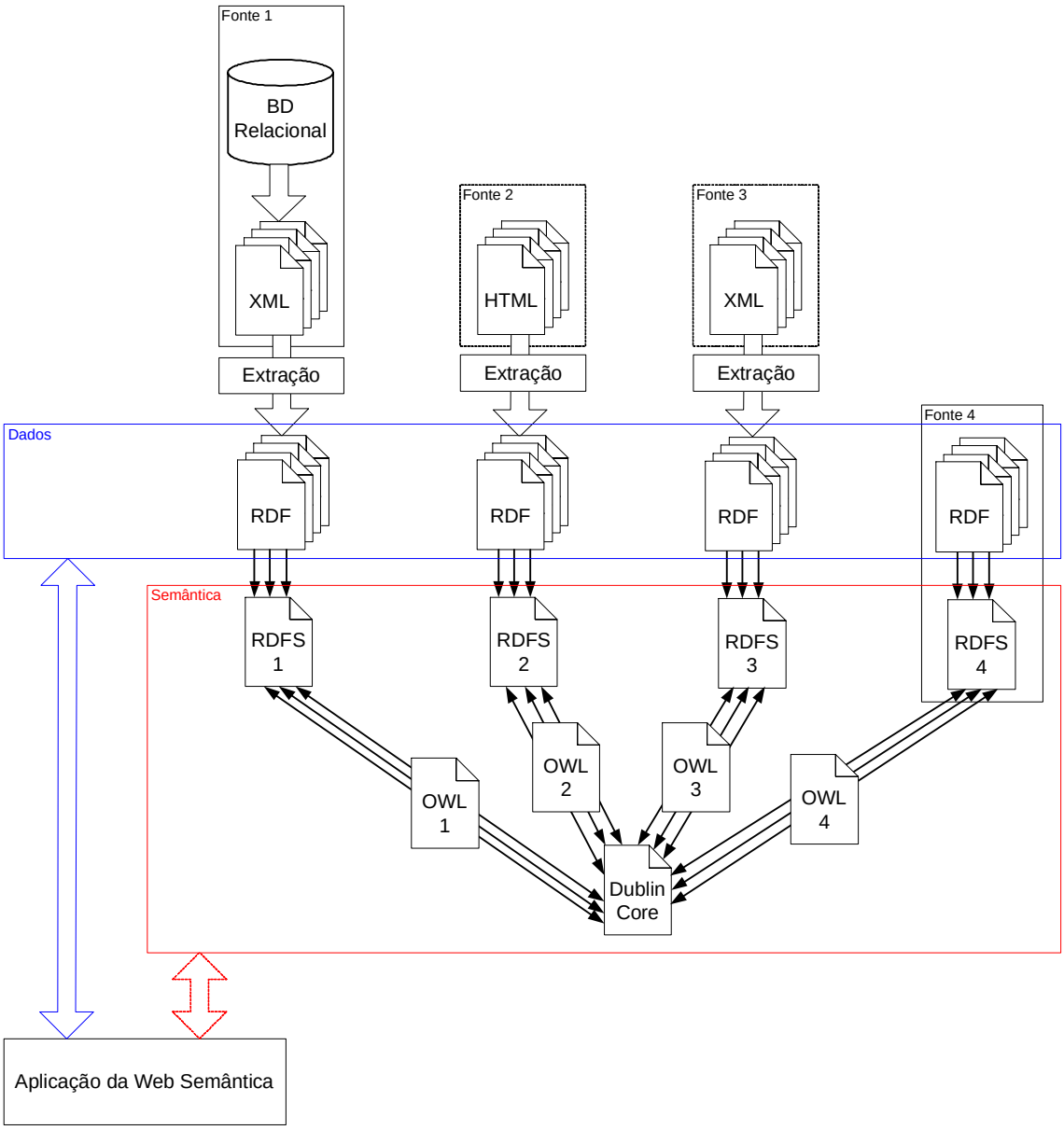


Figura 3.8 – Arquitetura de Interoperabilidade utilizando RDF

Nesta arquitetura, o papel dos módulos de extração é a geração de dados em formato RDF. Por uma questão de isolamento de responsabilidades, o extrator de cada fonte gera como saída dados RDF em um vocabulário específico, semelhante ao vocabulário da fonte em questão. Estes dados referenciam, desta forma, um RDF *Schema* específico para cada uma das fontes. Esta parte parte da arquitetura faz o papel de uma **camada de modelo de dados**, responsável por eliminar as peculiaridades de sintaxe e prover um modelo de dados para cada

fonte. No exemplo, a camada de modelo de dados corresponde aos documentos RDF, juntamente com os RDF *Schemas* RDFS 1, RDFS 2, RDFS 3 e RDFS 4.

Para viabilizar a interoperabilidade, o modelo de dados de cada fonte terá que ser relacionado com uma ontologia comum, que estabelece os conceitos e propriedades que são de entendimento consensual. No nosso exemplo, essa **camada de ontologia** é ocupada pelo conjunto de elementos do Dublin Core[13].

O relacionamento entre os modelos de dados das fontes e a ontologia central é feito através da **camada de mapeamento**, composta por quatro ontologias (OWL 1, OWL 2, OWL 3 e OWL 4, na figura), cada uma mapeando conceitos no modelo de dados de uma das fontes nos conceitos presentes na ontologia central, através de relações de equivalência.

Uma aplicação da Web Semântica, como, por exemplo, um agente tentando achar um livro específico para alguém, ou um *crawler* de uma ferramenta de busca da próxima geração, terá seu processo de obtenção de dados dividido em dois passos. No primeiro, a aplicação obterá os RDF *Schemas* das fontes, as ontologias de mapeamento, e a ontologia central. Essa informação servirá para que a aplicação tenha um entendimento dos dados disponibilizados pelas fontes em formato RDF, obtidos no segundo passo do processo. Neste contexto, chamamos de entendimento o estabelecimento de relações entre os dados obtidos e os conceitos definidos pelo Dublin Core. Desta forma, é possível reconhecer informações como títulos de livros, nomes de autores, etc, apesar dos diferentes vocabulários utilizados.

Para ilustrar a importância da separação entre as camadas de modelo de dados e de mapeamento, imagine que, uma vez estabelecida a infra-estrutura acima, queira-se relacionar os dados obtidos com uma nova ontologia. Se a expressividade da nova ontologia estiver contida na expressividade do Dublin Core, basta criar uma ontologia de mapeamento, contendo relações de equivalência entre os conceitos do Dublin Core e da nova ontologia, como mostrado na figura abaixo.

[13] Dublin Core é uma ontologia amplamente conhecida, que define cerca de 14 termos básicos para documentos e publicações, como “title”, “creator”, “subject”, “publisher”, etc, cada um identificado por uma URI da forma http://purl.org/dc/elements/1.1/title

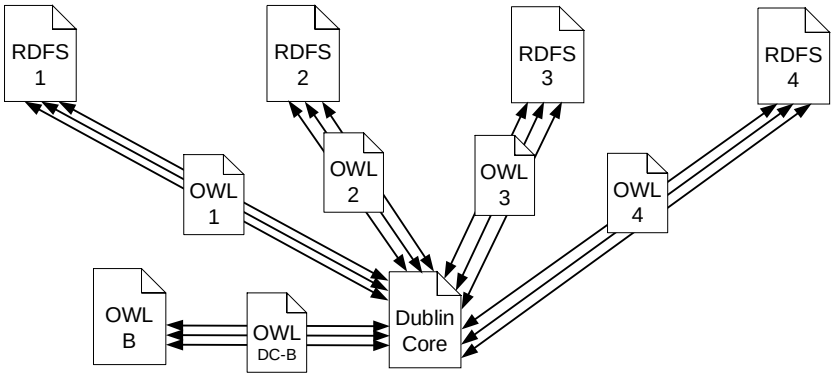


Figura 3.9 – Suporte a nova ontologia: caso 1

No entanto, se a nova ontologia for capaz de expressar conceitos que não estejam presentes no Dublin Core, mas que constem do modelo de dados das fontes, pode ser criado um novo conjunto de ontologias de mapeamento do modelo das fontes para a nova ontologia, como ilustrado na figura abaixo.

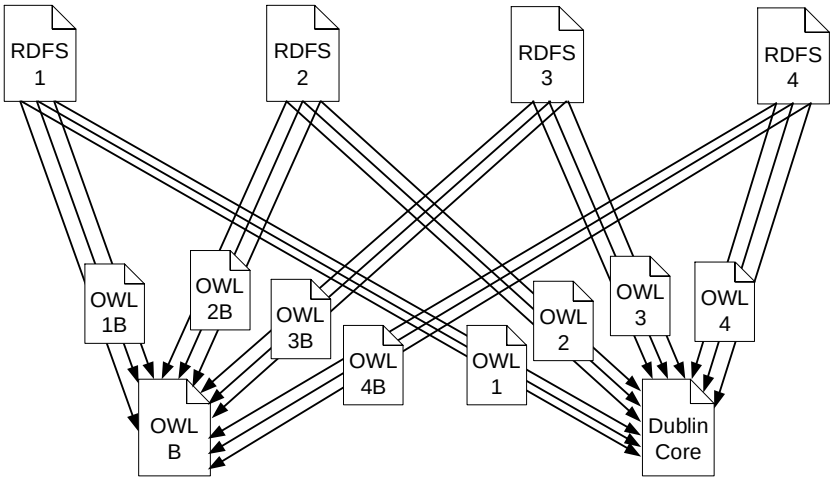


Figura 3.10 – Suporte a nova ontologia: caso 2

Em qualquer dos casos, é importante destacar que as tarefas de extração de dados não precisam ser alteradas, e que as duas ontologias podem ser suportadas simultaneamente.

3.4 Comparação entre RDF e XML como formatos de saída do processo de extração

Este trabalho adota RDF/XML[14] como formato de saída dos processos de extração de dados. A escolha entre RDF/XML e um formato XML tradicional, em diferentes contextos, tem gerado bastante discussão e dividido opiniões. Na verdade, RDF e XML são tecnologias complementares. A escolha uma ou outra, ou de uma combinação das duas, depende, muitas vezes, do contexto de aplicação específico que está sendo considerado. Enquanto XML tradicional é amplamente utilizado, tendo maior disponibilidade de ferramentas e exemplos, RDF permite que programas de manipulação tenham maior entendimento sobre o que os dados significam. Nesta sessão, apresentamos algumas vantagens trazidas pelo uso de RDF/XML em contextos de interoperabilidade entre fontes distintas.

RDF apresenta uma estrutura mais simples e manipulável.

- Um documento XML genérico traz informação estruturada na forma de uma árvore, na qual as partes que compõem entidades mais complexas devem ser declaradas dentro da estrutura desta. RDF apresenta um formato bem mais plano, composto por tripas que, apesar de independentes, relacionam-se e referenciam-se entre si. Este modelo menos hierárquico apresenta facilidades de manipulação em comparação com XML genérico, que pode apresentar hierarquias arbitrariamente complexas.
- Um exemplo da maior facilidade de manipulação de RDF é a construção e suporte a consultas. Devido às estruturas mais complexas, XML genérico

[14] RDF/XML é a serialização de RDF em sintaxe XML. RDF/XML é o formato mais comum para dados RDF, e é muitas vezes o que se quer dizer quando fala-se simplesmente RDF. No

traz maior dificuldade pra *parsing* e busca, que são facilitados na estrutura RDF.

- Apesar da estrutura simplificada, não há perda de expressividade no formato RDF, apenas padronização. Através da criação de relacionamentos na forma de propriedades baseadas em identificadores, é possível se libertar das estruturas hierárquicas impostas por XML genérico, e mesmo assim expressar relações entre as entidades descritas.

RDF permite entendimento parcial do modelo.

- Para utilizar dados XML através das tecnologias de manipulação genérica, é necessário que se tenha um entendimento do *Schema* completo. Através do modelo RDF, é possível utilizar somente as partes de informação que se entende, desprezando o restante.

RDF pode ter semântica associada

- Dados em formato RDF podem ser associados a conceitos expressos na forma de ontologias, através de RDF Schema ou linguagens mais ricas como OWL. Esta associação de dados com conceitos expressos de maneira formal é a base para a a interpretação automática dos dados por programas.

RDF traz Interoperabilidade Semântica, e não somente Sintática

- Utilizando XML genérico, se duas partes necessitam se comunicar, inicialmente elas precisam concordar em uma sintaxe comum para seus documentos. A dificuldade para se chegar a esse tipo de acordo é bastante conhecida. [Pow03] afirma que "é praticamente impossível fazer duas empresas concordarem com relação a o que os dados significam". Em um artigo sobre RDF, Tim Bray[15] afirma que "se bibliotecas, que existem há

---

entanto, não se deve esquecer que RDF/XML é apenas uma das possíveis serializações de RDF.

[15] Tim Bray, um canadense com 20 anos de carreira em Tecnologia da Informação, foi co-editor das especificações XML 1.0 e Namespaces, do W3C.

centenas de anos, não são capazes de concordar sobre um padrão único, não há muita chance de que a Web irá" [Bra03].

- Ao invés da interoperabilidade baseada em um padrão de documentos único, imposta por XML, RDF possibilita a interoperabilidade entre partes que utilizam vocabulários distintos, através do conceito de equivalências, que podem ser expressas através de ontologias de mapeamento, como no exemplo apresentado na seção anterior.
- Uma alternativa para a convivência entre formatos XML distintos é o uso de transformações XSLT ou consultas XQuery para realizar traduções para o padrão central (se existir), ou entre dois formatos existentes. Outra abordagem possível é a da tradução de consultas XQuery realizadas sobre o modelo central para consultas no modelo dos documentos XML originais. Em qualquer dos casos, no entanto, a complexidade e esforço serão muito maiores do que os relativos à manutenção de ontologias de mapeamento na arquitetura RDF.

Como não poderia deixar de ser, o uso de RDF também traz dificuldades adicionais, que o tornam não indicado para alguns cenários de aplicação.

Uso de RDF requer maior planejamento

- Normalmente, a concepção de um modelo de dados RDF exige maior investigação e planejamento. XML genérico pode ser gerado e utilizado por aplicações de uma forma mais *ad hoc*, para rápida prototipação e desenvolvimento

Nem todo contexto de aplicação é adequado para o uso de RDF

- O uso de RDF introduz complexidade adicional, que pode não ser necessária em alguns contextos para se alcançar os objetivos definidos. Em alguns casos, XML genérico será mais indicado, enquanto em outros RDF será apropriado. Alguns projetos podem fazer uso tanto de XML tradicional quanto de RDF, para atingir diferentes objetivos.

3.4 Trabalhos que abordam a Resiliência na Extração

Um processo de extração pode falhar quando a fonte em questão altera o formato dos documentos que retorna. Em *sites* web que retornam documentos HTML, isso ocorre freqüentemente, devido a remodelagens periódicas da interface com o usuário. O mesmo ocorre com fontes que retornam documentos XML, que podem ter seu formato alterado devido a novas necessidades do negócio, como mudança de vocabulário (nomes de tags), inclusão ou retirada de informações, reestruturação ou adoção de um novo "padrão" de codificação dos documentos.

A tarefa de verificação da extração consiste em determinar se um processo de extração sobre uma fonte Web ainda está funcionando corretamente, isto é, se ainda consegue realizar extrações, trazendo dados corretos e não fragmentos de texto que não correspondem aos dados.

A verificação da extração é, na verdade, um sub-problema de um problema mais amplo, que é a manutenção da extração. A rotina de verificação deve informar se o processo de extração ainda funciona corretamente, ou encontra-se em estado de falha. Manutenção é a tarefa de recuperar um processo de extração em estado de falha, de modo que este volte a operar corretamente. Manutenção da extração é o foco principal deste trabalho.

A seguir, é apresentada uma visão geral de trabalhos relacionados, que atacam os problemas da verificação e manutenção da extração.

3.4.1 RAPTURE

Em [Kus00a], Kushmerick propõe uma solução para o problema da verificação da extração como extensão do paradigma do teste regressivo [Bei95]. Para ilustrar o conceito, primeiro é apresentado um algoritmo mais simples, denominado EXACT-MATCH. A idéia é fornecer ao processo de extração um documento do qual se conhece os dados, e checar se os dados extraídos correspondem aos dados esperados. Para verificar um processo de extração, EXACT-MATCH realiza a extração sobre um documento retornado por uma certa consulta, e também sobre outro retornado anteriormente pela mesma

consulta, quando sabia-se que o processo de extração estava funcionando corretamente. O processo de extração é considerado operacional quando as duas extrações apresentam resultados idênticos. Devido à simplicidade, EXACT-MATCH apresenta alto índice de erro, acusando muitas vezes falha no processo de extração, quando na verdade este continua operando corretamente. O motivo para o elevado índice de erro é a restrição de igualdade imposta, demasiadamente severa, uma vez que nem sempre uma fonte apresenta os mesmos dados para uma mesma consulta submetida posteriormente. Quando os dados em si mudam, e não o formato do documento, a verificação acusa falha erroneamente.

Após esta introdução, Kushmerick apresenta o algoritmo RAPTURE [Kus00a], cuja idéia central é apreender um modelo probabilístico dos dados extraídos durante um período de treinamento, quando se sabe que o processo de extração está funcionando corretamente. Este modelo captura várias propriedades dos dados extraídos, como quantidade média de palavras, comprimento médio em caracteres, fração de caracteres numéricos presentes, densidade de *tags* HTML, entre outros. Para fazer a verificão do processo de extração, os novos dados extraídos são avaliados contra o modelo probabilístico apreendido, para estimar a probabilidade do processo de extração estar operando corretamente. Se esta probabilidade for menor do que o limite estipulado, assume-se falha no processo de extração.

3.4.2 DataPro

Em [LM00] é apresentado DataPro, um algoritmo para o aprendizado de protótipos de dados (*data prototypes*), através da localização de seqüências de *tokens* estatisticamente significativas. Uma seqüência é considerada significativa se ocorre mais freqüentemente do que se esperaria se os *tokens* fossem gerados de forma randômica e independente. Protótipos gerados através do algoritmo DataPro podem ser utilizados tanto na verificação como na manutenção da extração.

Inicialmente, um conjunto de consultas é utilizado para obter um conjunto de documentos que servirão como exemplos de treinamento. A partir destes

exemplos, DataPro aprende padrões para os inícios e fins de atributo, na forma de um modelo de distribuições probabilísticas estes padrões. Para verificar o funcionamento do *wrapper* para um conjunto de consultas, realiza-se a extração e compara-se as distribuições dos resultados com aquelas do treinamento. Se a diferença estiver fora de um limite máximo permitido, a extração é considerada falha.

Para a manutenção da extração, o trabalho é direcionado à geração automática de exemplos de treinamento para sistemas de indução de *wrappers*, como [Kus97] e [MMK99]. Quando o *wrapper* é verificado como falho, o modelo probabilístido aprendido por DataPro é usado para identificar prováveis fragmentos que correspondem aos dados nos novos documentos. Estes fragmentos e suas respectivas probabilidades, após processados para remoção de ruído, vão formar, juntamente com seus respectivos documentos de origem, a entrada para o algoritmo de indução do *wrapper*. Cada documento, juntamente com exemplos de dados a extrair contidos no mesmo, forma o que se chama um exemplo anotado para o algoritmo de aprendizado mecânico.

3.4.3 RoadRunner

RoadRunner [CMM01] é uma ferramenta para a geração totalmente automática da configuração necessária para a extração de dados. O princípio utilizado é a comparação entre a estrutura de dois ou mais documentos pertencentes à mesma “classe” de documentos, e a conseqüente geração de um esquema para os dados presentes em documentos desse tipo, e de um modelo de extração, capaz de extrair atributos e construir tuplas cuja estrutura corresponde ao esquema. A geração do esquema e do modelo de extração é baseada em um algoritmo que compara a estrutura e conteúdo dos documentos e constrói expressões regulares a partir das diferenças encontradas. Resumidamente, diferenças de *strings* (não envolvendo *tags*) são utilizadas para descobrir atributos, enquanto diferenças envolvendo *tags* apontam dados opcionais, ou multivalorados.

3.3.4 BYU

No trabalho desenvolvido pelo Data Extraction Group da Brigham Young University [ECJ+99], o processo de extração é baseado em um modelo conceitual, construído previamente na forma de uma ontologia. Nela são descritos os dados de interesse, incluindo relacionamentos, presença em léxicos específicos, e palavras-chave do contexto.

Antes da aplicação da ontologia para classificar os dados, é executado um outro procedimento que subdivide o documento em pedaços menores, cada um contendo um "registro". O uso de múltiplas heurísticas para determinar os separadores de registro garante precisão satisfatória ao processo.

Após a separação dos registros não estruturados, os mesmos são tratados para geração dos registros propriamente ditos, em dois passos. O primeiro produz uma tabela temporária contendo conjuntos descritor / *string* / posição para atributos candidados, localizados utilizando a ontologia. No segundo passo, heurísticas são aplicadas para filtrar os candidatos e gerar os registros propriamente ditos.

3.4.5 Quadro comparativo dos trabalhos

A seguir é apresentado um quadro comparativo dos trabalhos discutidos, destacando o suporte a um conjunto de funcionalidades que julgamos destacadamente importantes para técnicas que se propõem a tratar questões relacionadas à resiliência na extração de dados de fontes Web.

	Suporte à fontes não-HTML	Saída XML / RDF	Verificação da Extração	Manuten-ção da Extração	Nível de Automação
RAPTURE	parcial	não	sim	não	-
DataPro	parcial	não	sim	sim	manual
RoadRunner	nenhum	não	sim	sim	automatico
BYU	total	não	sim	sim	manual

Tabela 3.1 – Quadro comparativo dos trabalhos que abordam resiliência na extração

Na primeira coluna, está assinalado o grau de independência da marcação HTML apresentada por cada técnica. O valor “nenhum” assinala uma técnica que funciona somente com marcação HTML, “parcial” indica que a técnica suporta outras linguagens de marcação, como XML, e total significa que a técnica funciona mesmo na ausência total de marcação, ou seja, com texto livre. A última coluna marca o nível de automação do processo de manutenção da extração. Neste quesito, o destaque vai para o trabalho RoadRunner, único que apresenta uma automatização para o processo de manutenção. Vale destacar que técnicas baseadas em ontologias, como [ECJ+99], sofrem da necessidade de geração e manutenção manuais da(s) ontologia(s).

Capítulo 4

Uma Abordagem para Extração Resiliente

4.1 Introdução

Até esse ponto deste trabalho, a discussão sobre extração de dados de fontes Web mencionou apenas superficialmente um importante problema: a mudança do formato exportado pela fonte alvo da extração. No cenário característico, o software cliente do processo de extração, seja ele um sistema de integração de informações ou um agente, não exerce nenhum controle sobre as fontes de dados consultadas.

No caso de *sites* Web, que publicam seu conteúdo na forma de páginas HTML, mudanças de estrutura são freqüentes. Em documentos HTML, os dados estão misturados a diretivas de apresentação, que determinam aspectos visuais da página, como *design* e diagramação. Os aspectos de apresentação da interface com o usuário apresentam alta freqüência de mudança na maioria das aplicações, e de uma forma ainda mais acentuada nas interfaces HTML.

Para fontes Web que retornam documentos XML como resposta às interações, também podem ocorrer mudanças na estrutura dos documentos retornados. Caracteristicamente, as fontes alvo dos processos de extração são independentes, não tendo obrigação de informar aos extratores no caso de decidirem mudar a estrutura dos documentos XML que retornam.

Como discutimos, os processos de extração guiam-se por informações estruturais e, no caso de documentos HTML, em informações de formatação. O problema é que, se a fonte modificar a estrutura ou formatação de seus documentos resposta, os detalhes que guiavam o processo de extração podem

passar a não mais valer, e o processo falhará. Para que isso não acontecesse, necessitaríamos que o processo de extração de dados apresentasse **resiliência**, isto é, a capacidade de continuar a funcionar corretamente na ocorrência de mudanças nos documentos alvo da extração. Para tal, o processo teria que ser capaz de reconhecer uma situação de falha, e se reconfigurar automaticamente.

Para conseguirmos resiliência, é preciso resolver dois problemas distintos. O mais simples deles é o que chamamos de **verificação da extração**, que seria determinar se o processo de extração ainda está operando corretamente.

Como discutido anteriormente, um processo de extração baseia-se em um modelo de extração, que é um conjunto de regras que configuram o programa genérico para fazer extração de dados de uma fonte em particular. O modelo de extração, que pode ser imaginado como um arquivo de configuração, descreve como obter os dados de interesse, considerando as particularidades de estrutura e formatação da fonte em questão. Se os documentos exportados pela fonte mudarem seu formato de uma maneira que inviabilize a extração, o processo de verificação da extração indicará a situação de falha. Surge então o desafio de ajustar o processo de extração à mudança ocorrida na fonte, o que significa ajustar o modelo de extração. Tal ajuste pode ser feito via intervenção humana, mas nosso interesse é a automatização do processo, ou seja, a **manutenção automática da extração**. Isto pode ser conseguido se conseguirmos ajustar automaticamente o modelo de extração, ou gerar um novo modelo de extração de forma automatizada. Este trabalho adota o caminho da geração automática de um novo modelo de extração.

A idéia por trás da proposta de reescrita automática do modelo de extração apresentada neste trabalho é fazer o caminho inverso do processo de extração. A base do processo está em, a partir de um valor esperado de um atributo, extrair sua vizinhança (ou sua informação posicional, no caso de um modelo de extração posicional). O conjunto das vizinhanças obtidas para um determinado atributo, assim como o conjunto de valores encontrados para o mesmo, são usadas no processo de construção do novo modelo de extração.

Entretanto, como obter valores esperados para as informações que se deseja extrair? Esta é outra questão chave da abordagem proposta. A solução está na catalogação de consultas realizadas à fonte, juntamente com seus valores retornados.

Assim sendo, este trabalho baseia o processo de reconfiguração automática da extração em dados previamente extraídos da fonte em questão, quando a extração funcionava corretamente. Sendo baseada unicamente nos dados, a abordagem proposta apresenta real independência do formato de publicação das fontes.

Neste capítulo, é apresentada inicialmente uma formalização para os conceitos de extração de dados. Esta base é então utilizada para a formalização dos problemas relacionados à extração de dados resiliente. Finalmente, é proposta uma abordagem para solução destes problemas, baseada nos dados catalogados e em reconsultas à fonte em questão.

4.2 Uma Formalização para os Conceitos de Extração de Dados

Informalmente, podemos dizer que o processo de extração de dados consiste na obtenção de um conjunto de tuplas do mesmo tipo contidas em um documento resposta retornado da fonte textual em questão, e a conseqüente disponibilização destas tuplas em formato conveniente.

Nesta seção, será apresentada uma abordagem mais formal para os conceitos discutidos até agora, através do estabelecimento de algumas definições que serão utilizadas no restante do trabalho.

4.2.1 Modelo Abstrato de Dados

Adotamos como modelo abstrato de dados um modelo relacional estendido, no qual tuplas são formadas por atributos, sendo que um determinado tipo de atributo pode ocorrer 0, 1 ou várias vezes em uma tupla.

Uma instância de tupla t é formada por instâncias de atributos $a_1, ..., a_k$. Dentre estas instâncias de atributos, sequências a_r, a_{r+1} , ..., a_s correspondem ao mesmo tipo de atributo A_i.

Uma instância de tupla t é de um tipo de tupla T , definido como se segue.

Definição 4.1. O tipo de tupla T (ou esquema relacional estendido) é definido através da representação abstrata

$$T = (A_1, A_2, ..., A_n) ,$$

onde A_i representa um tipo de atributo A_i presente em T.

Um tipo de atributo A_i é definido como se segue.

Definição 4.2. O tipo de atributo A_i é definido através da representação abstrata

$$A_i = (A_i^{\text{label}}, A_i^{\text{dataType}}, A_i^{\text{minOccurs}}, A_i^{\text{maxOccurs}}) ,$$

onde A_i^{label} é o rótulo que identifica A_i, possivelmente trazendo algum valor semântico acerca do papel de A_i em T, A_i^{dataType} determina o tipo de dado de A_i, ou seja, uma referência ao conjunto contável de possíveis valores para instâncias de A_i , e $A_i^{\text{minOccurs}}$ e $A_i^{\text{maxOccurs}}$ são respectivamente os números mínimo e máximo permitidos de ocorrências de A_i no tipo de tupla T correspondente.

Uma instância de tupla t do tipo T é definida a seguir.

Definição 4.3. A instância de tupla t é definida como um conjunto de instâncias de atributo, da forma

$$t = \{ a_1, ..., a_k \}.$$

O conjunto de instâncias $\{ a_1, ..., a_k \}$ pode ser dividido em sequências de instâncias de atributo a_r, a_{r+1} , ..., a_s correspondentes ao mesmo tipo de atributo A_i, que representamos como A_i^t. Assim, t pode ser definida também através da representação abstrata

$$t = (A_1^t, A_2^t, ..., A_n^t) ,$$

onde A_i^t corresponde ao conjunto de instâncias de atributo presentes na tupla t que são do tipo de atributo A_i .

De uma maneira mais formal, A_i^t é definido como a seguir.

Definição 4.4. O conjunto A_i^t de instâncias de atributo do tipo de atributo A_i presentes na instância de tupla t é definido como

$$A_i^t = \{ a_j \mid \quad a_j \in t \quad \text{e} \quad a_j \text{ é do tipo } A_i \}.$$

O conjunto A_i^t é, portanto, um subconjunto de t, ou seja

$$A_i^t = \{ a_r, a_{r+1}, ..., a_s \} \subset t .$$

Uma instância de atributo a_j que compõe t é definida da maneira seguinte.

Definição 4.5. Uma instância de atributo a_j é definida como

$$a_j = (A_i, a_j^{\text{value}}) ,$$

onde A_i é a referência ao tipo de atributo de a_j, e a_j^{value} é o valor da instância de atributo a_j. Usando como referência o próprio *label* do tipo de atributo A_i^{label}, chega-se à representação abstrata

$$a_j = (A_i^{\text{label}}, a_j^{\text{value}}) .$$

A partir desta representação abstrata para uma instância de atributo, é possível desenvolver uma outra representação abstrata para uma instância de tupla t, que é dada a seguir.

Definição 4.6. Como vimos, uma instância de tupla t pode ser definida nas duas formas abaixo.

$$t = \{ a_1, ..., a_k \} \quad \text{e}$$

$$t = (A_1^t, A_2^t, ..., A_n^t) .$$

Usando a representação da definição 4.5 para instâncias de atributos, teremos

$$t = \{ (A_1^{\text{label}}, a_1^{\text{value}}) , (A_1^{\text{label}}, a_2^{\text{value}}) , ... , (A_i^{\text{label}}, a_j^{\text{value}}) , ... , (A_n^{\text{label}}, a_k^{\text{value}}) \} ,$$

onde a_j é do tipo A_i para todo $(A_i^{\text{label}}, a_j^{\text{value}})$ pertencente a t.

A representação acima constitui a representação para uma instância de tupla no modelo abstrato de dados apresentado.

4.2.2 Codificação de Marcação

Como foi dito, as instâncias de tupla presentes em um documento resposta encontram-se codificadas, sendo parte integrante de um documento escrito em uma linguagem de marcação, HTML ou XML. Definimos a codificação de marcação de uma instância de tupla construtivamente, começando pelas instâncias de atributos.

A codificação de uma instância de atributo, representada por a_j^{enc}, é definida a seguir.

Definição 4.7. A codificação a_j^{enc} de uma instância de atributo a_j é definida como

$$a_j^{enc} = a_j^{start} \,.\, a_j^{value} \,.\, a_j^{end} \,,$$

onde a_j^{start} e a_j^{end} são os delimitadores anterior e posterior de a_j^{enc}, respectivamente, e "." representa a operação de concatenação de *strings*. Os delimitadores correspondem a um determinado número de *tokens* de texto imediatamente adjacentes ao valor da instância de atributo a_j^{value}.

Um *token* de texto é um fragmento de uma *string* maior, reconhecido pela presença de caracteres que são considerados separadores de *tokens*. Em se tratando de codificação de marcação, possíveis separadores de tokens são "<", ">" e espaço em branco.

A codificação de um conjunto A_i^t de instâncias de um tipo de atributo A_i presentes em uma tupla t, representada por $[A_i^t]^{enc}$, é definida a seguir.

Definição 4.8. A codificação $[A_i^t]^{enc}$ de um conjunto de instâncias de um tipo de atributo A_i presentes em uma tupla t é definida como

$$[A_i^t]^{enc} = a_r^{enc} \,.\, a_{r+1}^{enc} \,.\, \ldots \,.\, a_s^{enc} \,,$$

onde $A_i^t = \{ a_r, a_{r+1}, \ldots, a_s \}$, e "." representa a operação de concatenação de *strings*.

Finalmente, temos a definição da codificação de uma instância de tupla, representada por t^{enc}.

Definição 4.9. A codificação t^{enc} de uma instância de tupla t é uma concatenação de $[A_1^t]^{enc}$, $[A_2^t]^{enc}$, ... , $[A_n^t]^{enc}$, em uma ordem arbitrária.

A definição acima deixa claro que a ordem dos tipos de atributo pode mudar nas codificações de diferentes instâncias de tupla.

A definição para um documento d em codificação de marcação é dada a seguir.

Definição 4.10. Um documento d em codificação de marcação é definido como

$$d = s_0 \,.\, t_1^{enc} \,.\, s_1 \,.\, t_2^{enc} \,.\, \ldots \,.\, s_{m-1} \,.\, t_m^{enc} \,.\, s_m \,,$$

onde s_0, s_1, ... , s_m são fragmentos de texto do documento d que não correspondem a instâncias de tupla codificadas. Cada s_q é uma *string* arbitrária, podendo inclusive ser a *string* vazia.

4.2.3 Extração de Dados

Realizar a extração dos dados contidos no documento d consiste em, a partir do texto de d, que contém instâncias codificadas de tupla t_1^{enc}, t_2^{enc}, ... , t_m^{enc}, obter as instâncias de tupla t_1 , t_2 , ... , t_m em suas representações abstratas.

Para montar a representação abstratata de uma instância de tupla t, será necessário, portanto, obter o valor a_j^{value} de cada instância de atributo a_j pertencente a t, assim como o *label* do tipo de atributo correspondente, A_i^{label}, onde $a_j \in A_i^t$. Quanto aos *labels*, que permitem associar uma instância de atributo com o tipo de atributo correspondente, não á garantias que os mesmos serão preservados no processo de codificação. Já os valores, presentes na codificação, precisam ser localizados em meio aos outros fragmentos de texto.

O processo de extração necessita, portanto, de informações que o auxiliem na localização e reconhecimento de instâncias de atributos codificadas. Estas informações estão presentes no descritor de extração, como veremos a seguir.

Uma vez reconhecidas e obtidas as instâncias de atributos, é de responsabilidade do processo de extração agrupá-las novamente em tuplas. Isto é feito por um algoritmo que normalmente se baseia em um critério de proximidade entre as instâncias de atributos obtidas. Por isso, o passo de obtenção das instâncias de atributos deve marcar em cada instância obtida a posição absoluta no texto do documento *d* em que a mesma foi localizada. Temos assim o conceito de instância de atributo obtida de documento, definida a seguir.

Definição 4.11. Uma instância de atributo obtida a_j^d, recuperada a partir da instância codificada a_j^{enc}, localizada no documento *d*, é definida como

$$a_j^d = (A_i^{label}, a_j^{pos}, a_j^{start}, a_j^{value}, a_j^{end}) ,$$

onde A_i^{label} é o *label* que identifica o tipo de atributo de a_j, a_j^{value} é o valor da instância de atributo a_j, a_j^{pos} é a posição absoluta de a_j^{enc} no texto de *d*, a_j^{start} e a_j^{end} são respectivamente os delimitadores anterior e posterior que compõem, juntamente com a_j^{value}, a instância codificada a_j^{enc}.

4.2.4 Fonte de Dados Textual em Codificação de Marcação

A seguir é dada uma definição para o conceito de tipo de documento em codificação de marcação.

Definição 4.12. Sejam d_1 e d_2 documentos em linguagem de marcação e os conjuntos de instâncias de tuplas ST_1 e ST_2 tais que

$$t_i \in ST_1 \Rightarrow t_i^{enc} \in d_1 \text{ e}$$

$$t_j \in ST_2 \Rightarrow t_j^{enc} \in d_2 .$$

Os documentos d_1 e d_2 são do mesmo tipo de documento *D* se as instâncias de tuplas de ST_1 e ST_2 forem todas do mesmo tipo de tupla *T*, e se a codificação utilizada nas instâncias de tupla codificadas presentes em d_1 e d_2 for similar, ou

seja, se para cada tipo de atributo A_i presente no tipo de tupla T, existe uma expressão de padrão capaz de reconhecer as ocorrências de A_i nas tuplas de ST_1 e ST_2.

Esta expressão de padrão é exatamente o que chamamos de padrão de extração de atributo, ou PEA, mencionado anteriormente, e que voltaremos a discutir com mais detalhes em breve.

Uma fonte de dados textual em codificação de marcação F é um processo computacional interativo que, em resposta a uma consulta q, retorna um documento textual em codificação de marcação d.

Para uma determinada fonte F, consultas de um determinado tipo Q retornam documentos de um determinado tipo D, que contêm instâncias codificadas de tuplas do tipo T.

Duas consultas q_1 e q_2 de um mesmo tipo Q diferenciam-se somente pelo conjunto do que chamamos parâmetros de consulta. Uma consulta q possui parâmetros de consulta $\{p_1, p_2, \ldots, p_k\}$, e os valores dos parâmetros deste conjunto forma um critério de consulta, que determinará quais instâncias de tuplas estarão presentes, codificadas, no documento resposta d.

4.2.5 Descritor de Extração e Modelo de Extração

Para que o processo de extração seja capaz de obter instâncias de tuplas a partir do documento D, que contém suas codificações, este necessita de um modelo de extração, definido a seguir.

Definição 4.13. O modelo de extração M_T^D, que configura a extração de instâncias de tuplas do tipo T a partir de documentos do tipo D, é definido como

$$M_T^D = [\ T\ ,\ E_T^D\ ]\ ,$$

onde E_T^D é o descritor de extração de T em D.

O modelo de extração é formado, portanto, pela definição do tipo de tupla da extração, juntamente com o descritor de extração.

O descritor de extração de T em D é definido a seguir.

Definição 4.14. O descritor de extração E_T^D, que descreve a extração de instâncias de tuplas do tipo T a partir de documentos do tipo D, é definido como

$$E_T^D = ([A_1^D]^{\text{pattern}} , [A_2^D]^{\text{pattern}} , \ldots , [A_n^D]^{\text{pattern}}) ,$$

onde $[A_i^D]^{\text{pattern}}$ é o padrão de extração do tipo de atributo A_i capaz de reconhecer instâncias de atributo deste tipo em documentos do tipo D.

Finalmente, o conceito de padrão de extração de tipo de atributo é definido a seguir.

Definição 4.15. O padrão de extração de atributo $[A_i^D]^{\text{pattern}}$ do tipo de atributo A_i no tipo de documento D é definido como

$$[A_i^D]^{\text{pattern}} = [A_i^D]^{\text{start}} . [A_i^{\text{dataType}}]^{\text{pattern}} . [A_i^D]^{\text{end}}$$

onde $[A_i^D]^{\text{start}}$ é a expressão de padrão[16] capaz de reconhecer os delimitadores anteriores de instâncias de A_i, em documentos do tido D, $[A_i^D]^{\text{end}}$ é a expressão de padrão capaz de reconhecer os delimitadores posteriores de instâncias de A_i em documentos do tido D, e $[A_i^{\text{dataType}}]^{\text{pattern}}$ é a expressão de padrão capaz de reconhecer valores do tipo de dado de A_i.

4.3 Uma Formalização para os problemas relativos à Extração Resiliente

4.3.1 Verificação da Extração

Seja ST o conjunto de instâncias obtidas por uma execução do processo de extração de T em D, ou seja

$$ST = \{ t_1 , t_2 , \ldots , t_m \} .$$

Verificação da extração é o processo que, dado ST, informa se a extração deve ser considerada bem sucedida ou mal sucedida. Esta conclusão é tomada com base em regras que determinam condições para que o conjunto ST seja

[16] Conforme mencionado no capítulo 2, expressões de padrão são *strings* utilizadas por rotinas de busca baseada em padrões, que tentam encontrar, dentro de um conteúdo textual, fragmentos que correspondem ao padrão dado pela expressão.

considerado aceitável, e a extração que o gerou, portanto, bem sucedida. Cada processo de extração, ao ser configurado, determina seu conjunto de regras de verificação $R = \{ r_1 , r_2 , ... , r_k \}$, onde cada regra r_i consiste de uma expressão condicional, que dependendo de características de *ST*, resultará como verdadeira ou falsa. Tais regras consideram, entre outras coisas, a adequação das tuplas de *ST* ao tipo de tupla esperado *T*, determinado pelo modelo de extração M_T^D.

Se todas as regras $r_1 , r_2 , ... , r_k$ que compõem *R* resultarem verdadeiras para *ST*, o processo de extração é considerado bem sucedido. Colocando de outra forma, se pelo menos uma das regra r_i de *R* resultar falsa para *ST*, o processo de extração é considerado em situação de falha.

4.3.4 Manutenção da Extração

Como vimos na definição 4.13, o modelo de extração M_T^D para instâncias de tupla do tipo *T* em documentos do tipo *D* é dado por $M_T^D = [T , E_T^D]$, onde *T* é a representação abstrata do tipo de tupla e E_T^D é o descritor de extração de *T* em *D*.

Em um cenário de extração de dados resiliente, manutenção da extração é o processo, disparado quando o processo de verificação da extração detecta falha no processo de extração, que tem por objetivo atualizar o modelo de extração, de modo a fazer o processo de extração voltar a funcionar satisfatoriamente, ou seja, de forma que o mesmo volte a realizar extrações que sejam consideradas bem sucedidas pelo processo de verificacão da extração.

Caracteristicamente, a atualização no modelo de extração compreenderá uma atualização no descritor de extração E_T^D, visto que a mudança causadora da falha terá relação com a codificação dos documentos. O que ocorre nestes casos é que o tipo de documento retornado pela fonte mudou de *D* para um certo $D' \neq D$, e precisamos gerar o $E_T^{D'}$ correspondente, de modo a atualizar o modelo de extração, que passará a ser $M_T^{D'} = [T , E_T^{D'}]$.

Como vimos na definição 4.14, $E_T^{D'}$ será formado pelos padrões de extração de atributo, na forma $E_T^{D'} = ([A_1^{D'}]^{pattern} , [A_2^{D'}]^{pattern} , ... , [A_n^{D'}]^{pattern})$. A tarefa de gerar um novo descritor de extração consiste então em gerar um novo padrão de

extração $[A_i^{D'}]^{pattern}$ para cada tipo de atributo A_i. É importante destacar que para gerar um novo padrão de extração de atributo $[A_i^{D'}]^{pattern}$, o padrão de extração de atributo antigo do mesmo atributo, $[A_i^{D}]^{pattern}$, não tem serventia nenhuma, já que o novo tipo de documento *D'* pode ser arbitrariamente diferente de *D*, o que representa na verdade a raiz do problema.

Se a técnica de manutenção da extração permitir, o processo pode ser capaz de se recuperar também de pequenas alterações no tipo de tupla, que venha a mudar de *T* para um *T'* similar, porém diferente de *T*. Nestes casos, o descritor de extração também terá que ser atualizado, e portanto o modelo de extração será atualizado duplamente, passando a ser $M_{T'}^{D'} = [\ T'\ ,\ E_{T'}^{D'}\ ]$.

4.4 Técnica das Reconsultas para Extração de Dados Resiliente

4.4.1 Visão Geral

A seguir descrevemos, em linhas gerais, a técnica das reconsultas, que permite detecção de falha e manutenção automática do processo de extração.

Consideremos inicialmente um processo de extração em funcionamento normal, realizando extrações de forma correta. Assumimos que a configuração inicial do processo de extração foi feita através de um processo semi-automatizado, que gera o modelo de extração a partir de exemplos de resultados de extração. Este processo semi-automatizado, que utiliza os mesmos princípios da solução de manutenção da extração, será explicado após esta, na seção 4.4.2, o que facilitará a compreensão.

O modelo de extração M_T^D gerado inicialmente configura o processo para a extração de instâncias de tupla do tipo *T* de documentos do tipo *D*, retornados como resposta a consultas realizadas à fonte *F*.

Uma vez construído o modelo de extração M_T^D, o processo de extração pode ser utilizado. Cada vez que executa, ele submete uma consulta *q* e recebe como resposta um documento *d*, que traz em seu corpo, em meio ao conteúdo textual, um conjunto de instâncias de tuplas *ST*. Logo após a configuração de *M*, as extrações são garantidamente bem-sucedidas, pois os documentos resposta d_1, d_2,

... , d_r serão obrigatoriamente do tipo *D*, contendo portanto instâncias de tuplas do tipo *T*, ou seja, exatamente o previsto pelo modelo M_T^D.

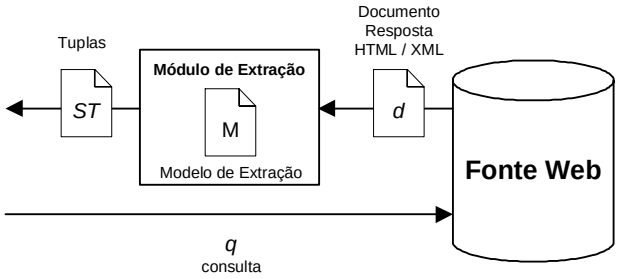


Figura 4.1 – Processo de Extração.

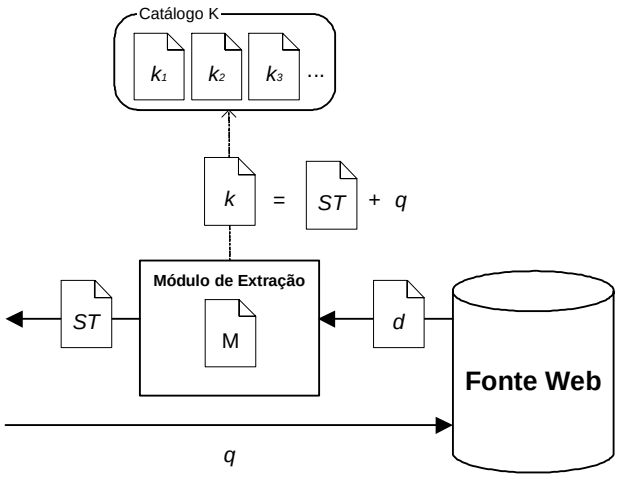


Figura 4.2 – Catalogação da Extração

Caracteristicamente, o processo de extração funcionará durante algum tempo com o modelo de extração M_T^D inalterado. No entanto, é possível que, depois de um período de extrações bem sucedidas, a rotina de **verificação da extração** comece a apontar falha no processo de extração. Como vimos, isso ocorre necessariamente devido a mudanças no tipo dos documentos resposta retornados pela fonte, que podem ou não vir acompanhadas de mudanças no tipo de tupla presente nos documentos.

Para que o processo de extração volte a funcionar com sucesso, é necessário um modelo de extração $M_{T'}^{D'}$ compatível com o novo tipo de documento *D'* e com o possível novo tipo de tupla *T'*. Para compor $M_{T'}^{D'}$, será necessário gerar um novo descritor de extração $M_{T'}^{D}$, e possivelmente a representação abstrata do novo tipo de tupla *T'*. Isto poderia ser feito novamente por intervenção do usuário, de forma manual ou semi-automatizada. Porém, nosso interesse, e principal objetivo, é automatizar esse processo. Neste caso, é de responsabilidade da rotina de **manutenção da extração** promover ajustes no modelo de extração, de forma a adequá-lo às mudanças ocorridas.

Um dos pilares da nossa proposta de uma técnica para a manutenção automatizada da extração é a catalogação de consultas realizadas e os respectivos resultados obtidos, durante os períodos de funcionamento sem falha do processo de extração. Mantém-se um catálogo *K*, da forma

$K = \{ (q_1, ST_1), (q_2, ST_2), \ldots ,(q_k, ST_k) \}$,

onde ST_i é o conjunto de instâncias de tupla obtidas via extração, a partir do documento d_i, obtido como resposta à consulta q_i. A quantidade *k* de pares consulta e resultado a ser mantido no catálogo é parametrizável. A renovação do catálogo respeita o modelo de fila: se o catálogo já possuir *k* pares, um novo par catalogado deve substituir o mais antigo, que será desprezado. Desta forma, garante-se que o catálogo será mantido sempre atualizado.

Consultas e seus respectivos resultados são catalogadas com o intuito de serem utilizadas, em situação de falha do processo de extração, para obtenção de informações que serão usadas para gerar o novo modelo de extração. O uso de pares consulta e resposta catalogados baseia-se no princípio explicado a seguir.

Suponha que a fonte *F* alterou a forma de seus documentos resposta de um modo que fez o processo de extração passar a falhar. Nesta situação, selecionamos um par consulta e resultado (q_1, ST_1), catalogado antes da situação de falha. Submetendo a consulta q_1 novamente à fonte *F*, obtém-se um documento resposta d_1'. Apesar de diferente do documento d_1 obtido originalmente quando o par (q_1, ST_1) foi catalogado, esperamos encontrar em d_1' o mesmo conjunto de instâncias de tupla ST_1. Isto se justifica por (q_1, ST_1) ter

sido catalogado recentemente, o que é garantido pela política de renovação do catálogo *K*.

Porém, o conjunto de instâncias de tuplas ST_1' de fato presente em d_1' pode não ser idêntico a ST_1. Pode ter havido alterações nos dados da fonte *F* entre o momento da catalogação de (q_1, ST_1), e o momento em que q_1 foi submetida novamente à fonte *F* pela rotina de manutenção da extração, apesar da garantia de que a diferença de tempo entre os dois eventos é pequena. Outra causa de diferenças pode ser pequenas alterações no tipo de tupla, como a ausência de um atributo obrigatório, ou repetição de um atributo que deveria ser monovalorado, o que faria as instâncias de ST_1' e ST_1, apesar de similares, diferentes por definição, por serem de tipos de tupla distintos.

De qualquer forma, apesar de não termos garantia de que ST_1' será igual a ST_1, ou seja, de que teremos absolutamente as mesmas instâncias de tupla em ST_1 e ST_1', sabemos que teremos uma grande quantidade de instâncias de atributos pertencentes às tuplas de ST_1 que estarão presentes, codificadas, dentro do documento d_1'.

A partir de cada instância de atributo presente nas instâncias de tuplas que compõem ST_1, podemos tentar localizar a codificação da instância de atributo no documento d_1', por busca da *string* correspondente ao valor da instância de atributo no texto de d_1'. Se for localizada, a instância codificada (valor da instância, juntamente com delimitadores anterior e posterior) é guardada.

Após repetir o processo para outros pares consulta e resultado do catálogo, estaremos de posse de várias instâncias codificadas de atributos, que alimentarão o processo de geração dos novos padrões de extração de atributos (PEAs), que formam o descritor de extração.

4.4.2 Geração Semi-Automatizada do Modelo de Extração Original

Conforme mencionado, o princípio utilizado na geração semi-automatizada do modelo de extração original é o mesmo do processo de manutenção da extração, através da regeração do modelo de extração.

Foi visto que no processo de regeração do modelo de extração utiliza-se de um catálogo *K* composto por pares ordenados (*q*, *ST*) de consultas e seus respectivos resultados de extração. O princípio da geração semi-automatizada do modelo inicial é o mesmo. Na situação inicial, como não se tem ainda um catálogo populado, uma vez que ainda não foram executadas extrações. Assim, para gerar o modelo de extração, o usuário deverá prover um conjunto de pares exemplo de consultas e resultados, simulando, assim um catálogo. Cada exemplo deve ser criado pelo usuário observando o documento resposta de cada consulta, e determinando qual deveria ser o conjunto de tuplas de resposta de um processo de extração bem sucedido sobre o documento. Normalmente, um conjunto de dois ou três pares consulta e resultado será suficiente para gerar o modelo de extração inicial.

4.4.3 Verificação da Extração na Abordagem das Reconsultas

Como vimos, o processo de verificação da extração determina o processo de extração foi bem sucedido ou falhou, baseado em um conjunto de regras *R* que classificam o conjunto de tuplas *ST* obtido pelo processo de extração como aceitável ou não.

Em nossa abordagem, regras de verificação se baseiam em não-conformidades de instâncias de *ST* em relação ao tipo de tupla esperado *T*. Como vimos, quando utilizamos uma estratégia de extração *bottom-up*, instâncias de atributos são extraídas de forma independente, para posteriormente serem agrupadas em tuplas. Pode acontecer que uma instância *t* assim montada não corresponda ao tipo *T*. Isso pode acontecer, por exemplo, devido à falta de um ou mais atributos obrigatórios[17]. Neste caso, dizemos que a tupla *t* é **incompleta** com relação ao tipo *T*. Também pode acontecer de um atributo que deveria ocorrer uma única vez estar presente repetidas vezes. Nesta situação, não sendo a tupla *t* incompleta, dizemos que t é **sobrecompleta** em relação ao tipo *T*. A seguir são definidos os dois casos.

[17] Um tipo de atributo é obrigatório se o valor de minOccurs é maior ou igual a 1

Definição 4.16. Dizemos que a instância de tupla t é incompleta em relação ao tipo T, que representamos $t \subset T$, na seguinte condição:

$$t \subset T \Leftrightarrow \exists \text{ atributo } A_i \text{ componente de } t, \text{ tal que } \|A_i^t\| < A_i^{\text{minOccurs}}$$

onde $\|A_i^t\|$ é a cardinalidade do conjunto de ocorrências de A_i em t, ou seja, o número de ocorrências de A_i em t, e $A_i^{\text{minOccurs}}$ é o número mínimo permitido de ocorrências de A_i em T.

Definição 4.17. Dizemos que a instância de tupla t é sobrecompleta com relação ao tipo T, que representamos $t \supset T$, na seguinte condição:

$$t \supset T \Leftrightarrow \quad \forall \text{ atributo } A_i \text{ componente de } t, \text{ tal que } \|A_i^t\| \geq A_i^{\text{minOccurs}} \quad \text{e}$$
$$\exists \text{ atributo } A_i \text{ componente de } t, \text{ tal que } \|A_i^t\| > A_i^{\text{maxOccurs}}$$

onde $\|A_i^t\|$ é a cardinalidade do conjunto de ocorrências de A_i em t, ou seja, o número de ocorrências de A_i em t, e $A_i^{\text{maxOccurs}}$ é o número máximo permitido de ocorrências de A_i em T.

A ocorrência de instâncias incompletas ou supercompletas com relação ao tipo de tupla T, dentro do conjunto de instâncias extraídas ST, é indício de que o processo de extração está em situação de falha.

Seja SI o subconjunto de ST, formado pelas instâncias de ST que são incompletas em relação ao tipo T, e SS o subconjunto de ST das instâncias sobrecompletas em relação ao tipo T. Caracteristicamente, o conjunto de regras de verificação de um extração incluirá regras da forma

r_1: $\|ST\| \geq p_1$;

r_2: $\|SI\| / \|ST\| \leq p_2$;

r_3: $\|SS\| / \|ST\| \leq p_3$.

As regras são normalmente parametrizáveis. Nos exemplos acima, p_1, p_2 e p_3 são parâmetros da verificação, que configuram o processo de verificação, de modo a torná-lo mais ou menos rígido, na sua tarefa de qualificar uma extração como bem sucedida.

Em resumo, na abordagem proposta, o processo de verificação da extração baseia-se na solução de um problema central:

- **Teste de conformidade de tupla com relação a um tipo de tupla.** Este processo deve responder se uma tupla é instância válida, incompleta ou sobrecompleta em relação ao tipo de tupla em questão.

4.4.4 Manutenção da Extração na Abordagem das Reconsultas

Explicaremos agora a estratégia proposta para a rotina de manutenção da extração, que possibilitará que a mesma seja capaz de atualizar o modelo de extração, quando disparada pela rotina de verificação da extração, ao ser detectada falha no processo de extração.

Ao ser disparada, após falha detectada pelo processo de verificação da extração, a rotina de manutenção da extração utilizará os pares consulta e resultado do catálogo *K*, para obter um conjunto de instâncias de atributos codificadas segundo o novo tipo de documento *D'* retornado pela fonte *F*. Esse conjunto de instâncias codificadas de atributo serão utilizados para gerar os novos padrões de extração de atributo (PEAs) $[A_1^{D'}]^{\text{pattern}}$, que juntos formam o novo descritor de extração $E_T^{D'}$, parte do novo modelo de extração a ser gerado.

Seja (q_r, ST_r) um item do catálogo *K*. As instâncias de tupla *t* de ST_r são compostas pelas instâncias de atributo $a_1, a_2, \ldots, a_m$. Como vimos na definição 2.6, cada instância de atributo a_j é da forma $a_j = (A_i^{\text{label}}, a_j^{\text{value}})$, onde a_j é do tipo A_i. A idéia básica por trás da geração dos PEA é a obtenção da instância codificada a_j^{enc}, que corresponde à codificação de a_j presente em *d'*. A localização de a_j^{enc} é feita por procura pela *string* exata a_j^{value} no texto de *d'*. Na verdade, essa busca pode encontrar diversas ocorrências da *string* a_j^{value} no texto de *d'*, assim como pode encontrar uma só, ou nenhuma. Da forma que for, cada ocorrência encontrada será guardada como uma instância de atributo candidata c_k, segundo a definição 2.11, na forma $c_k = (A_i^{\text{label}}, a_j^{\text{value}}, c_k^{\text{pos}}, c_k^{\text{start}}, c_k^{\text{end}})$. Estas instâncias obtidas serão guardadas em um conjunto *SC* das instâncias candidatas.

O processo se repete para todos os atributos de todas as tuplas de todos os resultados do catálogo *K*, sempre acrescentando instâncias candidatas ao

conjunto *SC*. Ao final deste processo, teremos o conjunto *SC* contendo todas as instâncias candidatas que o processo foi capaz de localizar.

O próximo passo será gerar o padrão de extração de atributo (PEA) para cada tipo de atributo. Para isso, dividimos o conjunto *SC* por tipo de atributo, sendo SC_i o subconjunto de *SC* correspondente ao tipo de atributo A_i. O PEA de A_i em D' será $[A_i^{D'}]^{pattern} = [SC_i]^{start} \,.\, [A_i^{dataType}]^{pattern} \,.\, [SC_i]^{end}$, onde $[SC_i]^{start}$ e $[SC_i]^{start}$ são, respectivamente, a expressão de padrão capaz de reconhecer os delimitadores anteriores e posteriores de A_i em D', geradas a partir de SC_i.

Os algoritmos de geração dos padrões de extração devem ser capazes de lidar com a existência de falsas instâncias obtidas em *SC*, de forma que estas não venham a influenciar as expressões de padrão geradas.

Após gerados todos os PEAs, o descritor de extração estará pronto. Será realizada então uma extração *bottom-up* de análise, utilizando o descritor de extração recém gerado. A partir de análise das instâncias de atributos obtidas, pode ser detectada alteração na estrutura das tuplas, ou seja, alteração no tipo de tupla. Neste caso, uma nova representação de tipo de tupla *T'* será gerada. É possível determinar também se o tipo de tupla apresenta constância na disposição dos atributos, que possibilite a utilização de uma estratégia de extração *top-down*. Nestes casos, gerá gerado um padrão de extração de tupla (PET).

Assim, estará completo o modelo de extração, e o processo de extração poderá voltar a funcionar sem falha.

Na abordagem apresentada, para que a manutenção da extração, ou seja, a reescrita do modelo de extração, tenha sucesso, faz-se necessário desenvolver soluções para os seguintes problemas principais:

- **Escolha das instâncias candidatas de um determinado tipo de atributo que serão utilizadas como base para geração do padrão de extração de atributo (PEA).** Dentre as supostas ocorrências encontradas de um determinado tipo de atributo, algumas serão falsas ocorrências, que não devem influenciar o processo de geração do PEA.
- **Generalização do padrão de extração de atributo (PEA).** A partir das ocorrências reconhecidas como reais, generalizar um PEA capaz de

reconhecer instâncias do tipo de atributo, considerando as variações que podem ocorrer.

4.4.5 Extração de Hierarquias de Entidades

Na visão apresentada neste trabalho, cada processo de extração baseia-se em um determinado modelo de extração M_T^D, que configura a extração do tipo de tupla *T* a partir do tipo de documento *D*. Em primeira análise, pode parecer que a abordagem proposta não é capaz de extrair entidades mais complexas, formadas pela agregação de mais de uma tupla. Veremos que não é o caso.

Para exemplificar, retomemos o exemplo das bibliotecas que desejam divulgar seus acervos de livros, explorado na seção 3.3. Vamos assumir que cada ocorrência de livro traga consigo vários campos de informações sobre o autor, como nome, nacionalidade, data de nascimento e data de falecimento (opcional). Uma possível linha de ação seria configurar a extração de modo a trazer informações de livro e autor juntas nas tuplas geradas. Em outras palavras, haveria um único tipo de tupla, que teria tanto informações de livro quanto informações de autor, de forma desnormalizada. A principal desvantagem deste formato é a repetição desnecessária de informação: as informações de um autor se repetem para cada livro escrito por ele. Note que este mesmo problema existiria em uma representação XML hierarquizada, onde o autor aparecesse como uma sub-entidade de livro. A repetição continuaria ocorrendo, mas com a vantagem da hierarquia que demonstra uma certa independência entre as entidades.

```
<?xml version="1.0"?>
<livros xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.biblioteca-1.org/termos-biblioteca#"
        xml:base="http://www.biblioteca-1.org/livros">
   <Livro rdf:ID="ISBN-8508040814">
      <titulo>Dom Casmurro</titulo>
      <assunto>Literatura Brasileira - Romance</assunto>
      <ano-criacao>1900</ano-criacao>
      <editora>Ática</editora>
      <ano-edicao>1997</ano-edicao>
      <paginas>246</paginas>
      <nome-autor>Machado de Assis</nome-autor>
      <nacionalidade-autor>Brasileiro</nacionalidade-autor>
      <data-nascimento-autor>1839</data-nascimento-autor>
```

```
      <data-falecimento-autor>1908</data-falecimento-autor>
   </Livro>
   ...
</livros>
```

Figura 3.11 – Informações de livro e autor presentes na mesma sintaxe RDF

```
<?xml version="1.0"?>
<livros>
   <Livro>
      <titulo>Dom Casmurro</titulo>
      <assunto>Literatura Brasileira - Romance</assunto>
      <ano-criacao>1900</ano-criacao>
      <editora>Ática</editora>
      <ano-edicao>1997</ano-edicao>
      <paginas>246</paginas>
      <autor>
         <nome>Machado de Assis</nome-autor>
         <nacionalidade>Brasileiro</nacionalidade-autor>
         <data-nascimento>1839</data-nascimento-autor>
         <data-falecimento>1908</data-falecimento-autor>
      </autor>
   </Livro>
   ...
</livros>
```

Figura 3.12 – Possível sintaxe XML para informações de livro e autor

Analisando a abordagem para extração de dados apresentada por este trabalho, pode-se ter a falsa impressão de que só é possível configurar um único processo de extração por tipo de documento. Na verdade, pode-se ter diversos processos de extração configurados sobre a mesma fonte, ou seja, sobre o mesmo tipo de documento. Desta forma, para o problema da extração das informações de livros e autores, podemos ter dois processos de extração distintos: um voltado à extração de tuplas de livros, e outro responsável por extrair tuplas de autores. Surge então a questão: como relacionar as tuplas de livros com as tuplas de seus respectivos autores?

O modelo RDF suporta esse tipo de relacionamento. Como discutido na seção 3.4, RDF suporta a criação de relacionamentos entre recursos, baseados em propriedades e no conceito de identidade de recurso. Para exemplificar, serão

mostrados dois documentos RDF distintos, um contendo informações de livros e outro com informações de autores. Apesar da separação, é possível notar que as informações mantém ligações entre si.

```
<?xml version="1.0"?>
<livros xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns="http://www.biblioteca-1.org/termos-biblioteca#"
        xml:base="http://www.biblioteca-1.org/livros">
   <Livro rdf:ID="ISBN-8508040814">
      <titulo>Dom Casmurro</titulo>
      <assunto>Literatura Brasileira - Romance</assunto>
      <ano-criacao>1900</ano-criacao>
      <editora>Ática</editora>
      <ano-edicao>1997</ano-edicao>
      <paginas>246</paginas>
      <autor rdf:resourse="http://www.biblioteca-1.org/autores#Assis,Machado"/>
   </Livro>
   ...
</livros>
```

Figura 3.13 – Relacionamento entre recursos em RDF

Como pode ser visto, cada recurso livro no primeiro documento traz uma ou várias propriedades “autor”, cada uma tendo como objeto um outro recurso, do tipo autor. Estes recursos do tipo autor, no entanto, não estão presentes localmente com relação aos livros. Ao invés disso, o autor é referenciado através de um identificador, que aponta para o autor, presente em outro documento.

```
<?xml version="1.0"?>
<autores xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://www.biblioteca-1.org/termos-biblioteca#"
         xml:base="http://www.biblioteca-1.org/autores">
   <Autor rdf:ID="Assis,Machado">
      <nome>Machado de Assis</nome-autor>
      <nacionalidade>Brasileiro</nacionalidade-autor>
      <data-nascimento>1839</data-nascimento-autor>
      <data-falecimento>1908</data-falecimento-autor>
   </Autor>
   ...
</autores>
```

Figura 3.14 – Relacionamento entre recursos em RDF

O esquema de identificação ilustrado baseia-se no uso de rdf:resource. Cada autor recebe assim um identificador geral, que é a URI correspondente ao valor de rdf:resource. Obviamente, será necessária uma estratégia para geração de ids únicos, na qual podem ser usadas informações das próprias tuplas, de sistemas externos, geração de números sequenciais, *hash codes*, etc. Estratégias para geração de ids fortes, apropriados para identificação de recursos, no entanto, está além do escopo deste trabalho.

4.4.6 Limitações da Técnica das Reconsultas

Apesar do modelo de extração ser composto por duas partes, a definição do tipo de tupla T e o descritor de extração $E_{T'}^{D}$, a abordagem para manutenção da extração proposta concentra-se principalmente em mudanças que afetam o descritor de extração. O suporte que é dado a possíveis mudanças no tipo de tupla é bastante limitado.

As técnicas de verificação da extração discutidas são capazes de obter informações que podem indicar alterações na composição da tupla, a partir dos tipos de atributos envolvidos. Ocorrências de tuplas incompletas sinalizam uma possível diminuição no número de ocorrências mínimo (minOccurs) para o atributo gerados da desconformidade. No caso extremo, quando não é mais detectada nenhuma ocorrência do tipo de atributo, é possível que o mesmo não componha mais o tipo de tupla. Por outro lado, a ocorrência de grande quantidade de tuplas sobrecompletas pode indicar que um determinado atributo aumentou seu número máximo de ocorrências. No entanto, não é possível, na abordagem apresentada, detectar automaticamente o surgimento de um novo tipo de atributo componente do tipo de tupla. A inclusão do novo atributo terá que ser realizada de forma semi-automatizad, através da geração de exemplos de extração contendo o novo tipo de atributo, e da execução da rotina de geração do modelo de extração a partir dos exemplos.

Limitação similar existe na manutenção de relacionamentos estruturais entre diferentes tipos de tupla oriundos da mesma fonte Web, como descrito na seção

anterior. Se estes relacionamentos sofrerem alteração nas fontes em questão, os mesmos não poderão ser atualizados de forma automática, sendo necessária a intervenção manual do usuário.

4.5 Conclusões do Capítulo

Este capítulo iniciou com a apresentação de uma formalização dos conceitos de extração de dados, para servir como base para a discussão dos pontos seguintes.

Foi discutido em detalhe um importante problema presente em cenários de extração de dados de fontes textuais dinâmicas, que são as eventuais mudanças que podem ocorrer no formato exportado pelas fontes.

Para enfrentar esse problema, o desafio é a construção de processos de extração resilientes, ou seja, capazes de conviver com as mudanças. A primeira necessidade é dotar o processo geral com um mecanismo de verificação da extração, ou seja, com uma forma de decidir se a extração está funcionando adequadamente, ou encontra-se em situação de falha. Uma vez detectada falha, o processo deverá reconfigurar a extração, de modo que esta volte a funcionar corretamente.

Neste capítulo, apresentamos uma abordagem para construção de processos de extração resilientes, dotados de capacidade de verificação e manutenção automática da extração. A técnica baseia-se em dados históricos coletados das fontes, mantendo assim total independência de informações sintáticas de publicação dos dados. Apresentamos então uma formalização para os problemas de verificação e manutenção automática da extração, baseadas na abordagem apresentada. Finalmente, foi esclarecida a forma de extração de hierarquias, assim como as limitações da abordagem.

Capítulo 5

Construção de um Módulo de Extração Resiliente

Como vimos, para que uma solução de extração seja capaz de adaptar-se a possíveis mudanças na fonte consultada, faz-se necessário acoplar à tarefa de extração uma outra funcionalidade, que chamamos manutenção da extração.

Para fazer o processo de extração voltar a funcionar, a funcionalidade de manutenção da extração deverá alterar automaticamente o modelo de extração, de forma que este passe a refletir a nova situação, e possa assim voltar a ser usado com sucesso pelo processo de extração.

Como podemos perceber, o modelo de extração é o elo que liga extração e sua manutenção. O escolha de um modelo de extração ao mesmo tempo funcional e simples de ser regerado, é fator chave para um esquema de extração resiliente. O presente trabalho utiliza um modelo de extração baseado em expressões regulares, como discutido na seção 2 deste capítulo. Antes disso, discutimos a organização da implementação na forma de um framework, o que possibilita a criação de soluções similares, nos mesmos moldes.

5.1 Framework para construção de extratores e mantenedores baseados na técnica das reconsultas

Em desenvolvimento de software, o conceito de se ter uma infra-estrutura de apoio para a construção de sistemas, que promova a reutilização e minimize o retrabalho, não é novo. Estes já eram os objetivos das ferramentas CASE, numa abordagem que previa interação em alto nível com o desenvolvedor, e a conseqüente geração semi-automatizada de código.

Com a disseminação do paradigma da orientação a objetos, a abordagem da geração de código foi sendo substituída por uma abordagem de composição para construção de aplicações baseadas numa infra-estrutura existente. Neste caso, tal infra-estrutura consiste de um conjunto de classes inter-relacionadas, denominado *framework*. As classes de um novo sistema que é criado guardam relação com as classes desta infra-estrutura, através dos mecanismos da orientação a objetos, seja pelo uso direto de classes disponibilizadas, seja pela especialização das mesma através do mecanismo de herança. Esta relação permite o reuso de arquitetura e *design* de aplicação estabelecidos pelas classes do *framework*, além de disciplinar o desenvolvimento de modo a seguir tais diretivas. No momento da distribuição, o código de infra-estrutura é usado conjuntamente com o código específico da aplicação para a geração do sistema.

Na construção da parte implementada deste trabalho, organizamos o desenvolvimento, isolando as funcionalidades de infra-estrutura na forma de um *framework* para construção de extratores e seus respectivos mantenedores, responsáveis pela manutenção da extração.

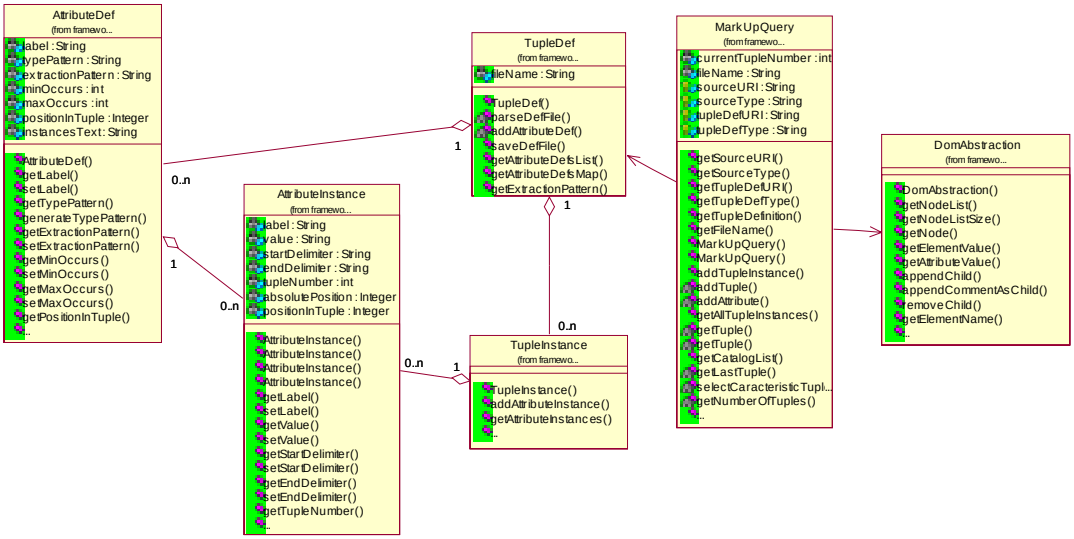


Figura 5.1 – Classes básicas do *framework*

O *framework* traz um conjunto de classes básicas que encapsulam os conceitos essenciais do domínio do problemas. Juntas, essas classes formam uma API que disponibiliza funcionalidades essenciais para resolução dos problemas em um nível superior de abstração, simplificando o desenvolvimento. A figura acima apresenta as classes básicas do *framework* na forma de um diagrama de classes UML.

Dentre as funcionalidades disponibilizadas pelo conjunto de classes básicas, pode-se destacar:

- Abstrações para os conceitos básicos de tipo de tupla (`TupleDef`), instância de tupla (`Tuple`), tipo de atributo (`AttributeDef`) e instância de atributo (`Attribute`), incluido as relações estruturais entre os mesmos. Um tipo de tupla agrega um conjunto de tipos de atributo. Cada tipo de atributo traz consigo o seu padrão de extração. Desta forma, cada tipo de tupla define um modelo de extração.
- Suporte para consulta às fontes, incluindo obtenção do conteúdo dos documentos via HTTP, e associação de uma consulta com um modelo de extração, através de um tipo de tupla.
- Suporte à geração de resultados de extrações em formato RDF.
- Suporte à catalogação de resultados de extrações realizadas em uma fonte;
- Manipulação de alto nível do conteúdo de resultados catalogados.

Além das funcionalidades oferecidas pelas classes básicas, o *framework* estabelece um *design* a ser seguido na construção de um novo caso. A idéia é determinar o comportamento geral de extratores e mantenedores, e deixar sob responsabilidade do desenvolvedor da nova aplicação decisões sobre como implementar certos passos deste comportamento. Isto é alcançado pelo uso de um padrão de design baseado em métodos *Template* [GHJV94], que nos permite seguir sempre a mesma diretriz geral, determinando soluções específicas para as partes que compõem o problema maior. Este *Design Pattern* define o esqueleto de um algoritmo em uma operação, deferindo alguns passos para subclasses. Sua

utilização ajuda a organizar o pensamento e dividir o problema em partes componentes modulares, que podem ser evoluídas de forma independente.

A figura abaixo mostra as hierarquias de classes do *framework* que devem ser estendidas para construção de um novo extrator.

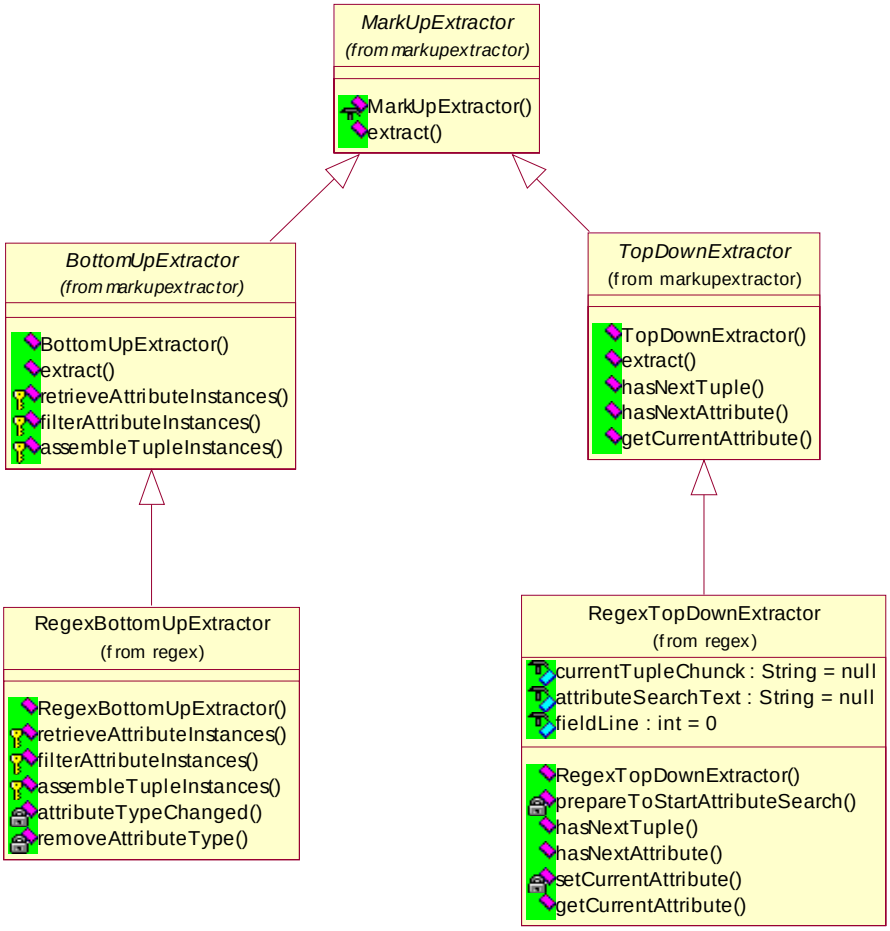


Figura 5.2 – Hierarquia de classes dos extratores

Para construir um novo extrator, o desenvolvedor deverá, inicialmente, decidir se este utilizará estratégia de extração *top-down* ou *bottom-up*. Dependendo desta decisão, ele irá especializar, via herança, a classe `TopDownExtractor` ou `BottomUpExtractor`, respectivamente. Cada uma traz sua implementação para a operação `extract()`, definida em sua superclasse comum

MarkUpExtractor. Estas implementações de `extract()`presentes nas duas classes são escritas cada uma em função de um conjunto de operações abstratas, ou seja, operações para as quais a classe em questão não provê implementação, ficando a cargo das subclasses determinar a lógica de execução destas operações. Desta forma, suas subclasses podem alterar certos passos do processo geral, customizando-o em pontos específicos, não podendo, no entanto, alterar seu fluxo geral de execução. Uma grande vantagem dessa abordagem é que a complexidade desses algoritmos pode crescer de forma independente, permitindo o contínuo refinamento do processo em pontos específicos.

Na implementação trazida por *TopDownExtractor*, o método `extract()` executa de forma iterativa, realizando a extração enquanto existirem tuplas a extrair, e para cada tupla extrai as instâncias de todos os atributos existentes. Desta forma, as subclasses devem manter contexto sobre a tupla corrente, para serem capazes de prover as seguintes operações:

- `hasNextTuple()`: informa se existe próxima tupla, e em caso afirmativo, faz da mesma a tupla corrente;
- `hasNextAttribute()`: informa se existe próximo atributo na tupla corrente, e em caso afirmativo, faz do mesmo o atributo corrente;
- `getCurrentAttribute()`: retorna o atributo corrente;

Já a implementação de `extract()` em *BottomUpExtractor* obtém inicialmente todas as instâncias de atributos presentes no documento, e num segundo passo, realiza seu agrupamento em tuplas. Para tal, uma subclasse precisa implementar os seguintes métodos:

- `retrieveAttributeInstances(AttributeDef)`: para um determinado tipo de atributo, retorna todas as instâncias de atributo encontradas no documento;
- `assembleTupleInstances()`: agrupa as instâncias de atributos encontradas no documento em tuplas;

Para implementar o processo de verificação da extração, o desenvolvedor tem à sua disposição as funcionalidades disponibilizadas pela superclasse *`Verifier`*. Esta classe define o método *template* `verify()`, que comanda o processo de verificação da extração. Na parte inicial de sua execução, este método comanda o levantamento do número de tuplas incompletas e sobrecompletas presentes em um resultado de extração, deixando esta informação disponível. Isto é feito através de um método privado, disparado pela implementação do método *template*. O método `analyzeResult()` recebe o nome do resultado a ser analisado, instancia-o, e em seguida percorre todas as suas tuplas, classificando cada uma como incompleta, sobrecompleta ou em conformidade com o tipo de tupla, registrando ao final a quantidade total de tuplas em cada situação. Após a execução do método, estes valores podem ser obtidos através dos métodos `getTotalTuples()`,`getNumIncomplete()` e `getNumOvercomplete()`. A seguir é mostrada a hierarquia para construção de um verificador.

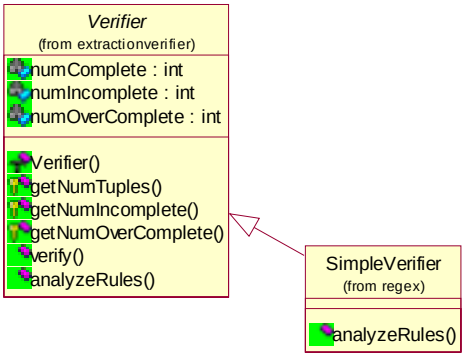


Figura 5.3 – Hierarquia de classes para verificadores

Fica a cargo do desenvolvedor da subclasse específica complementar a execução, através da implementação de um único método:

- `analyzeRules()`: utilizando informações obtidas através dos métodos `getTotalTuples()`, `getNumIncomplete()` e `getNumOvercomplete()`, além de outras que possa obter, este método deve responder se a extração está funcionando corretamente, ou encontra-se em estado de falha;

O diagrama mostrado a seguir traz a hierarquia de classes do *framework* que devem ser estendidas para construção de um novo mantenedor de extração. A árvore hierárquica de um mantenedor se inicia na superclasse `ModelGenerator`, que exige que suas subclasses implementem o método abstrato `generate()`. Esta operação é definida em alto nível nas subclasses diretas, na forma de um método *template*, como acontece em `PatternBasedGenerator`. O método `generate()` comanda o processo de regeração do modelo de extração, usando para isso resultados antigos presentes no catalogado para uma dada consulta. De posse de um desses resultados antigos, o processo tenta localizar, no novo documento resposta obtido para a consulta, os mesmos valores presentes nos atributos das tuplas do resultado catalogado, que devem corresponder a instâncias de atributos presentes no novo documento. A partir dessas instâncias, o novo modelo de extração é gerado.

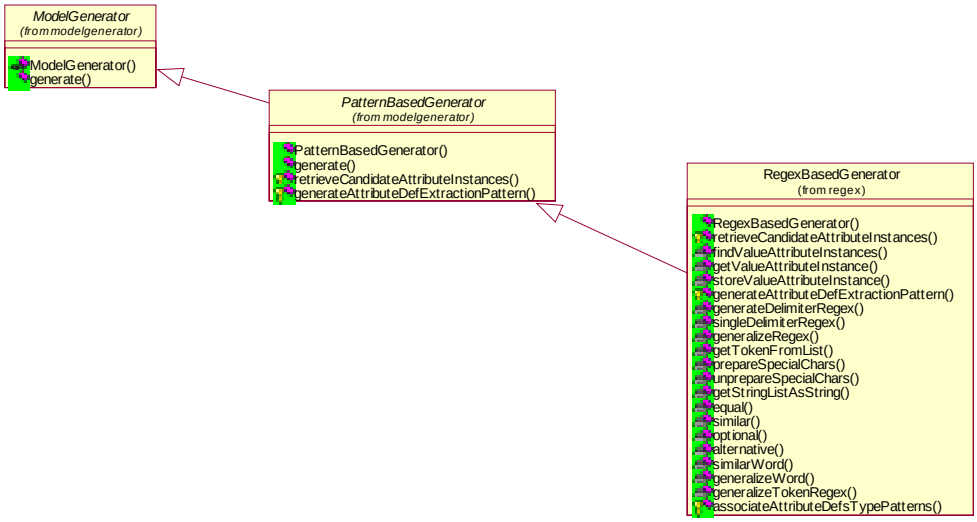


Figura 5.4 – Hierarquia de classes dos mantenedores

Para que o processo funcione, subclasses de `PatternBasedGenerator` devem implementar os seguintes métodos:

- `retrieveCandidateAttributeInstances()`: encontra prováveis instâncias de atributo a partir dos valores dos atributos de uma tupla do catálogo;
- `generateAttributeDefExtractionPattern(AttributeDef)`: gera padrão de extração para um tipo de atributo a partir das instâncias candidatas localizadas;

Além do *framework* em si, o trabalho traz uma implementação específica de um conjunto extrator e mantenedor. O extrator é subclasse de *`BottomUpExtractor`*, enquanto que o mantenedor é uma especialização de *`PatternBasedGenerator`*. O ponto de ligação entre os dois, o modelo de extração, utiliza expressões regulares para as expressões de padrão que formam os padrões de extração de atributos (PEAs).

5.2 Modelo de extração baseado em Expressões Regulares

5.2.1 Usando expressões regulares

O problema típico de reconhecimento de padrões em *strings* envolve uma *string* de texto e uma outra *string* representando um padrão. A tarefa consiste em localizar uma ou várias *substrings* da *string* de texto que se enquadrem no padrão dado. Este padrão é, tipicamente, uma expressão regular.

Uma expressão regular é constituída de caracteres e operadores, além de caracteres especiais, que funcionam como uma abreviação, substituindo uma combinação comum de caracteres e operadores. Os símbolos usados para os operadores e caracteres especiais podem mudar de uma implementação para outra. Desta forma, dependendo do contexto ou linguagem de programação com suporte à expressões regulares que se está usando – grep, sed, awk, Perl ou mesmo Java – a sintaxe usada para escrever expressões regulares pode mudar. No entanto, a similaridade é grande, e a funcionalidade idêntica.

Para ilustrar, vejamos alguns exemplos de expressões regulares na sintaxe que utilizamos nas implementações, determinada pelo suporte à expressões regulares dado pelo pacote java.util.regex [JAVA02], disponível a partir do JSDK 1.4:

Expressão Regular	Descrição
`casa`	reconhece a string “casa”
`ca.a`	reconhece “casa”, “cama”, “capa”, “ca7a”, etc
`ca[drt]a`	reconhece “cada”, “cara” e “cata”
`ca[^drt]a`	reconhece “casa”, “cama”, “capa”, etc, exceto “cada”, “cara” e “cata”
`a[j-n]a`	reconhece “aja”, “aka”, “ala”, “ama” e “ana”
`asa\d`	reconhece “asa1”, “asa2”, “asa3”, etc
`car+o`	reconhece “caro”, “carro”, “carrro”, “carrrro”, etc
`carros?`	reconhece “carro” ou “carros”
`toc(toc)*`	“toc”, “toctoc”, “toctoctoc”, “toctoctoctoc”, etc
`(\d\d)\1`	reconhece “1212”, “1313”, “2121”, “8989”, etc
`^Portanto`	reconhece a palavra “Portanto” no início de uma linha
`etc\.$`	reconhece a string “etc.” no fim de uma linha

Tabela 5.1 – Exemplos de Expressões Regulares

Uma visão mais detalhada sobre expressões regulares e processos computacionais que as utilizam, em particular o processo de busca por *strings* dentro de um texto com base em um padrão, é apresentada no Apêndice A.

5.2.2 Modelo e Descritor de Extração

Pela definição 4.13, um modelo de extração é dado por $M_T^D = [\ T\ ,\ E_T^D\ ]$, onde T representa o tipo de tupla e E_T^D o descritor de extração de T em documentos do tipo D. Desta forma, o descritor de extração deve trazer tanto informações sobre a estrutura dos objetos a serem extraídos, assim como informações que permitam a extração dos mesmos. No caso de processos de extração baseados em expressões de padrão, o descritor de extração é o conjunto dos padrões de extração dos atributos (PEAs). No nosso caso, os PEAs serão expressos na forma de expressões regulares. O principal fator motivador para a adoção de expressões regulares no descritor de extração foi a possibilidade de

reescrita automatizada destas expressões, uma vez que a reescrita do modelo de extração é o foco central do mantenedor da extração e, por conseqüência, deste trabalho.

Neste trabalho, as definições necessárias para o modelo de extração, visando tanto o aspecto estrutural quanto o da definição da extração, são expressas na forma de um RDF *Schema*. A figura abaixo traz um exemplo de modelo de extração, que será referenciado nas explicações a seguir.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                xmlns:rex="http://www.rex.org/rex-ns#"
                xml:base="http://www.rex.org/ear/citacoes">

    . . .

    <rdfs:Class rdf:ID="Citacao"/>

    <rdf:Property rdf:ID="NomeEditora">
        <rdfs:domain rdf:resource="#Citacao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        <rex:extractionPattern>
            <![CDATA[ <pubnm>{WORDS}</pubnm> ]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
    </rdf:Property>

    <rdf:Property rdf:ID="LocalEditora">
        <rdfs:domain rdf:resource="#Citacao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        < rex:extractionPattern>
            <![CDATA[<publoc>{MULTIPLE}</publoc> ]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
    </rdf:Property>

   <rdf:Property rdf:ID="TituloArtigo">
        <rdfs:domain rdf:resource="#Citacao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        <rex:extractionPattern>
            <![CDATA[ <ti>{MULTIPLE}</ti> ]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
```

```
    </rdf:Property>

    . . .

</rdf:RDF>
```

Figura 5.5 – Arquivo de definição do Modelo de Extração, em RDF *Schema*

Nesta representação em RDF *Schema*, um tipo de tupla é representado na forma de uma classe RDF, e cada tipo de atributo que a compõe é representado por uma propriedade que tem como domínio a respectiva classe.

Além desta definição da estrutura da informação RDF a ser extraída, o modelo de extração traz outras informações. São incluídos dados necessários ao processo de extração, através de um vocabulário próprio, definido em um *namespace* independente, referenciado pelo prefixo `rex:`. Para cada atributo, é informado o respectivo padrão de extração, assim como sua cardinalidade máxima e mínima.

Pode parecer estranho colocar informações sobre cardinalidade das propriedades em um RDF *Schema*, já que este não suporta nativamente tal informação, enquanto outros padrões, como XML *Schema*, as suporta. No entanto, o problema de relacionar informações em RDF *Schemas* e XML *Schemas* já foi discutido no capítulo 3. Devido a isso, e por simplificação do desenvolvimento, optou-se por agregar as informações de cardinalidade, assim como os PEAs, diretamente ao RDF *Schema*.

A partir de todas estas informações, o processo de extração carrega sua definição de tupla, composta por suas definições de atributos, que irão guiar o processo de extração em tempo de execução.

5.3 Extração

Dependendo da estratégia de extração que será adotada para a fonte em questão, os PEAs podem ser usados de forma ligeiramente distinta.

A estratégia de extração a ser adotada para uma fonte específica é determinada através de informações colhidas durante o processo de manutenção da extração, como será discutido na próxima seção. Em resumo, se as tuplas apresentarem um

alto grau de constância, com atributos sempre na mesma ordem e sempre presentes, então será possível utilizar uma estratégia *top-down*, mais simples. Neste caso, os PEAs serão agrupados em um padrão de extração de tupla, ou PET.

5.3.1 Extração *Top-Down*

Apesar deste trabalho destinar maior atenção para a estratégia de extração *bottom-up*, mais genérica e poderosa, um processo de extração *top-down* também é suportado pela implementação.

Se uma determinada fonte apresenta tuplas com alto grau de constância, com atributos sempre na mesma ordem e sempre presentes, então será possível utilizar uma estratégia *top-down*, mais simples. Neste caso, os PEAs serão agrupados em um padrão de extração de tupla, ou PET.

O PET também será, por sua vez, uma expressão regular, formada pela junção dos PEAs, sendo que entre dois PEAs vizinhos, são incluídos caracteres curinga para aceitar qualquer texto entre atributos.

Sendo possível utilizar uma estratégia *top-down*, utilizaremos então a classe `RegexTopDownExtractor`, nossa especialização de *`TopDownExtractor`* que implementa os métodos da seguinte forma:

Método: `hasNextTuple()`

Objetivo: informa se existe próxima tupla, e em caso afirmativo, faz da mesma a tupla corrente

Implementação:

Utilizando o PET, que é uma expressão regular, faz-se uma busca na parte do texto do documento ainda não pesquisada, tentando encontrar o próximo fragmento de texto que corresponde a uma tupla. Se não for encontrado mais nenhum, retorna falso. Caso contrário, o retorno é verdadeiro, e o fragmento encontrado é guardado internamente como fragmento de tupla corrente. O primeiro tipo de atributo que forma as tuplas é setado como tipo de atributo corrente.

Método: `hasNextAttributeInstance()`

Objetivo: informa se existe próxima instância de atributo na tupla corrente, e em caso afirmativo, faz da mesma a instância de atributo corrente, que será retornada pela próxima chamada de `getCurrentAttributeInstance()`. Se necessário, o tipo de atributo corrente será setado para o próximo tipo de atributo na seqüência da estrutura da tupla.

Implementação:

Utilizando o PEA do tipo de atributo corrente, que é uma expressão regular, faz-se uma busca no fragmento de texto da tupla corrente, tentando encontrar o próximo fragmento de texto que corresponde a uma instância do atributo. Se não for encontrado mais nenhum, seta o próximo tipo de atributo como corrente, e busca-se uma instância deste. Se não houver mais instâncias nem tipos de atributo, retorna falso. Caso contrário, o retorno é verdadeiro, e o fragmento encontrado é setado internamente como fragmento correspondente à instância de atributo corrente.

Método: `getCurrentAttributeInstance()`

Objetivo: obter a instância de atributo corrente.

Implementação:

Retorna a instância de atributo corrente.

5.3.2 Extração *Bottom-Up*

Quando as tuplas se mostram bastante inconstantes no seu formato, com atributos ausentes ou apresentando ordens diversas, não é possível construir um PET que represente todas as variações de forma que uma tupla pode assumir, não sendo possível aplicar uma estratégia de extração *top-down*. Neste caso mais geral, será utilizada uma estratégia *bottom-up*, que trabalhará com os PEAs individualmente. Usaremos nestes casos nossa classe `RegexBottomUpExtractor`, que herda de *`BottomUpExtractor`*, implementando os métodos da seguinte forma:

Método: `retrieveAttributeInstances(AttributeDef ad)`

Objetivo: para um determinado atributo, retorna todas as instâncias daquele atributo encontradas no documento

Implementação:

Utilizando o PEA do atributo como expressão regular, localiza todas as instâncias de um dado atributo no documento inteiro, não importando a tupla.

Método: `assembleTupleInstances()`

Objetivo: agrupa todas as instâncias de atributos encontradas no documento em tuplas

Implementação:

Ordena-se as instâncias pelas posições absolutas dentro do documento. Percorre-se o conjunto ordenado, associando instâncias à tupla atual, até decidir-se que já se iniciou uma nova tupla, e então passa-se a associar instâncias a esta nova tupla. O algoritmo abaixo mostra nossa estratégia para a tarefa de associação.

Algoritmo: Monta Instâncias de Tupla

Entrada: Conjunto *SA* de instâncias de atributo candidatas a_1, a_2, ... , a_n , encontradas em um mesmo documento *D*.

Saída: Conjunto *ST* de instâncias de tupla t_1, t_2, ... , t_m , montadas a partir das instâncias de atributo a_1, a_2, ... , a_n .

```
    procedimento MONTA_TUPLAS(SA):
1.  ordena SA pela posição onde cada instância foi encontrada
2.  ST ← ∅
3.  L ← ∅
4.  t ← nova instância de tupla
5.  θanterior ← null
6.  enquanto SA ≠ ∅ faça
        inicio
7.          seja a o primeiro elemento de SA
8.          seja θa o tipo de atributo de a
9.          se θa ≠ θanterior
```

```
                inicio
10.                 adiciona θanterior a L
                fim
11.         se A ∈ L então
                inicio
12.                 adicione t a ST
13.                 t ← nova instância de tupla
14.                 L ← ∅
                fim
15.         adiciona a como atributo componente de t
16.         θanterior = θa
17.         SA ← SA – a
        fim
18. retorna ST
```

Figura 5.6 - Algoritmo Monta Instâncias de Tupla

5.4 Verificação da Extração

Conforme mencionado anteriormente, o desenvolvedor de um verificador baseado no *framework* proposto precisa implementar um único método, que complementa a execução do método `verify()`, fazendo uso das informações coletadas pelo método `analyzeResult()`.

Método: `analyzeRules()`

Objetivo: informa se a extração encontra-se em funcionamento aceitável ou em situação de falha, baseado nas regras de verificação.

Implementação:

Na implementação realizada, a verificação da extração é baseada em três parâmetros de configuração:

- MIN_NUM_TUPLES: indica o número mínimo de tuplas que deve estar presente em um resultado. Se o valor obtido através de `getTotalTuples()`, for menor do que este parâmetro, o processo de extração será considerado falho;
- MAX_INCOMPLETE_TUPLES_RATIO: indica a maior razão aceitável entre tuplas incompletas e o total de tuplas do resultado. Se a razão entre

os valores obtidos de `getNumIncomplete()` e `getTotalTuples()` for maior que o valor deste parâmetro, a extração será considerada falha;

- MAX_OVERCOMPLETE_TUPLES_RATIO: indica o valor máximo aceitável para a razão entre tuplas sobrecompletas e o total de tuplas do resultado. Se a razão entre os valores obtidos de `getNumOvercomplete()` e `getTotalTuples()` extrapolar o valor estipulado do parâmetro, considera-se o processo de extração em estado de falha.

Se nenhuma das situações de falha for observada, o processo de extração é considerado em estado operacional.

5.5 Manutenção da Extração

Será discutido agora como cada uma das tarefas previstas para um mantenedor compatível com o padrão proposto pelo *framework* é realizada na implementação de referência.

5.5.1 Obtenção de instâncias de atributo candidatas

O primeiro passo é, a partir dos pares consulta vs. resultado do catálogo, obter instâncias de atributo candidatas, localizadas nos novos documentos retornados pelas consultas. Para cada par consulta vs. resultado do catálogo, obtém-se o documento correspondente no novo formato, resubmetendo-se a consulta. Em seguida, para cada tupla do resultado, obtém-se instâncias de atributos candidatas a partir do texto do documento, através do método descrito abaixo.

Método: `retrieveCandidateAttributeInstances (TupleInstance t)`

Objetivo: localizar instâncias de atributos candidatas presentes no documento, a partir dos valores dos atributos de uma dada instância de tupla do catálogo. À medida que vão sendo encontradas, instâncias de atributos candidatas vão sendo armazenadas, separadas por tipo de atributo.

Implementação:

Para cada atributo da tupla, localiza-se todas as ocorrências do valor exato do atributo no novo documento, e registra-se a posição absoluta (dentro do

documento) e seus delimitadores. O algoritmo que, dado os valores dos atributos de uma tupla, gera um conjunto de instâncias de atributo candidatas é descrito a seguir.

Algoritmo: Obtém Instâncias de Atributos Candidatas

Entrada: Instância de tupla *t*.

Saída: Instâncias de atributo candidatas são geradas e armazenadas, separadas por tipo de atributo.

```
    procedimento OBTEM_INTANCIAS_ATRIBUTO_CANDIDATAS(t):
1.  para cada a_j ∈ t faça
    inicio
2.      seja A_i o tipo de atributo de a_j
3.      para cada ocorrência de a_j^value no texto de d faça
        inicio
4.          se A_i^label . a_j^value ∉ SV então
            inicio
5.              cria nova instância de atributo candidata c
6.              c^label = a_j^label
7.              c^value = a_j^value
8.              c^pos = posição de a_j^value no texto de d
9.              atribui delimitador anterior c^start
10.             atribui delimitador posterior c^end
11.             adiciona c a SC_i
12.             adiciona A_i^label . a_j^value a SV
            fim
        fim
    fim
```

Figura 5.7 – Algoritmo Obtém Instâncias de Atributo Candidatas

Ao final da execução, cada SC_i trará o conjunto das instâncias candidatas de A_i em *d*. O controle feito através do conjunto *SV* evita que um valor utilizado na busca por instâncias de atributo candidatas a partir dos valores de atributos da tupla t_r seja usado novamente na busca a partir dos valores em uma tupla posterior t_s.

Para gerar os padrões de extração de atributos, as instâncias de atributos candidatas obtidas para um determinado tipo de atributo serão utilizadas como subsídio. Seguindo a estrutura proposta pelas classes do *framework*, o método

`generateAttributeDefExtractionPattern(AttributeDef)`, que deve ser implementado pelas subclasses de *`PatternBasedGenerator`*, é o responsável pela geração dos padrões de extração para os tipos de atributo. A implementação deste método utiliza outros métodos, responsáveis por partes modulares que, juntas, formam a solução do problema. Estes subproblemas, assim como os métodos que os atacam, são abordados nas sessões seguintes.

5.5.2 Seleção de instâncias de atributo candidatas

Para geração dos padrões de extração dos tipos de atributos, é necessário um passo intermediário. Para cada tipo de atributo, é necessário fazer um levantamento das possibilidades encontradas para delimitadores anteriores e posteriores, e classificar estas possibilidades por nível de importância. Isto é feito pelo método detalhado a seguir.

Método: `distinctDelimiters (List candidateAttributeInstances)`

Objetivo: a partir da lista de instâncias de atributo candidatas de um determinado tipo de atributo, gerar duas listas distintas: uma contendo os possíveis delimitadores anteriores, e outra com os possíveis delimitadores posteriores encontrados, ambas ordenadas pelo número de ocorrências do possível delimitador.

Implementação:

Para cada elemento da lista de instâncias de atributo candidatas, obtém-se o delimitador anterior. Se este constar da lista de delimitadores distintos do tipo de atributo, incrementar o número de ocorrências deste delimitador distinto. Caso o delimitador em questão não conste ainda, adicioná-lo à lista dos delimitadores distintos. Ao final do processo, a lista deve ser ordenada pelo número de ocorrências de cada delimitador distinto. O mesmo processo deve ser feito para os delimitadores posteriores. O algoritmo detalhado é mostrado a seguir.

Algoritmo: Obtém Delimitadores Distintos

Entrada: Lista de instâncias candidatas SC_i do tipo de atributo A_i.

Saída: São geradas e armazenadas duas listas ordenadas por número de ocorrências: uma com os delimitadores anteriores distintos de A_i, e outra com os posteriores.

```
    procedimento OBTEM_DELIMITADORES_DISTINTOS(SA_i):
 1. para cada c_j ∈ SA_i faça
    inicio
 2.     se c_j^start ∈ SS_i então
        inicio
 3.         incrementa(c_j^start , SS_i)
        fim
 4.     senão
        inicio
 5.         inclui(c_j^start , SS_i)
        fim
 6.     se c_j^end ∈ SE_i então
        inicio
 7.         incrementa(c_j^end , SE_i)
        fim
 8.     senão
        inicio
 9.         inclui(c_j^end , SE_i)
        fim
    fim
 10. ordena(SS_i)
 11. ordena(SE_i)
```

Figura 5.8 – Algoritmo Obtém Delimitadores Distintos

Como resultado da execução, teremos que SS_i e SE_i trarão, respectivamente, o conjunto dos delimitadores distintos anteriores e posteriores encontrados no conjunto de instâncias candidatas, ambos ordenados por número de ocorrências. A informação presente em SS_i e SE_i – quais os possíveis delimitadores e qual a freqüência de cada um – será utilizada no processo de geração do padrão de extração do tipo de atributo. Neste processo, os delimitadores mais freqüentes terão importância superior aos delimitadores menos freqüentes, sendo que os últimos tenderão a ser desprezados. Nesta abordagem, o padrão de extração de atributo de A_i é redefinido em função de SS_i e SE_i , sendo escrita como $[A_i^{D'}]^{\text{pattern}} = [SS_i]^{\text{pattern}} \,.\, [A_i^{\text{dataType}}]^{\text{pattern}} \,.\, [SE_i]^{\text{pattern}}$, onde $[SS_i]^{\text{pattern}}$ e $[SE_i]^{\text{pattern}}$ são, respectivamente, a expressão de padrão capaz de reconhecer os delimitadores mais freqüentes dos conjuntos SS_i e SE_i.

5.5.3 Geração de padrões de extração de atributos

O objetivo deste módulo é gerar uma única expressão regular capaz de reconhecer todas as instâncias encontradas de um determinado atributo. Esta expressão regular será formada de três partes, sendo cada uma delas uma expressão regular, e o padrão de extração do atributo gerado no final será a expressão regular formada pela concatenação destas três.

A primeira parte deve ser uma expressão regular capaz de reconhecer qualquer um dos delimitadores anteriores componentes das instâncias encontradas. Esta deverá ser gerada a partir deste conjunto de delimitadores através do algoritmo abaixo, inspirado na técnica ACME (*Align, Collapse under Mismatch, and Extract*) relatada em [CMM01]. A terceira e última parte do padrão de extração deve ser uma expressão regular gerada da mesma forma a partir do conjunto de delimitadores posteriores das instâncias.

Método: `generateDelimiterRegex(SortedSet orderedDistinctDelimiters)`

Objetivo: gerar uma expressão regular geral que reconheça ocorrências de um determinado delimitador, a partir de diferentes versões desse delimitador, apresentadas em ordem de freqüência de ocorrência.

Implementação:

O algoritmo descrito a seguir é a base desse processo, sendo capaz de generalizar uma expressão regular para reconhecer um novo delimitador.

A execução do algoritmo se dará em dois níveis. No primeiro, os *tokens* das coleções corresponderão a *tags* (ou generalizações de tags na forma de trechos na forma de expressões regulares) da expressão original e do texto do novo delimitador. Já no segundo nível, *tokens* corresponderão a palavras (ou generalizações de palavras como expressões regulares).

Algoritmo: Generaliza Expressão Regular

Entrada: dois conjuntos ordenados de *tokens* de texto, o primeiro contendo os fragmentos de texto que compõem a expressão original, e o segundo contendo os

fragmentos que compõem a nova *string* que provocará a generalização da expressão original.

Saída: nova expressão regular, que será uma generalização da expressão regular original, de modo a reconhecer a nova *string*.

```
    procedimento GENERALIZA_EXPRESSAO_REGULAR(SO, SN):
1.  po = primeiro token de SO
2.  pn = primeiro token de SN
3.  enquanto SO ≠ ∅ e SN ≠ ∅ faça
    inicio
4.      se equivalente(po, pn) então
        inicio
5.          SG = SG ∪ { po }
6.          SO = SO - { po }
7.          SN = SN - { pn }
8.          po = próximo token de SO
9.          pn = próximo token de SN
        fim
10.     senão se similar(po, pn) então
        inicio
11.         SPO = conjunto de palavras de po
12.         SPN = conjunto de palavras de pn
13.         SG = SG ∪ { GENERALIZA_EXPRESSAO_REGULAR(SPO, SPN) }
14.         SO = SO - { po }
15.         SN = SN - { pn }
16.         po = próximo token de SO
17.         pn = próximo token de SN
        fim
18.     senão
        inicio
19.         px = posterior(po)
20.         py = posterior(pn)
21.         se similar(po, py) então
            inicio
22.             SG = SG ∪ { opcional(pn) }
23.             po = px
            fim
24.         senão se similar(pn, px) então
            inicio
25.             SG = SG ∪ { opcional(po) }
26.             pn = py
            fim
27.         senão se similar(px, py) então
            inicio
28.             SG = SG ∪ { alternative(po, pn) }
29.             po = px
30.             pn = py
            fim
31.         senão
            inicio
32.             // não é possível fazer a generalização
```

```
33.                 retorne nulo
                fim
            fim
        fim
34. retorne SG na forma de string
```

Figura 5.9 – Algoritmo Generaliza Expressão Regular

O algoritmo se baseia em métodos de alto nível envolvendo *tokens*, que são explicados a seguir.

O método `equivalente(token1, token2)` compara *tokens*, retornando verdadeiro se os dois *tokens* forem considerados equivalentes, e falso caso contrário. Neste contexto, `token1` e `token2` serão considerados equivalentes se forem *strings* idênticas, ou se `token1` for uma expressão regular e `token2` estiver contido na linguagem definida por `token1`.

Analogamente, para serem considerados similares pela método `similar(token1, token2)`, os dois *tokens* devem ser iniciados ou finalizados pela mesma palavra, ou, no caso de serem compostos de uma única palavra, iniciar ou terminar pela mesma seqüência de dois caracteres.

O método `opcional(token1)` retorna uma expressão regular baseada em `token1`, que acrescenta a *string* vazia à linguagem reconhecida por `token1`. No caso de `token1` não ser uma expressão regular no sentido estrito (isto é, `token1` é a expressão regular que reconhece somente `token1`), a presença de `token1` passa a ser opcional.

Finalmente, o método `alternative(token1,token2)` retorna uma expressão regular baseada em `token1` e `token2` que reconhece a união das linguagens reconhecidas por `token1` e `token2` isoladamente. Desta forma, `token1` e `token2` passam a representar alternativas para o que pode ser aceito pela expressão regular resultante.

A discussão até o momento envolveu a geração de expressões capazes de reconhecer os possíveis delimitadores das instâncias de atributos. Será abordada agora a obtenção da parte média da expressão, correspondente aos valores propriamente ditos das instâncias de atributos.

A parte central do padrão de extração será uma expressão regular que não será gerada, e sim escolhida a partir de um conjunto fechado de possibilidades, a partir da classificação do conjunto de valores das instâncias encontradas como sendo todos representantes de uma determinada categoria de *strings*. A idéia da classificação em categorias foi inspirada em [LM00], assim como a organização das categorias em uma hierarquia.

No entanto, as semelhanças se resumem somente à idéia de classificar os valores, uma vez que as abordagens são profundamente distintas. Em [LM00], a classificação está relacionada ao aprendizado de protótipos [DK97]. No nosso caso, cada categoria corresponde a uma expressão regular capaz de reconhecer *strings* pertencentes ao conjunto representado. O processo classificatório se resume ao percorrimento da árvore correspondente à hierarquia de categoria, no sentido das categorias mais genéricas para as mais específicas, até encontrarmos a mais específica categoria cuja expressão regular reconhece o valor a classificar.

A figura abaixo traz a hierarquia de categoria que utilizamos, juntamente com as expressões regulares correspondentes:

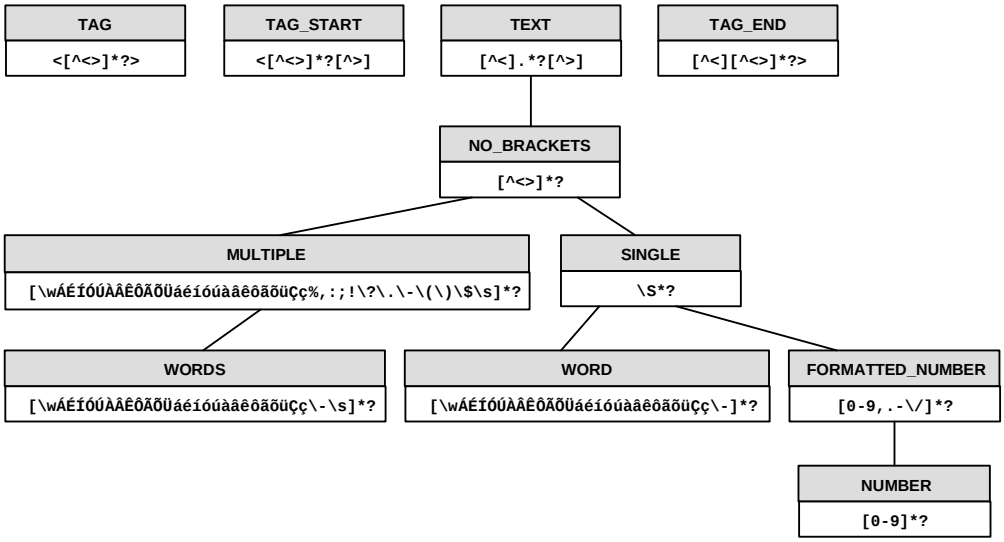


Figura 5.10 – Hierarquia de categorias para o conteúdo textual de atributos

5.6 Análise de Desempenho

5.6.1 Metodologia para análise de eficiência

Como dito, o objetivo principal deste trabalho é o desenvolvimento de uma técnica, baseada na abordagem das reconsultas, que permita que o processo de extração de dados de fontes textuais seja adaptativo. Isto é feito através do processo de manutenção da extração, que permite que o processo de extração se recupere de uma situação de falha, através da reescrita do modelo de extração.

No contexto da análise de desempenho da técnica apresentada, deve ser analisada a eficácia do processo de manutenção da extração, ou seja, em que fração dos casos propostos o modelo de extração é reescrito de forma a retornar o processo de extração para uma situação totalmente operacional. A eficiência do processo de manutenção da extração é determinada em função da eficiência do processo de extração após a recuperação do mesmo, realizada pelo processo de manutenção. Desta forma, faz-se necessário primeiramente determinar como será medida a eficiência do processo de extração.

Para tal, utilizaremos medidas de eficiência normalmente usadas em sistemas de recuperação de informação: precisão e *recall*. Precisão expressa a fração dos objetos relevantes, ou seja, que deveriam ter sido extraídos em relação aos que realmente o foram. *Recall* é a fração de objetos relevantes dentre os objetos que foram extraídos.

Na concepção utilizada neste trabalho, os objetos alvo do processo de extração correspondem a tuplas do modelo abstrato de dados definido. No entanto, para medir a eficiência do processo de extração, é necessário levar em conta a possibilidade de tuplas serem extraídas parcialmente, ou seja, com atributos faltando. Se o processo de extração segue uma estratégia *top-down*, atributos são extraídos de forma independente, existe tanto a possibilidade de um atributo não ser extraído, como do mesmo não ser corretamente associado à sua respectiva tupla. Em ambos os casos, o atributo será considerado não extraído. Fica claro que métricas de eficiência da extração devem ser baseadas nos atributos envolvidos, e não nas tuplas como um todo.

A tabela a seguir mostra quatro classificações para atributos envolvidos no processo de extração, considerando se os mesmos foram extraídos ou não, e se eram ou não relevantes, isto é, se deveriam ou não ter sido extraídos.

	Relevante	Irrelevante
Extraído	encontrado	alarme falso
Não Extraído	esquecido	rejeitado

Tabela 5.2 – Classificações para objetos envolvidos no processo de extração

Com base nas quantidades de atributos em cada uma das categorias, as medidas de eficiência do processo de extração podem ser assim calculadas:

$$\textit{precisão} = \frac{[\text{encontrado}]}{[\text{encontrado}] + [\text{alarme falso}]}$$

$$\textit{recall} = \frac{[\text{encontrado}]}{[\text{encontrado}] + [\text{esquecido}]}$$

Precisão e *recall* são medidas relacionadas que representam diferentes aspectos do conjunto de objetos extraídos. Em muitos casos é possível regular o processo de extração de forma a maximizar uma das medidas em detrimento da outra. Desta forma, é interessante ter uma medida única que englobe os dois aspectos. Para situações onde os valores podem ser inversamente proporcionais, a média harmônica é a composição recomendada. Seguindo a fórmula para o caso de dois valores, a média harmônica *F* entre precisão e recall é calculada como se segue:

$$F = \frac{2 \times \textit{precisão} \times \textit{recall}}{\textit{precisão} + \textit{recall}}$$

A eficiência da manutenção da extração será calculada em função da eficiência do processo de extração, após a manutenção. Neste cenário, é particularmente interessante ter uma medida única para a eficiência da extração.

O princípio para o cálculo da eficiência da manutenção é baseado na determinação de um limite mínimo para o valor da média harmônica entre

precisão e *recall* para as extrações pós manutenção. A eficiência da manutenção da extração $E_{manutenção}$ será dada pela razão entre as extrações pós manutenção acima do limite mínimo e o número total de extrações pós manutenção, como dado abaixo:

$$E_{manutenção} = \frac{\text{[pós extrações acima do limite mínimo]}}{\text{[pós extrações]}}$$

5.6.2 Cálculo da eficiência do processo de manutenção proposto

Para realizar o cálculo da eficiência do processo de manutenção da extração implementado neste trabalho, escolhemos 5 fontes de dados textuais distintas. Para cada uma, configura-se manualmente o processo de extração, através da criação manual do modelo de extração, ou da geração do mesmo a partir de documentos resposta criados manualmente. Uma vez configurado o processo de extração, o mesmo é executado até que se tenha 5 consultas distintas no catálogo. Então simula-se uma situação de falha no processo de extração: o modelo de extração é apagado, e tem que ser gerado a partir dos itens do catálogo. Uma vez regerado o modelo de extração, são realizadas 5 novas consultas, que podem ser distintas das consultas correspondentes aos itens do catálogo, e são calculados os valores de precisão e *recall* para cada consulta, observando as tuplas presentes nos documentos origem e nos respectivos resultados. Em seguida, calcula-se a média harmônica entre precisão e *recall* para cada consulta. Feito isso, é possível calcular a eficiência do processo de extração para cada fonte, observando a fração de consultas àquela fonte com média harmônica superior ao limite mínimo estipulado.

Abaixo seguem os resultados obtidos para as 5 fontes selecionadas e suas 5 consultas. Neste levantamento, foi estipulado um limite mínimo de 95% para a média harmônica.

Fonte: Citações American Economic Review (SGML)			
Url: http://www.aeaweb.org/aer/contents/aer_sgml_index.html			
Consulta	**% *P***	**% *R***	**% *F***

aer9301.sgml	100	93,3	96,5
aer9204.sgml	100	93,6	96,7
aer9203.sgml	100	93,3	96,5
aer9202.sgml	100	98,2	99,1
aer9201.sgml	100	99,1	99,5
***E* (*F* acima do limite mínimo): 100 %**			

Tabela 5.3 – Cálculo de Eficiência para Fonte 1

Fonte: Terra – Guia de Cidades – São Paulo – Teatro e Dança (HTML)			
Url: http://cidades.terra.com.br/sao/espetaculos/0,7558,L:N:TODOS:1,00.html			
Consulta	**% *P***	**% *R***	**% *F***
Consulta 1: 23/06/2003	100	100	100
Consulta 2: 16/08/2003	100	100	100
Consulta 3: 18/08/2003	100	100	100
Consulta 4: 20/08/2003	100	100	100
Consulta 5: 22/08/2003	100	100	100
***E* (*F* acima do limite mínimo): 100 %**			

Tabela 5.4 – Cálculo de Eficiência para Fonte 2

Fonte: Nasdaq – Cotações de 10 Ações Selecionadas (XML)			
Url: http://quotes.nasdaq.com/Quote.dll?page=xml&mode=stock&symbol=X			
Consulta	**% *P***	**% *R***	**% *F***
Consulta 1: 19/08/2003	100	100	100
Consulta 2: 20/08/2003	100	100	100
Consulta 3: 22/08/2003	100	100	100
Consulta 4: 23/08/2003	100	100	100
Consulta 5: 24/08/2003	100	100	100
***E* (*F* acima do limite mínimo): 100 %**			

Tabela 5.5 – Cálculo de Eficiência para Fonte 3

Fonte: Climatempo – Condições Climáticas nos Aeroportos do Brasil (HTML)			
Url: http://www.climatempo.com.br/novo/aeroportos_on_line.php			
Consulta	**% *P***	**% *R***	**% *F***
Consulta 1: 16/08/2003	98,2	93,3	95,7
Consulta 2: 18/08/2003	99,1	98,5	98,8
Consulta 3: 20/08/2003	99,0	94,2	96,5
Consulta 4: 23/08/2003	98,6	96,4	97,5
Consulta 5: 24/08/2003	99,3	92,1	95,6
***E* (*F* acima do limite mínimo): 100 %**			

Tabela 5.6 – Cálculo de Eficiência para Fonte 4

Fonte: Estadão – Indicadores Econômicos – Taxas de Referência (HTML)			
Url: http://www.estadao.com.br/economia/financas/cotacoes/indic.htm			
Consulta	**% *P***	**% *R***	**% *F***
Consulta 1: 16/08/2003	100	100	100
Consulta 2: 18/08/2003	100	100	100
Consulta 2: 20/08/2003	100	100	100
Consulta 4: 22/08/2003	100	100	100
Consulta 5: 23/08/2003	100	100	100
***E* (*F* acima do limite mínimo): 100 %**			

Tabela 5.7 – Cálculo de Eficiência para Fonte 5

5.6.3 Análise dos Resultados

Os resultados de eficiência da extração, obtidos nas medições realizadas após evento de manutenção da estração, foram bastante expressivos, com média harmônica superior a 95%, na totalidade dos casos. Os resultados foram considerados satisfatórios, uma vez que a eficiência da extração não foi comprometida significativamente pelos eventos de manutenção da extração.

Nos casos nos quais as medidas de precisão e *recall* ficaram abaixo de 100%, podem ser apontadas como principais causas:

- Não reconhecimento de uma estrutura aninhada, para a fonte 1: em um grupo aninhado de dois atributos presentes nas tuplas, apenas um foi reconhecido pelo processo de extração;
- Elevado grau de similaridade entre delimitadores de diferentes tipos de atributo, para a fonte 4: o alto grau de semelhança entre delimitadores

acarretou problemas na fase de montagem das tuplas, acarretando reconhecimento de atributos com tipos errados, e agrupamento de atributos em tuplas erradas.

A ocorrência destes problemas, no entanto, não prejudicou o processo de extração de forma generalizada, uma fez que este manteve boas medidas de eficiência, mesmo nestas situações, o que vem reiterar o grau de resiliência satisfatório proporcionado pela técnica proposta.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Contribuições

Este trabalho propôs uma abordagem para o problema da resiliência em extrações de dados a partir de fontes Web, que pode ser resumida nos pontos principais listados a seguir:

- Estratégia de extração *bottom-up*, que permite o melhor tratamento de variações de estrutura, comuns em dados semi-estruturados, assim como sua utilização para diferentes formatos de documentos em linguagem de marcação, como HTML, XML e SGML;
- Funcionalidade de verificação do processo de extração, capaz de sinalizar quando o mesmo encontra-se em situação de falha;
- Funcionalidade de manutenção automática do processo de extração, baseada em dados históricos extraídos da fonte, que permite total independência da sintaxe de formatação e estruturação dos documentos.

Além da proposta em si, o trabalho apresenta uma implementação para esta abordagem, que complementa as contribuições, principalmente através da organização do desenvolvimento da solução na forma de um *framework* de classes Java, o que permite a fácil extensão futura das funcionalidades, como a troca de algorítmos, assim como o uso das classes genéricas em outros trabalhos.

A implementação de referência apresentada é construída utilizando o *framework* proposto, e baseia-se em padrões de extração expressos através de expressões regulares. A tarefa principal – a manutenção da extração – é implementada em três passos, através de três algoritmos:

- Obtém Instâncias de Atributo Candidatas: a partir de valores de atributo presentes no catálogo histórico de extrações, obtém fragmentos do novo

documento que contém um destes valores, como possível instância do tipo de atributo, que chamamos de instância candidata;

- Obtém Delimitadores Distintos: agrupa as diversas variações para os delimitadores (anteriores e posteriores, separadamente) de cada tipo de atributo, ordenando as variações encontradas por freqüência de ocorrência;
- Generaliza Expressão Regular: seguindo a ordem de freqüência dos delimitadores distintos, e tomando o mais freqüente como ponto inicial, vai passo a passo generalizando uma expressão regular que reconheça as variações; o ponto de corte desse processo é um limite do número de variações que entrarão na generalização, já que podem haver falsos candidatos no conjunto.

Comparando com outros trabalhos que tratam de extração resiliente, destacamos os seguintes pontos:

- Suporte a fontes em linguagem de marcação não-HTML, como XML e SGML;
- Saída da extração em formato RDF;
- Funcionalidades de verificação e manutenção da extração;
- Automatização total da manutenção da extração;

Outra importante contribuição é a contextualização dos trabalhos de extração de dados dentro do cenário das tecnologias Web, explicitando a relação entre técnicas de extração de dados de fontes HTML e técnicas de manipulação de documentos XML, como XSLT e XQuery. Dentro do conceito desenvolvido, estas tecnologias são classificadas como soluções de extração de dados.

Ainda mais importante é a sinalização da importância das técnicas de extração de dados no contexto da chamada Web Semântica. Este trabalho demonstra como as tecnologias de extração de dados, em particular as técnicas resilientes para extração de dados, podem desempenhar papel fundamental na evolução rumo à Web Semântica, através da disponibilização de dados em formato RDF, com impacto reduzido para as fontes provedoras de informação.

6.2 Trabalhos Futuros

6.2.1 Uso de XSLT 2.0 para extrações de fontes XML

A versão 2.0 da especificação XSLT [Kay03] traz, como novidade, o suporte ao reconhecimento de padrões de *strings*, através de expressões regulares. A funcionalidade está sendo incorporada, na verdade, pela especificação XPath 2.0 [BCF+03]. O suporte é provido através das funções `fn:matches`, `fn:replace` e `fn:tokenize`.

Uma possível evolução do trabalho seria a tentativa de utilizar *stylesheets* XSLT 2.0 para realizar a extração de dados a partir de documentos XML. Ao invés de uma classe Java ler o modelo de extração, obter os padrões de extração e controlar a extração dos atributos, haveria uma classe Java que, a partir do modelo de extração em formato RDF *Schema*, geraria uma *stylesheet* XSLT 2.0, que seria utilizada na extração.

6.2.2 Integração com soluções de interoperabilidade baseadas em RDF

Um objetivo natural da geração de dados em formato RDF é a disponibilização dos dados para usos externos de alto nível, como consultas realizadas por agentes, ou a participação em cenários de integração e interoperabilidade.

Seria interessante utilizar as técnicas apresentadas neste trabalho para a montagem de cenários de interoperabilidade semântica entre fontes Web. O diferencial seria que não ficaria sob responsabilidade das fontes a disponibilização dos dados RDF, que seriam obtidos através das técnicas propostas neste trabalho.

6.2.3 Geração e manutenção semi-automatizada de Ontologias

Como foi mencionado anteriormente, uma das grandes dificuldades para a construção de cenários de integração baseados em ontologias é o elevado esforço necessário para a construção destas.

As técnicas apresentadas neste trabalho, voltadas à geração e manutenção de modelos de extração, poderiam ser adaptadas para a tarefa de obtenção semi-

automatizada e manutenção de ontologias utilizadas em cenários de integração e interoperabilidade entre fontes XML, funcionalidade explorada por alguns trabalhos da área de ontologias para Web [Hwa99, DEFS99].

7

Apêndice A – Expressões Regulares

7.1 Definição

Como vimos, uma expressão regular representa um padrão, no qual se enquadram um conjunto de strings. De uma maneira mais formal, chamamos este conjunto da linguagem descrita pela expressão regular. Como vemos, para aprofundarmos o conceito de expressões regulares, é preciso estabelecer primeiramente alguns outros conceitos.

Para definir linguagem, precisamos primeiramente definir alfabeto. Um alfabeto é um conjunto finito de símbolos, usado para formar as strings pertencentes a uma linguagem. Por exemplo, podemos ter um alfabeto E formado pelos símbolos a e b, ou seja, $E = \{a, b\}$. Uma string sobre um alfabeto é uma seqüência, de comprimento finito, de símbolos deste alfabeto. Assim, *a*, *b*, *aa*, *ab*, *bba*, *baba*, *aaaaabbaab*, são todas strings sobre E. A string vazia, denotada pelo símbolo ε, é a string sem símbolos, que é uma string sobre qualquer alfabeto. O fechamento de E, representado por E^*, é o conjunto de todas as strings sobre E. Portanto, em nosso exemplo,

$E^* = \{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, ...\}$

Uma linguagem sobre um alfabeto é um conjunto de strings sobre este alfabeto. São exemplos de linguagens sobre E:

$\{\varepsilon, a, aa, bb\}$

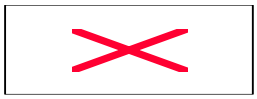
$\{x \in \{a, b\}^* \mid$ x começa com a e termina com $b\}$

Como linguagens são conjuntos, é possível gerar novas linguagens aplicando operações de conjuntos sobre linguagens existentes. Por exemplo, se L_1 e L_2 são

linguagens sobre E, então $L_1 \cup L_2$, $L_1 \cap L_2$ e $L_1 - L_2$ são também linguagens sobre E.

Além das operações de conjuntos, temos a operação de concatenação, que define a linguagem L_1L_2 como $\{xy \mid x \in L_1 \text{ e } y \in L_2\}$, onde xy é a concatenação das strings x e y.

Estendendo o conceito de concatenação, podemos definir L^k como sendo a concatenação de L com L, e o resultado novamente com L, k vezes, onde $L^0 = \{\varepsilon\}$ e $L^1 = L$. O fechamento de Kleene de L, ou simplesmente fechamento de L, denotado por L^*, pode ser assim definido:



Estamos interessados em um subconjunto das linguagens existentes, que são as linguagens regulares. O conjunto das linguagens regulares contém todas as linguagens formadas por uma única string de comprimento 1, ou pela string vazia unicamente. Além destas, contém também todas as linguagens que possam ser formadas aplicando-se as operações de união, concatenação e fechamento às strings iniciais, e também sobre as strings resultantes que forem sendo obtidas.

De outro modo, dizemos que uma linguagem é regular quando pode ser descrita por uma expressão regular. Assim, o conjunto de strings que se enquadram no padrão descrito por uma expressão regular forma uma linguagem regular. Segue uma definição formal de expressões regulares em função das linguagens que as mesmas descrevem, que pode ser encontrada em [AHU74].

Definição. Seja I um alfabeto. As expressões regulares sobre I e as linguagens (sobre I) que elas descrevem são definidas recursivamente como se segue:

1. $\emptyset$ é a expressão regular que descreve o conjunto vazio.
2. ε é a expressão regular que descreve o conjunto $\{\varepsilon\}$.
3. Para cada símbolo a em I, a é a expressão regular que descreve o conjunto $\{a\}$.

4. Se p e q são expressões regulares que descrevem as linguagem regulares P e Q, respectivamente, então $(p + q)$, (pq) e (p^*) são expressões regulares que descrevem as linguagens regulares $P \cup Q$, PQ e P^*, respectivamente.

7.2 Busca por expressões regulares, autômatos finitos e complexidade

Busca baseada em expressões regulares é feita construindo-se um autômato finito que encontra todas as ocorrências de strings que se enquadram presentes no texto.

Um autômato finito é um dispositivo simples para reconhecimento de strings, que são lidas a partir de uma fita de entrada dividida em quadrados, cada um trazendo um símbolo, que são lidos, um de cada vez, seqüencialmente da esquerda para a direita, por uma cabeça de leitura. O dispositivo encontra-se sempre em um dos estados previstos, sendo este conjunto de estados finito. Dentre estes estados, destacam-se o estado inicial, e alguns estados finais ou de aceite. Em seu funcionamento, um autômato finito realiza movimentos. Em um movimento, o dispositivo pode assumir um novo estado, dependendo do estado e do símbolo atuais, e a cabeça de leitura pode mover para o próximo símbolo.

Existem diferentes tipos de autômatos finitos, que se diferenciam exatamente nas regras que regem os movimentos. Em movimentos de um **autômato finito determinístico (AFD)**, para cada combinação de estado atual e símbolo, existe um único próximo estado definido, e a cabeça de leitura deve obrigatoriamente mover um quadrado para a direita.

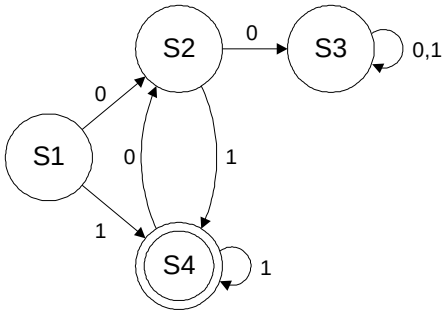


Figura 7.1 – Autômato Finito Determinístico (AFD)

Já para um um **autômato finito não determinístico (AFND)**, existem zero ou mais opções de próximo estado para cada combinação de estado e símbolo, e também existe a opção de mudar de estado sem mover a cabeça de leitura.

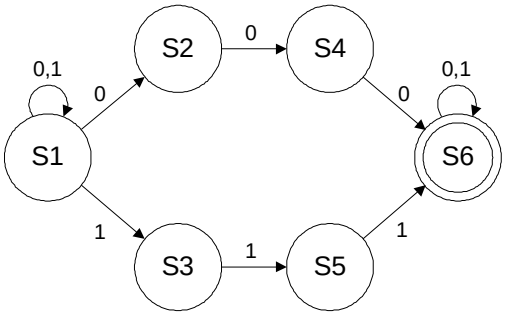


Figura 7.2 – Autômato Finito Não-Determinístico (AFND)

Autômatos finitos são normalmente representados na forma de um diagrama de transição, como os apresentados nas figuras acima. Para um programa de computador, a função de transição de um autômato finito é representada por uma tabela de transição, que possui uma linha para cada possível estado do autômato, e uma coluna para cada símbolo de entrada possível, e mais uma coluna para ε. Uma posição da tabela, localizada na intersecção da linha correspondente ao estado *s* com a coluna correspondente ao símbolo *c*, traz o conjunto de estados { s_i, s_{i+1} , ... , s_j} acessíveis a partir do estado *s*, recebendo-se na entrada o caracter *c*. Para um AFND, cada posição da tabela de transição guarda um conjunto de estados, enquanto a tabela de transição de um AFD traz apenas um estado em cada posição.

Para uma dada expressão regular α, existe um autômato finito não-determinístico que reconhece a linguagem descrita por α. Existem duas estratégias principais para a construção de um reconhecedor de expressões regulares, ou seja, um programa capaz de realizar busca por strings que se enquadrem no padrão determinado por uma expressão regular. Uma estratégia é

construir um AFND a partir da expressão regular, e em seguida simular o comportamento deste AFND. Esta estratégia é chamada de estratégia AFND. A outra estratégia principal, conhecida como estratégia AFD, constrói-se inicialmente o AFND, mas ao invés de simulá-lo diretamente, gera-se um AFD a partir do AFND, e o AFD é simulado.

Seguindo a estratégia AFND, a construção do AFND a partir da expressão regular leva tempo $O(m)$, onde m é o comprimento da expressão regular. A tabela de transição criada terá tamanho $O(m)$. Para realizar busca por strings que se enquadrem no padrão determinado pela expressão regular em um texto de comprimento n, será preciso tempo $O(m \times n)$. Desta forma, a estratégia AFND necessita de tempo total $O(m \times n)$ e espaço total $O(m)$. Esta estratégia apresenta melhor resultado quando a busca se realiza somente uma vez, e a string a ser vasculhada não é muito longa.

Usando-se a estratégia AFD, uma vez gerado o AFD, a busca por strings no texto de entrada de comprimento n é feita em tempo $O(n)$. No entanto, a tabela de transição do AFND pode chegar a um tamanho $O(2^m)$. Recomenda-se o uso desta estratégia quando a busca é feita repetidas vezes, pois o tempo consumido na conversão do AFND em um AFD, que é $O(m^2)$, dilui-se. Sempre que é possível gastar mais espaço para se economizar tempo, a estratégia AFD é interessante.

Dentre as linguagens de programação e ferramentas com suporte para expressões regulares, umas usam estratégia AFND, enquanto outras utilizam estratégia AFD. Estratégia AFND é usada em Tcl, Perl, Python, Emacs, ed, sed, vi e mawk , enquanto estratégia AFD é usada em grep, egrep, awk, lex, flex e expr.

A parte implementada do presente trabalho foi desenvolvida em Java, que a partir da versão 1.4, traz suporte nativo para expressões regulares, através do pacote java.util.regex. [J2SE].

8

Apêndice B – Exemplo de Extração

Neste apêndice, é apresentado um exemplo de conjunto de documentos de um cenário de extração que utiliza a abordagem proposta no trabalho. São mostrados o documento original, do qual se deseja extrair os dados, o modelo de extração, que configura o processo de extração para a fonte específica, e o documento RDF de resultado da extração. Por questões de espaço, apenas um exemplo é apresentado.

8.1 Nasdaq – Cotações de 10 Ações Selecionadas (XML)

8.1.1 Documento Original

Para esta fonte de dados, o documento alvo do processo de extração é, na realidade, a concatenação de 10 outros documentos, gerada em um passo inicial de pré-processamento.

```
<?xml version="1.0" ?>
<!--<!DOCTYPE nasdaqamex-dot-com SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
<!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

Copyright
...
 -->

<nasdaqamex-dot-com>
    <equity-quote   symbol="DELL"   ilx-symbol="DELL"   hyperfeed-symbol="DELL"
telesphere-symbol="DELL" cusip="24702R10">
        <issue-name>Dell Inc.</issue-name>
        <market-status>C</market-status>
        <market-center-code>NASDAQ-NM</market-center-code>
        <issue-type-code>Common Stock</issue-type-code>
        <todays-high-price>33.7</todays-high-price>
        <todays-low-price>32.74</todays-low-price>
        <fifty-two-wk-high-price>34.52</fifty-two-wk-high-price>
```

```
        <fifty-two-wk-low-price>22.59</fifty-two-wk-low-price>
        <last-sale-price>32.82</last-sale-price>
        <net-change-price>-0.39</net-change-price>
        <net-change-pct>-1.17%</net-change-pct>
        <share-volume-qty>21078175</share-volume-qty>
        <previous-close-price>33.21</previous-close-price>
        <best-bid-price>32.81</best-bid-price>
        <best-ask-price>32.82</best-ask-price>
        <best-bid-price session-type="AfterHours">32.6</best-bid-price>
        <best-ask-price session-type="AfterHours">32.65</best-ask-price>
        <current-pe-ratio>36.0659</current-pe-ratio>
        <total-outstanding-shares-qty>2568252000</total-outstanding-shares-qty>
        <current-yield-pct>0.000000</current-yield-pct>
        <earnings-actual-eps-amt>0.91</earnings-actual-eps-amt>
        <cash-dividend-amt>0</cash-dividend-amt>
        <cash-dividend-ex-date>19691231</cash-dividend-ex-date>
        <sp500-beta-num>1.320000</sp500-beta-num>
        <trade-datetime>20030819 16:00:00</trade-datetime>
        <issuer-address-line1-txt>One Dell Way</issuer-address-line1-txt>
        <issuer-city-state-zip-txt>Round Rock TX 78682  United States</issuer-
city-state-zip-txt>
        <issuer-phone-num> 512-338-4400</issuer-phone-num>
        <issuer-web-site-url>http://www.dell.com/</issuer-web-site-url>
        <issuer-logo-url>http://content.nasdaq.com/logos/DELL.GIF</issuer-logo-
url>
        <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=101133&amp;client=thenasdaq&amp;ticke
r=DELL</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>84290030640</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
 </nasdaqamex-dot-com>

 <?xml version="1.0" ?>
 <!--<!DOCTYPE                    nasdaqamex-dot-com                    SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
 <!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

 Copyright
 ...
  -->

 <nasdaqamex-dot-com>
    <equity-quote   symbol="VRSN"   ilx-symbol="VRSN"   hyperfeed-symbol="VRSN"
telesphere-symbol="VRSN" cusip="92343E10">
```

```
        <issue-name>VeriSign, Inc.</issue-name>
        <market-status>C</market-status>
        <market-center-code>NASDAQ-NM</market-center-code>
        <issue-type-code>Common Stock</issue-type-code>
        <todays-high-price>14.26</todays-high-price>
        <todays-low-price>13.48</todays-low-price>
        <fifty-two-wk-high-price>16.2</fifty-two-wk-high-price>
        <fifty-two-wk-low-price>3.92</fifty-two-wk-low-price>
        <last-sale-price>14.17</last-sale-price>
        <net-change-price>0.33</net-change-price>
        <net-change-pct>2.38%</net-change-pct>
        <share-volume-qty>2895781</share-volume-qty>
        <previous-close-price>13.84</previous-close-price>
        <best-bid-price>14.16</best-bid-price>
        <best-ask-price>14.17</best-ask-price>
        <best-bid-price session-type="AfterHours">13.16</best-bid-price>
        <best-ask-price session-type="AfterHours">14.17</best-ask-price>
        <current-pe-ratio>NE</current-pe-ratio>
        <total-outstanding-shares-qty>238756000</total-outstanding-shares-qty>
        <current-yield-pct>0.000000</current-yield-pct>
        <earnings-actual-eps-amt>-1.33</earnings-actual-eps-amt>
        <cash-dividend-amt>0</cash-dividend-amt>
        <cash-dividend-ex-date>19691231</cash-dividend-ex-date>
        <sp500-beta-num>2.040000</sp500-beta-num>
        <trade-datetime>20030819 16:00:00</trade-datetime>
        <issuer-address-line1-txt>1390 Shorebird Way</issuer-address-line1-txt>
        <issuer-city-state-zip-txt>Mountain    View    CA    94043        United
States</issuer-city-state-zip-txt>
        <issuer-phone-num> 650-961-7500</issuer-phone-num>
        <issuer-web-site-url>http://www.verisign.com</issuer-web-site-url>
        <issuer-logo-url>http://content.nasdaq.com/logos/VRSN.GIF</issuer-logo-
url>
        <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=100903&amp;client=thenasdaq&amp;ticke
r=VRSN</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>3383172520</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
 </nasdaqamex-dot-com>

 <?xml version="1.0" ?>
 <!--<!DOCTYPE                    nasdaqamex-dot-com                    SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
 <!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.
```

```
<nasdaqamex-dot-com>
    <equity-quote   symbol="SYMC"   ilx-symbol="SYMC"   hyperfeed-symbol="SYMC"
telesphere-symbol="SYMC" cusip="87150310">
        <issue-name>Symantec Corporation</issue-name>
        <market-status>C</market-status>
        <market-center-code>NASDAQ-NM</market-center-code>
        <issue-type-code>Common Stock</issue-type-code>
        <todays-high-price>50.28</todays-high-price>
        <todays-low-price>48.65</todays-low-price>
        <fifty-two-wk-high-price>51.53</fifty-two-wk-high-price>
        <fifty-two-wk-low-price>27.21</fifty-two-wk-low-price>
        <last-sale-price>50.25</last-sale-price>
        <net-change-price>1.11</net-change-price>
        <net-change-pct>2.26%</net-change-pct>
        <share-volume-qty>3557365</share-volume-qty>
        <previous-close-price>49.14</previous-close-price>
        <best-bid-price>50.25</best-bid-price>
        <best-ask-price>50.26</best-ask-price>
        <best-bid-price session-type="AfterHours">49.8</best-bid-price>
        <best-ask-price session-type="AfterHours">50.65</best-ask-price>
        <current-pe-ratio>31.6038</current-pe-ratio>
        <total-outstanding-shares-qty>147711000</total-outstanding-shares-qty>
        <current-yield-pct>0.000000</current-yield-pct>
        <earnings-actual-eps-amt>1.59</earnings-actual-eps-amt>
        <cash-dividend-amt>0</cash-dividend-amt>
        <cash-dividend-ex-date>19691231</cash-dividend-ex-date>
        <sp500-beta-num>1.250000</sp500-beta-num>
        <trade-datetime>20030819 16:00:00</trade-datetime>
        <issuer-address-line1-txt>10201 Torre Avenue</issuer-address-line1-txt>
        <issuer-city-state-zip-txt>Cupertino CA 950142132 United States</issuer-
city-state-zip-txt>
        <issuer-phone-num> 408-253-9600</issuer-phone-num>
        <issuer-web-site-url>http://www.symantec.com/</issuer-web-site-url>
        <issuer-logo-url>http://content.nasdaq.com/logos/SYMC.GIF</issuer-logo-
url>
        <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=89422&amp;client=thenasdaq&amp;ticker
=SYMC</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>7422477750</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
```

```
 </nasdaqamex-dot-com>

 <?xml version="1.0" ?>
 <!--<!DOCTYPE                      nasdaqamex-dot-com                      SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
 <!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

 Copyright
 ...
  -->

 <nasdaqamex-dot-com>
     <equity-quote   symbol="YHOO"   ilx-symbol="YHOO"   hyperfeed-symbol="YHOO"
telesphere-symbol="YHOO" cusip="98433210">
         <issue-name>Yahoo! Inc.</issue-name>
         <market-status>C</market-status>
         <market-center-code>NASDAQ-NM</market-center-code>
         <issue-type-code>Common Stock</issue-type-code>
         <todays-high-price>32.29</todays-high-price>
         <todays-low-price>31.352</todays-low-price>
         <fifty-two-wk-high-price>35.79</fifty-two-wk-high-price>
         <fifty-two-wk-low-price>8.94</fifty-two-wk-low-price>
         <last-sale-price>32.03</last-sale-price>
         <net-change-price>0.74</net-change-price>
         <net-change-pct>2.36%</net-change-pct>
         <share-volume-qty>13738038</share-volume-qty>
         <previous-close-price>31.29</previous-close-price>
         <best-bid-price>32.02</best-bid-price>
         <best-ask-price>32.03</best-ask-price>
         <best-bid-price session-type="AfterHours">31.7</best-bid-price>
         <best-ask-price session-type="AfterHours">31.8</best-ask-price>
         <current-pe-ratio>110.4483</current-pe-ratio>
         <total-outstanding-shares-qty>603222000</total-outstanding-shares-qty>
         <current-yield-pct>0.000000</current-yield-pct>
         <earnings-actual-eps-amt>0.29</earnings-actual-eps-amt>
         <cash-dividend-amt>0</cash-dividend-amt>
         <cash-dividend-ex-date>19691231</cash-dividend-ex-date>
         <sp500-beta-num>1.790000</sp500-beta-num>
         <trade-datetime>20030819 16:00:00</trade-datetime>
         <issuer-address-line1-txt>3400 Central Expressway</issuer-address-line1-
txt>
         <issuer-address-line2-txt>Suite 201</issuer-address-line2-txt>
         <issuer-city-state-zip-txt>Santa Clara CA 95051  United States</issuer-
city-state-zip-txt>
         <issuer-phone-num> 408-731-3300</issuer-phone-num>
         <issuer-web-site-url>http://www.yahoo.com</issuer-web-site-url>
         <issuer-logo-url>http://content.nasdaq.com/logos/YHOO.GIF</issuer-logo-
url>
         <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
```

```
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=104252&amp;client=thenasdaq&amp;ticke
r=YHOO</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>19321200660</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
 </nasdaqamex-dot-com>

 <?xml version="1.0" ?>
 <!--<!DOCTYPE                      nasdaqamex-dot-com                      SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
 <!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

 Copyright
 ...
  -->

 <nasdaqamex-dot-com>
    <equity-quote   symbol="EBAY"   ilx-symbol="EBAY"   hyperfeed-symbol="EBAY"
telesphere-symbol="EBAY" cusip="27864210">
        <issue-name>eBay Inc.</issue-name>
        <market-status>C</market-status>
        <market-center-code>NASDAQ-NM</market-center-code>
        <issue-type-code>Common Stock</issue-type-code>
        <todays-high-price>111.35</todays-high-price>
        <todays-low-price>107.75</todays-low-price>
        <fifty-two-wk-high-price>117.86</fifty-two-wk-high-price>
        <fifty-two-wk-low-price>50.224</fifty-two-wk-low-price>
        <last-sale-price>110.23</last-sale-price>
        <net-change-price>1.97</net-change-price>
        <net-change-pct>1.82%</net-change-pct>
        <share-volume-qty>9564653</share-volume-qty>
        <previous-close-price>108.26</previous-close-price>
        <best-bid-price>110.17</best-bid-price>
        <best-ask-price>110.03</best-ask-price>
        <best-bid-price session-type="AfterHours">109.6</best-bid-price>
        <best-ask-price session-type="AfterHours">109.97</best-ask-price>
        <current-pe-ratio>116.0316</current-pe-ratio>
        <total-outstanding-shares-qty>317498000</total-outstanding-shares-qty>
        <current-yield-pct>0.000000</current-yield-pct>
        <earnings-actual-eps-amt>0.95</earnings-actual-eps-amt>
        <cash-dividend-amt>0</cash-dividend-amt>
        <cash-dividend-ex-date>19691231</cash-dividend-ex-date>
        <sp500-beta-num>1.050000</sp500-beta-num>
        <trade-datetime>20030819 16:00:00</trade-datetime>
        <issuer-address-line1-txt>2005   Hamilton   Avenue</issuer-address-line1-
txt>
        <issuer-address-line2-txt>Suite 350</issuer-address-line2-txt>
```

```
        <issuer-city-state-zip-txt>San Jose CA 95125 United States</issuer-city-
state-zip-txt>
        <issuer-phone-num> (408) 369-4830</issuer-phone-num>
        <issuer-web-site-url>http://www.ebay.com</issuer-web-site-url>
        <issuer-logo-url>http://content.nasdaq.com/logos/EBAY.GIF</issuer-logo-
url>
        <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=108016&amp;client=thenasdaq&amp;ticke
r=EBAY</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>34997804540</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
 </nasdaqamex-dot-com>


 <?xml version="1.0" ?>
 <!--<!DOCTYPE                    nasdaqamex-dot-com                    SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
 <!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

 Copyright
 ...
  -->

 <nasdaqamex-dot-com>
     <equity-quote   symbol="ADBE"   ilx-symbol="ADBE"   hyperfeed-symbol="ADBE"
telesphere-symbol="ADBE" cusip="00724F10">
        <issue-name>Adobe Systems Incorporated</issue-name>
        <market-status>C</market-status>
        <market-center-code>NASDAQ-NM</market-center-code>
        <issue-type-code>Common Stock</issue-type-code>
        <todays-high-price>36.89</todays-high-price>
        <todays-low-price>35.81</todays-low-price>
        <fifty-two-wk-high-price>40</fifty-two-wk-high-price>
        <fifty-two-wk-low-price>17.71</fifty-two-wk-low-price>
        <last-sale-price>36.8</last-sale-price>
        <net-change-price>0.59</net-change-price>
        <net-change-pct>1.63%</net-change-pct>
        <share-volume-qty>2766520</share-volume-qty>
        <previous-close-price>36.21</previous-close-price>
        <best-bid-price>36.69</best-bid-price>
        <best-ask-price>36.67</best-ask-price>
        <best-bid-price session-type="AfterHours">36.5</best-bid-price>
        <best-ask-price session-type="AfterHours">36.75</best-ask-price>
        <current-pe-ratio>42.7907</current-pe-ratio>
```

```
        <total-outstanding-shares-qty>231852000</total-outstanding-shares-qty>
        <current-yield-pct>0.200000</current-yield-pct>
        <earnings-actual-eps-amt>0.86</earnings-actual-eps-amt>
        <cash-dividend-amt>0.05</cash-dividend-amt>
        <cash-dividend-ex-date>20030619</cash-dividend-ex-date>
        <sp500-beta-num>1.530000</sp500-beta-num>
        <trade-datetime>20030819 16:00:00</trade-datetime>
        <issuer-address-line1-txt>345 Park Avenue</issuer-address-line1-txt>
        <issuer-city-state-zip-txt>San Jose CA 951102704 United States</issuer-
city-state-zip-txt>
        <issuer-phone-num> 408-536-6000</issuer-phone-num>
        <issuer-web-site-url>http://www.adobe.com/</issuer-web-site-url>
        <issuer-logo-url>http://content.nasdaq.com/logos/ADBE.GIF</issuer-logo-
url>
        <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=110578&amp;client=thenasdaq&amp;ticke
r=ADBE</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>8532153600</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
</nasdaqamex-dot-com>

<?xml version="1.0" ?>
<!--<!DOCTYPE                    nasdaqamex-dot-com                    SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
<!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

Copyright
...
 -->

<nasdaqamex-dot-com>
    <equity-quote   symbol="NXTL"   ilx-symbol="NXTL"   hyperfeed-symbol="NXTL"
telesphere-symbol="NXTL" cusip="65332V10">
        <issue-name>Nextel Communications, Inc.</issue-name>
        <market-status>C</market-status>
        <market-center-code>NASDAQ-NM</market-center-code>
        <issue-type-code>Class A Common Stock</issue-type-code>
        <todays-high-price>18.96</todays-high-price>
        <todays-low-price>18.1</todays-low-price>
        <fifty-two-wk-high-price>20.53</fifty-two-wk-high-price>
        <fifty-two-wk-low-price>5.75</fifty-two-wk-low-price>
        <last-sale-price>18.81</last-sale-price>
        <net-change-price>0.75</net-change-price>
        <net-change-pct>4.15%</net-change-pct>
```

```
        <share-volume-qty>23877608</share-volume-qty>
        <previous-close-price>18.06</previous-close-price>
        <best-bid-price>18.81</best-bid-price>
        <best-ask-price>18.77</best-ask-price>
        <best-bid-price session-type="AfterHours">18.75</best-bid-price>
        <best-ask-price session-type="AfterHours">18.8</best-ask-price>
        <current-pe-ratio>7.8375</current-pe-ratio>
        <total-outstanding-shares-qty>992411000</total-outstanding-shares-qty>
        <current-yield-pct>0.000000</current-yield-pct>
        <earnings-actual-eps-amt>2.4</earnings-actual-eps-amt>
        <cash-dividend-amt>0</cash-dividend-amt>
        <cash-dividend-ex-date>19691231</cash-dividend-ex-date>
        <sp500-beta-num>1.750000</sp500-beta-num>
        <trade-datetime>20030819 16:00:00</trade-datetime>
        <issuer-address-line1-txt>2001   Edmund   Halley   Drive</issuer-address-
line1-txt>
        <issuer-city-state-zip-txt>Reston  VA  20191  United  States</issuer-city-
state-zip-txt>
        <issuer-phone-num> 703-433-4000</issuer-phone-num>
        <issuer-web-site-url>http://www.nextel.com</issuer-web-site-url>
        <issuer-logo-url>http://content.nasdaq.com/logos/NXTL.GIF</issuer-logo-
url>
        <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=63347&amp;client=thenasdaq&amp;ticker
=NXTL</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>18667250910</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
 </nasdaqamex-dot-com>

 <?xml version="1.0" ?>
 <!--<!DOCTYPE                    nasdaqamex-dot-com                    SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
 <!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

 Copyright
 ...
  -->

 <nasdaqamex-dot-com>
    <equity-quote   symbol="SBUX"   ilx-symbol="SBUX"   hyperfeed-symbol="SBUX"
telesphere-symbol="SBUX" cusip="85524410">
        <issue-name>Starbucks Corporation</issue-name>
        <market-status>C</market-status>
        <market-center-code>NASDAQ-NM</market-center-code>
```

```
        <issue-type-code>Common Stock</issue-type-code>
        <todays-high-price>27.43</todays-high-price>
        <todays-low-price>26.81</todays-low-price>
        <fifty-two-wk-high-price>27.769</fifty-two-wk-high-price>
        <fifty-two-wk-low-price>19.6</fifty-two-wk-low-price>
        <last-sale-price>27.03</last-sale-price>
        <net-change-price>-0.20</net-change-price>
        <net-change-pct>-0.73%</net-change-pct>
        <share-volume-qty>1875225</share-volume-qty>
        <previous-close-price>27.23</previous-close-price>
        <best-bid-price>27.08</best-bid-price>
        <best-ask-price>27.1</best-ask-price>
        <best-bid-price session-type="AfterHours">26.8</best-bid-price>
        <best-ask-price session-type="AfterHours">27.2</best-ask-price>
        <current-pe-ratio>41.5846</current-pe-ratio>
        <total-outstanding-shares-qty>390533000</total-outstanding-shares-qty>
        <current-yield-pct>0.000000</current-yield-pct>
        <earnings-actual-eps-amt>0.65</earnings-actual-eps-amt>
        <cash-dividend-amt>0</cash-dividend-amt>
        <cash-dividend-ex-date>19691231</cash-dividend-ex-date>
        <sp500-beta-num>0.950000</sp500-beta-num>
        <trade-datetime>20030819 16:00:00</trade-datetime>
        <issuer-address-line1-txt>2401 Utah Avenue South</issuer-address-line1-
txt>
        <issuer-city-state-zip-txt>Seattle WA 981340000 United States</issuer-
city-state-zip-txt>
        <issuer-phone-num> 206-447-1575</issuer-phone-num>
        <issuer-web-site-url>http://www.starbucks.com</issuer-web-site-url>
        <issuer-logo-url>http://content.nasdaq.com/logos/SBUX.GIF</issuer-logo-
url>
        <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=99518&amp;client=thenasdaq&amp;ticker
=SBUX</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>10556106990</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
 </nasdaqamex-dot-com>

 <?xml version="1.0" ?>
 <!--<!DOCTYPE                    nasdaqamex-dot-com                    SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">
 -->
 <!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

 Copyright
```

```
 ...
  -->

 <nasdaqamex-dot-com>
     <equity-quote   symbol="INTC"   ilx-symbol="INTC"   hyperfeed-symbol="INTC"
telesphere-symbol="INTC" cusip="45814010">
         <issue-name>Intel Corporation</issue-name>
         <market-status>C</market-status>
         <market-center-code>NASDAQ-NM</market-center-code>
         <issue-type-code>Common Stock</issue-type-code>
         <todays-high-price>26.54</todays-high-price>
         <todays-low-price>25.92</todays-low-price>
         <fifty-two-wk-high-price>26.23</fifty-two-wk-high-price>
         <fifty-two-wk-low-price>12.95</fifty-two-wk-low-price>
         <last-sale-price>26.47</last-sale-price>
         <net-change-price>0.28</net-change-price>
         <net-change-pct>1.07%</net-change-pct>
         <share-volume-qty>55968998</share-volume-qty>
         <previous-close-price>26.19</previous-close-price>
         <best-bid-price>26.5</best-bid-price>
         <best-ask-price>26.5</best-ask-price>
         <best-bid-price session-type="AfterHours">26.23</best-bid-price>
         <best-ask-price session-type="AfterHours">26.27</best-ask-price>
         <current-pe-ratio>49.0185</current-pe-ratio>
         <total-outstanding-shares-qty>6534000000</total-outstanding-shares-qty>
         <current-yield-pct>0.400000</current-yield-pct>
         <earnings-actual-eps-amt>0.54</earnings-actual-eps-amt>
         <cash-dividend-amt>0.08</cash-dividend-amt>
         <cash-dividend-ex-date>20030805</cash-dividend-ex-date>
         <sp500-beta-num>1.760000</sp500-beta-num>
         <trade-datetime>20030819 16:00:00</trade-datetime>
         <issuer-address-line1-txt>2200   Mission   College   Boulevard</issuer-
address-line1-txt>
         <issuer-city-state-zip-txt>Santa    Clara    CA    950528119    United
States</issuer-city-state-zip-txt>
         <issuer-phone-num> 408-765-8080</issuer-phone-num>
         <issuer-web-site-url>http://www.intc.com</issuer-web-site-url>
         <issuer-logo-url>http://content.nasdaq.com/logos/INTC.GIF</issuer-logo-
url>
         <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
         <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=101302&amp;client=thenasdaq&amp;ticke
r=INTC</issuer-audio-report-href-url>
         <trading-status>ACTIVE</trading-status>
         <market-capitalization-amt>172954980000</market-capitalization-amt>
         <option-root-symbol symbol=""/>
         <index-participation-elt>
             <index-quote symbol="IXNDX"/>
         </index-participation-elt>
     </equity-quote>
```

```
 </nasdaqamex-dot-com>

 <?xml version="1.0" ?>
 <!--<!DOCTYPE                    nasdaqamex-dot-com                    SYSTEM
"http://nasdaq.com/reference/NasdaqDotCom.dtd">-->
 <!-- (c) Copyright 2003, The Nasdaq Stock Market, Inc. All Rights Reserved.

 Copyright
 ...
  -->

 <nasdaqamex-dot-com>
     <equity-quote   symbol="SUNW"   ilx-symbol="SUNW"   hyperfeed-symbol="SUNW"
telesphere-symbol="SUNW" cusip="86681010">
         <issue-name>Sun Microsystems, Inc.</issue-name>
         <market-status>C</market-status>
         <market-center-code>NASDAQ-NM</market-center-code>
         <issue-type-code>Common Stock</issue-type-code>
         <todays-high-price>3.85</todays-high-price>
         <todays-low-price>3.6</todays-low-price>
         <fifty-two-wk-high-price>5.64</fifty-two-wk-high-price>
         <fifty-two-wk-low-price>2.34</fifty-two-wk-low-price>
         <last-sale-price>3.83</last-sale-price>
         <net-change-price>0.12</net-change-price>
         <net-change-pct>3.23%</net-change-pct>
         <share-volume-qty>57999032</share-volume-qty>
         <previous-close-price>3.71</previous-close-price>
         <best-bid-price>3.82</best-bid-price>
         <best-ask-price>3.82</best-ask-price>
         <best-bid-price session-type="AfterHours">3.76</best-bid-price>
         <best-ask-price session-type="AfterHours">3.78</best-ask-price>
         <current-pe-ratio>NE</current-pe-ratio>
         <total-outstanding-shares-qty>3229645000</total-outstanding-shares-qty>
         <current-yield-pct>0.000000</current-yield-pct>
         <earnings-actual-eps-amt>-0.76</earnings-actual-eps-amt>
         <cash-dividend-amt>0</cash-dividend-amt>
         <cash-dividend-ex-date>19691231</cash-dividend-ex-date>
         <sp500-beta-num>1.750000</sp500-beta-num>
         <trade-datetime>20030819 16:00:00</trade-datetime>
         <issuer-address-line1-txt>901  San  Antonio  Road</issuer-address-line1-
txt>
         <issuer-city-state-zip-txt>Palo     Alto     CA     94303-4900     United
States</issuer-city-state-zip-txt>
         <issuer-phone-num> 650-960-1300</issuer-phone-num>
         <issuer-web-site-url>http://www.sun.com/</issuer-web-site-url>
         <issuer-logo-url>http://content.nasdaq.com/logos/SUNW.GIF</issuer-logo-
url>
         <issuer-audio-report-logo-
url>http://content.nasdaq.com/images/conferenceCallsIcon.gif</issuer-audio-
report-logo-url>
```

```
        <issuer-audio-report-href-
url>http://nasdaq.ccbn.com/company.asp?coid=68203&amp;client=thenasdaq&amp;ticker
=SUNW</issuer-audio-report-href-url>
        <trading-status>ACTIVE</trading-status>
        <market-capitalization-amt>12369540350</market-capitalization-amt>
        <option-root-symbol symbol=""/>
        <index-participation-elt>
            <index-quote symbol="IXNDX"/>
        </index-participation-elt>
    </equity-quote>
 </nasdaqamex-dot-com>
```

Figura 8.1 – Nasdaq – Documento Original (XMLs concatenados)

8.1.2 Modelo de Extração

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                xmlns:rex="http://www.rex.org/rex-ns#"
                xml:base="http://www.rex.org/bolsa/conceitos">

    . . .

    <rdfs:Class rdf:ID="Acao"/>

    <rdf:Property rdf:ID="Simbolo">
        <rdfs:domain rdf:resource="#Acao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        <rex:extractionPattern>
            <![CDATA[ <equity-quote symbol="{WORD}" ilx-symbol=" ]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
    </rdf:Property>

    <rdf:Property rdf:ID="Nome">
        <rdfs:domain rdf:resource="#Acao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        < rex:extractionPattern>
            <![CDATA[ <issue-name>{MULTIPLE}</issue-name> ]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
    </rdf:Property>

    <rdf:Property rdf:ID="PrecoMaxHoje">
```

```
        <rdfs:domain rdf:resource="#Acao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        < rex:extractionPattern>
            <![CDATA[   <todays-high-price>{FORMATTED_NUMBER}</todays-high-price>
]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
    </rdf:Property>

    <rdf:Property rdf:ID="PrecoMinHoje">
        <rdfs:domain rdf:resource="#Acao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        < rex:extractionPattern>
            <![CDATA[ <todays-low-price>{FORMATTED_NUMBER}</todays-low-price> ]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
    </rdf:Property>

    <rdf:Property rdf:ID="PrecoFechamento">
        <rdfs:domain rdf:resource="#Acao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        < rex:extractionPattern>
            <![CDATA[ <last-sale-price>{FORMATTED_NUMBER}</last-sale-price> ]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
    </rdf:Property>

    <rdf:Property rdf:ID="VolumeNegociado">
        <rdfs:domain rdf:resource="#Acao"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        < rex:extractionPattern>
            <![CDATA[    <lshare-volume-qty>{FORMATTED_NUMBER}</share-volume-qty>
]]>
        </rex:extractionPattern>
        <rex:minOccurs>1</rex:minOccurs>
        <rex:maxOccurs>1</rex:maxOccurs>
    </rdf:Property>

</rdf:RDF>
```

Figura 8.2 – Nasdaq – Modelo de Extração (RDF *Schema*)

8.1.1 Resultado da Extração


```xml
<?xml version="1.0" encoding="UTF-16"?>
<Result rdf:ID="nasdaq_selected_quotes_2003-08-19"
                xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xml:base="http://www.rex.org/nasdaq/cotacoes">
   <Tuples>
   <rdf:Seq>
      <!--Tuple Nr.0-->
      <Acao rdf:ID="DELL">
         <Nome>Dell Inc.</Nome>
         <PrecoMaxHoje>33.7</PrecoMaxHoje>
         <PrecoMinHoje>32.74</PrecoMinHoje>
         <VolumeNegociado>21078175</VolumeNegociado>
      </Acao>
      <!--Tuple Nr.1-->
      <Acao rdf:ID="VRSN">
         <Nome>VeriSign, Inc.</Nome>
         <PrecoMaxHoje>14.26</PrecoMaxHoje>
         <PrecoMinHoje>13.48</PrecoMinHoje>
         <VolumeNegociado>2895781</VolumeNegociado>
      </Acao>
      <!--Tuple Nr.2-->
      <Acao rdf:ID="SYMC">
         <Nome>Symantec Corporation</Nome>
         <PrecoMaxHoje>50.28</PrecoMaxHoje>
         <PrecoMinHoje>48.65</PrecoMinHoje>
         <VolumeNegociado>3557365</VolumeNegociado>
      </Acao>
      <!--Tuple Nr.3-->
      <Acao rdf:ID="YHOO">
         <Nome>Yahoo! Inc.</Nome>
         <PrecoMaxHoje>32.29</PrecoMaxHoje>
         <PrecoMinHoje>31.352</PrecoMinHoje>
         <VolumeNegociado>13738038</VolumeNegociado>
      </Acao>
      <!--Tuple Nr.4-->
      <Acao rdf:ID="EBAY">
         <Nome>eBay Inc.</Nome>
         <PrecoMaxHoje>111.35</PrecoMaxHoje>
         <PrecoMinHoje>107.75</PrecoMinHoje>
         <VolumeNegociado>9564653</VolumeNegociado>
      </Acao>
      <!--Tuple Nr.5-->
      <Acao rdf:ID="ADBE">
         <Nome>Adobe Systems Incorporated</Nome>
         <PrecoMaxHoje>36.89</PrecoMaxHoje>
         <PrecoMinHoje>35.81</PrecoMinHoje>
         <VolumeNegociado>2766520</VolumeNegociado>
      </Acao>
```

```
      <!--Tuple Nr.6-->
      <Acao rdf:ID="NXTL">
         <Nome>Nextel Communications, Inc.</Nome>
         <PrecoMaxHoje>18.96</PrecoMaxHoje>
         <PrecoMinHoje>18.1</PrecoMinHoje>
         <VolumeNegociado>23877608</VolumeNegociado>
      </Acao>
      <!--Tuple Nr.7-->
      <Acao rdf:ID="SBUX">
         <Nome>Starbucks Corporation</Nome>
         <PrecoMaxHoje>27.43</PrecoMaxHoje>
         <PrecoMinHoje>26.81</PrecoMinHoje>
         <VolumeNegociado>1875225</VolumeNegociado>
      </Acao>
      <!--Tuple Nr.8-->
      <Acao rdf:ID="INTC">
         <Nome>Intel Corporation</Nome>
         <PrecoMaxHoje>26.54</PrecoMaxHoje>
         <PrecoMinHoje>25.92</PrecoMinHoje>
         <VolumeNegociado>55968998</VolumeNegociado>
      </Acao>
      <!--Tuple Nr.9-->
      <Acao rdf:ID="SUNW">
         <Nome>Sun Microsystems, Inc.</Nome>
         <PrecoMaxHoje>3.85</PrecoMaxHoje>
         <PrecoMinHoje>3.6</PrecoMinHoje>
         <VolumeNegociado>57999032</VolumeNegociado>
      </Acao>
   </rdf:Seq>
   </Tuples>
</Result>
```


Figura 8.3 – Nasdaq – Resultado da Extração (RDF)

Referências Bibliográficas

[AAB+98] J.L. Ambite, N. Ashish, G. Barish, C. A. Knoblock, S. Minton, P. J. Modi, I. Muslea, A. Philpot, and S. Tejada. *ARIADNE: A system for constructing mediators for internet sources*. Proceedings ACM SIGMOD International Conference on Management of Data, páginas 561-563, Seattle, Washington, USA, junho 1998

[ABC+01] S. Adler, A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, S. Zilles. *Extensible Stylesheet Language (XSL) Version 1.0*. W3C Recommendation, outubro 2001; http://www.w3.org/TR/xsl/

[Ade98] B. Adelberg. *NoDoSE – A tool for Semi-Automatically Extracting Structured and Semistructured Data from Text Documents*. Proceedings ACM SIGMOD International Conference on Management of Data, páginas 283-294, Seattle, Washington, USA, junho 1998.

[AHU74] A. V. Aho, J. E. Hopcroft, J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[AM98] G. Arocena, A.Meldenzon. *WebOQL: Restructuring Documents, Databases and Webs*. Proceedings of the Fourteenth International Conference on Data Engineering, páginas 24-33, Orlando, Flórida, USA, fevereiro 1998.

[Bra03] T. Bray. *What is RDF?* XML.com, 2003; http://www.xml.com/pub/a/2001/01/24/rdf.html

[Bei95] B. Beizer. *Black_Box Testing*. John Wiley & Sons, 1995.

[BCF+03] A. Berglund, D. Chamberlin, M. F. Fernández, J. Robie, J. Siméon. *XML Path Language (XPath) 2.0*. W3C Working Draft, agosto 2003; http://www.w3.org/TR/xpath20/

[BG03] D. Brickley, R.V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Working Draft, janeiro 2003; http://www.w3.org/TR/rdf-schema/

[BHM+03] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard. *Web Services Architecture*. W3C Working Draft, agosto 2003; http://www.w3.org/2002/ws

[BLF00] T. Berners-Lee, M. I. Fischetti. *Weaving the Web: the original design and ultimate destiny of the World Wide Web by its inventor*. HarperBusiness, 2000.

[BLHL01] T. Berners-Lee, J. Hendler, O. Lassila. *The Semantic Web*. Scientific American, maio 2001.

[BP98] S. Brin, L. Page. *The anatomy of a large-scale hypertextual web search engine*. Computer Networks and ISDN Systems, volume 30, número 1-7, páginas 107-117, abril 1998.

[BPSM00] T. Bray, J. Paoli, C. M. Sperberg-McQueen. *Extensible MarkupLanguage (XML) 1.0 (Second Edition)*. W3C Recommendation, outubro 2000; http://www.w3.org/TR/html4

[BR99] R. Baeza-Yates, B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[BRJ97] G. Booch, J. Rumbaugh e I. Jacobson. *Unified Modeling Language, Version 1.0*. Rational Software Corporation, janeiro 1997.

[Cla99] J. Clark. *XSL Transformations (XSLT) Version 1.0*. W3C Recommendation, novembro 1999; http://www.w3.org/TR/xslt

[CCD+98] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, L. Tanca. *XML-GL: A Graphical Language for Querying and Reshaping XML Documents*. W3C QL'98 Position Paper, dezembro 1998; http://www.w3.org/TandS/QL/QL98/pp/xql.html

[CCF+03] D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Siméon. *XQuery 1.0: An XML Query Language*. W3C Working Draft, agosto 2003; http://www.w3.org/TR/xquery

[CCS00] V. Christophides, S. Cluet e J. Siméon. *On Wrapping Query Languages and Efficient XML Integration.* Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, volume 29, número 2, páginas 141-152, Dallas, Texas, USA, maio 2000.

[CD99] J. Clark, S. DeRose. Horrocks. *XML Path Language (XPath) Version 1.0*, W3C Recomendation, novembro 1999; http://www.w3.org/TR/xpath

[CJ03] R. L. Costello, D. B. Jacobs. *XML Design (A Gentle Transition from XML to RDF)*. The MITRE Corporation, 2003; http://www.xfront.com/rdf/

[CM98] P. Cotton, A. Malhotra. *Candidate Requirements for XML Query*. W3C QL'98 Summary of Requirements, dezembro 1998; http://www.w3.org/TandS/QL/QL98/pp/queryreq.html

[CMM01] V. Crescenzi, G. Mecca, P. Merialdo. *RoadRunner: Towards Automatic Data Extraction from Large Web Sites*. Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001), páginas 109-118, Roma, Itália, setembro 2001.

[CRB98] M.Chalmers, K. Rodden, D. Brodbeck. *The order of things: activity-centred information access*. Computer Networks and ISDN Systems, volume 30, números 1-7, páginas 359-367, abril 1998.

[CT01] J. Cowam, R. Tobin. *XML Information Set*, W3C Recomendation, outubro 2001; http://www.w3.org/TR/xml-infoset/

[CT01] J. Cowam, Richard Tobin. *XML Information Set*, W3C Recomendation, outubro 2001; http://www.w3.org/TR/xml-infoset/

[DAML00] *DAML: Website Oficial*; 2000. http://www.daml.org/

[DEFS99] S. Decker, M. Erdmann, D. Fensel, R. Studer. *Ontobroker: Ontology based access to distributed and semi-structured information*. Database Semantics 8 (DS - 8), volume 138, páginas 351-369, Rotorua, Nova Zelândia, janeiro 1999.

[DFF+98] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu. *XML-QL: a query language for XML*. W3C Note, agosto 1998; http://citeseer.nj.nec.com/390950.html

[DK97] P. Datta, D. Kibler. *Learning symbolic prototypes*. Proceedings of the Fourteenth International Conference on Machine Learning (ICML), páginas 75-82. Nashville, TN, USA, julho 1997.

[ECJ+99] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y. – K. Ng, R. D. Smith. *Conceptual Model-Based Data Extraction form Multiple-Record Web Pages.* Data & Knowledge Engineering, volume 31, número 3, páginas 227-251, novembro 1999.

[FE01] L. Fegaras, R. Elmasri. *Query Engines for Web-Accessible XML Data*. Proceedings of 27th International Conference on Very Large Data Bases, páginas 251-260, Roma, Itália, setembro 2001.

[FH99] D. Fensel, F. van Harmelen, I. Horrocks. *OIL: A Standard Proposal for the Semantic Web*. European ITS Project Deliverable, novembro 1999.

[FLM98] D. Florescu, A. Y. Levy, and A. O. Mendelzon. *Database Techniques for the word-wide web: A survey*. SIGMOD Record, volume 27, número 3, páginas 59-74, setembro 1998.

[FMF00] D. Florescu, I. Manolescu, J. Freire. *Querying the Web*. Tutorial, XV Simpósio Brasileiro de Banco de Dados, outubro 2000.

[Gol78] C. Goldfarb. *Document Composition Facility: Generalized Markup Language (GML) User's Guide*. Publication SH20-9160-0, IBM General Products Division, 1978.

[Gol97] G. Golovchinsky. *What the query told the link: the integration of Hypertext and Information Retrieval*. The Eigth ACM Conference on Hypertext, páginas 67-74, Universidade de Southhampton, Inglaterra, 1997.

[GHJV94] E. Gamma, R. Helm, R.Johnson, J. Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[GMPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, e J. Widow. *The TSIMMIS project: Integration of heterogenous information sources*. Proceedings of the Meeting of the Processing Society of Japan, 1997.

[GOOG03] *Google: Página Principal*; agosto 2003. http://www.google.com

[GRVB98] J.-R. Gruser, L. Raschid, M. E. Vidal e L. Bright. *Wrapper Generation for web acessible data sources*. Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems, páginas 14-23, Nova Iorque, Nova Iorque, USA, agosto 1998.

[Hun01] J. Hunter. Adding *Multimedia to the Semantic Web - Building an MPEG-7 Ontology* First Semantic Web Working Symposium (SWWS), Stanford, USA, 2001.

[HD98] Chun-Nan Hsu, Ming-Tzung Dung. *Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web*. Information Systems, volume 23 número 8, páginas 521-538, dezembro 1998.

[HGMC+98] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, e A. Crespo. *Extracting Semistructured Information from the Web*. Proceedings of the Workshop on Management of Semistructured Data. Tucson, Arizona, maio 1998.

[HH00] J. Heflin, J. Hendler. *Semantic interoperability on the web*. Extreme Markup Languages 2000, Montreal, Canadá, agosto 2000.

[HHPS01] I. Horrocks, F. van Harmelen, P. Patel-Schneider. *DAML+OIL Language Release*; março 2001. http://www.daml.org/

[HHW+00] A. Le Hors, P. Le Hégaret, L. Wood, G. Nicol, J. Robie, M. Champion, S. Byrne. *Document Object Model (DOM) Level 2 Core Specification*. W3C Recommendation, novembro 2000; http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113

[Hwa99]C. H. Hwang. *Incompletely and imprecisely speaking: Using dynamic ontologies for representing and retrieving information*. Technical Report, Microelectronics and Computer Technology Corporation (MCC), junho 1999.

[ISO86] ISO 8879. Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML), 1986.

[JAVA02] Sun Microsystems, Inc. *Java™ 2 Platform, Standard Edition, v 1.4.1 API Specification*; 2002. http://java.sun.com/j2se/1.4.1/docs/api/

[JC01] L. Jensen, W. Cohen. *Grouping extracted fields*. Proceedings of the Workshop on Adaptive Text Extraction and Mining, 2001.

[Kay03] M. Kay. *XSL Transformations (XSLT) Version 2.0*. W3C Working Draft, maio 2003; http://www.w3.org/TR/xslt20/

[Kus00a] N. Kushmerick. *Wrapper Verification*. World Wide Web Journal, volume 3, número 2, páginas 79-94, 2000.

[Kus00b] N. Kushmerick. *Wrapper induction: Efficiency and expressiveness*. Artificial Inteligence Journal, volume 118, números 1 e 2, páginas 15-68, Kansas, Missouri, novembro 2000.

[Kus97] N. Kushmerick. *Wrapper Induction for Information Extraction*. Phd. Thesis. University of Washington, 1997.

[KMA+98] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, P.J. Modi, I. Muslea, A. G. Philpot e S. Tejada. *Modeling Web Sources for Information Integration.* Proceedings of

the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, páginas 211-218, Madison, Wisconsin, USA, julho 1998.

[LB99] H. W. Lie, B. Bos. *XQuery 1.0: Cascading Style Sheets, level 1*. W3C Recommendation, dezembro 1996, revisada janeiro 1999; http://www.w3.org/TR/REC-CSS1

[LM00] K. Lerman, S. Milton. *Learning the Common Structure of Data*. Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, páginas 609-614, Austin, Texas, USA, julho 2000.

[LS99] O. Lassila, R. R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation, fevereiro 1999; http://www.w3.org/TR/1999/REC-rdf-syntax-19990222

[LS01] E. Lempinen, H. Saarikoski. *DEL – Data Extraction Language*, W3C Note, outubro 2001; http://www.w3.org/TR/2001/NOTE-data-extraction-20011031

[MMK99] I. Muslea, S. Minton, C. Knoblock. *A hierarchical approach to wrapper induction*. Proceedings of the Third International Conference on Autonomous Agents, páginas 190-197, Seattle, Washington, USA, 1999.

[Mor91] W. I. Morris. *American Heritage Dictionary*. Houghton Miffin, 1991.

[PAD98] G. Pringle, L. Allison, D. L. Dowe. *What is a tall poppy among Web pages?* Computer Networks and ISDN Systems, volume 30, números 1-7, páginas 369-377, Australia, abril 1998.

[Pem+02] S. Pemberton et al. *XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)*, W3C Recomendation, janeiro 2000, revisada em agosto 2002; http://www.w3.org/TR/xhtml1/

[PHH03] P. F. Patel-Schneider, P. Hayes, I. Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. W3C Candidate Recomendation, agosto 2003; http://www.w3.org/TR/2003/CR-owl-semantics-20030818/

[Pow03] S. Powers. *Practical RDF*. O'Reilly, 2003.

[Rag03] D. Raggett. *HTMLTidy: Website Oficial*; acessado em setembro 2003. http://www.w3.org/People/Raggett/tidy/

[RHJ99] D. Raggett, A. Le Hors, I. Jacobs. *HTML 4.01 Specification*. W3C Recommendation, dezembro 1999; http://www.w3.org/TR/html4

[RLS98] J. Robie, J. Lapp, D. Schach. *XML Query Language (XQL)*. W3C QL'98 Position Paper, dezembro 1998; http://www.w3.org/TandS/QL/QL98/pp/xql.html

[RM96] P. Resnick, J. Miller. *PICS Specifications*. W3C Recommendations, outubro 1996; http://www.w3.org/PICS

[RNLS99] B. Ribeiro-Neto, A. Laender, A. Da Silva. *Extracting Semi-Structured Data Through Examples*. Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, páginas 94-101, Kansas, Missouri, novembro 1999.

[RW80] B. K. Reid, J. H. Walker. *Scribe User's Manual*. Unilogic, Ltd., 3ª edição, 1980.

[SA01] A. Sahuguet e F. Azavant. *Building intelligent Web applications using lightweight wrappers*. Data and Knowledge Engineering, volume 36, número 3, páginas 283-316, 2001

[SAX03] *SAX: Website Oficial*; acessado em agosto 2003. http://www.saxproject.org

[TBM+01] H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, P. V. Biron, A. Malhotra, D. C. Fallside. *XML Schema*, W3C Recomendations, maio 2001; http://www.w3c.org/XML/Schema

[W3C03] *W3C – World Wide Web Consortium: Website Oficial*; acessado em agosto 2003; http://www.w3.org/

[WVV+01]H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann e S. Hübner. *Ontology-Based Integration of Information — A Survey of Existing Approaches*. Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing, páginas 108-117, Seattle, Washington, USA, 2001.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas