UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

MÁRIO LUÍS SCARPARI CITADIN

EXTRAÇÃO SEMI-AUTOMÁTICA DE DADOS NÃO ESTRUTURADOS NA WEB BASEADA EM ALGORITMOS DE SIMILARIDADE PARA ARMAZENAMENTO EM BANCO DE DADOS OBJETO-RELACIONAL

CRICIÚMA, JULHO DE 2010.

MÁRIO LUÍS SCARPARI CITADIN

EXTRAÇÃO SEMI-AUTOMÁTICA DE DADOS NÃO ESTRUTURADOS NA WEB BASEADA EM ALGORITMOS DE SIMILARIDADE PARA ARMAZENAMENTO EM BANCO DE DADOS OBJETO-RELACIONAL

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Professor MSc. Paracelso de Oliveira Caldas

CRICIÚMA, JULHO DE 2010.

MÁRIO LUÍS SCARPARI CITADIN

Extração Semi-Automática de Dados Não Estruturados na Web Baseada em Algoritmos de Similaridade para Armazenamento em Banco de Dados Objeto-Relacional

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Paracelso de Oliveira Caldas (UNESC)
Orientador

Prof. MSc. Paulo João Martins (UNESC)

Prof. MSc. Cristiane Raquel Woszezenki (UNESC)

Dedico este trabalho a toda minha família,
amigos, colegas e professores do curso.

AGRADECIMENTOS

Agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho. Em especial a minha família por me incentivar e dar forças para jamais desistir de meus objetivos e superar os obstáculos.

Ao meu Orientador Paracelso, pela dedicação e tempo disponibilizado, e desta forma, ter direcionado a alcançar meu propósito.

A minha namorada, pelo apoio, paciência e compreensão dedicados a mim.

Aos meus amigos de infância, Pierre Borges Simão e Maurício Julio de Oliveira, pela real amizade que perdura até hoje.

Aos meus colegas de curso e amigos, em especial ao Claudio e o Marcelo, pelos momentos de alegria compartilhados dentro e fora da sala de aula. Também aos colegas de trabalho, que, de várias formas, contribuem para o meu crescimento profissional e intelectual.

Enfim, agradeço a todas as pessoas que passaram na minha vida e contribuíram para que este sonho fosse realizado.

"Difícil é não parar nos momentos difíceis, quando o interior suplica pela criação e o exterior nos abate por seus obstáculos."

Silvia Mota

RESUMO

A crescente evolução e transformação da Web de forma desestruturada desencadeia uma série de dificuldades com relação ao tratamento e uso dos dados contidos neste ambiente. Com o objetivo de facilitar a manipulação destas informações e também recuperá-las de forma eficiente, este trabalho fundamenta e demonstra o funcionamento de um extrator de dados, que utiliza o conceito de *wrappers* para recuperar os dados do ambiente Web e transformá-los em um arquivo XML. Com a necessidade de tratar os dados extraídos e classificá-los de acordo com o que o usuário necessita, o protótipo desenvolvido na linguagem Java utiliza a biblioteca XSTREAM para realizar a leitura do arquivo XML e mapeá-lo para uma classe Java. Para classificar os dados, é utilizado o algoritmo editDistance para realizar a comparação entre as informações obtidas, e gerar um coeficiente de similaridade. Com o objetivo de armazenar um histórico das informações extraídas, estes dados podem ser armazenados, onde utiliza-se o *framework Hibernate* para fazer o mapeamento dos objetos para o Banco de dados. O resultado final deste processo é a obtenção de dados relevantes ao usuário operador do sistema e a possibilidade de recuperar estas informações posteriormente através dos algoritmos de similaridade.

Palavras-chave: Extração de Dados, XML, Algoritmos de Similaridade, Java, Banco de dados**.**

ABSTRACT

The growing development and transformation of the Web in an unstructured manner triggers a series of difficulties related to treatment and use of data in this environment. Aiming to facilitate handling of such information and also retrieve them efficiently, this work establishes and demonstrates the operation of a data extractor, which uses the concept of wrappers to retrieve data from the Web environment and turn them into a XML file. With the need to treat the extracted data and classify them according to what the user needs, the prototype developed in Java using the XStream library for reading the XML file and map it to a Java class. To sort data, editDistance algorithm is used to perform the comparison between the information obtained, and generate a similarity coefficient. Aiming to store a record of information collected, this data can be stored, which was used Hibernate framework to make the mapping of objects to the database. The outcome of this process is to obtain relevant data to the user system operator and the possibility of retrieving that information later through of the similarity algorithms.

Keywords: Data Extraction, XML, Simirity Algorithms, Java, Databases**.**

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interfaces*
BD	Banco de Dados
BSD	*Berkeley Software Distribution*
HQL	*Hibernate Query Language*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
J2EE	*Java 2 Enterprise Edition*
JDBC	*Java Database Connectivity*
LDAP	*Lightweight Directory Access Protocol*
LGPL	*Lesser General Public License*
ODBC	*Open Data Base Connectivity*
OIDs	*Object Identifier*
OJB	*Object-Relational Java Bridge*
OQL	*Object Query Language*
PDF	*Portable Document Format*
SGBD	Sistemas Gerenciadores de Banco de Dados
SGBDOO	Sistemas Gerenciadores de Banco de Dados Orientados a Objetos
SGBDOR	Sistemas Gerenciadores de Banco de Dados Objeto-Relacionais
SGBDR	Sistemas Gerenciadores de Banco de Dados Relacionais
SQL	*Structured Query Language*
SRI	Sistema de Recuperação de Informação
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
W3QL	*WWW Query System*
WWW	*World Wide Web*
XHTML	*eXtensible HyperText Markup Language*
XLST	*eXtensible Stylesheet Language Transformation*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Segundo ABITEBOUL (2000), as informações ou dados eletrônicos podem ser classificados em três tipos: dados não estruturados ou “crus”, como arquivos de textos; dados estruturados, como bancos de dados; e dados semi-estruturados que formam uma classe intermediária entre esses dois tipos.

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

Porém, nota-se atualmente que boa parte dos dados disponíveis para acesso eletrônico não estão mantidos em Banco de Dados. Alguns exemplos são diretórios de arquivos de documentos de uma organização (Atas de reuniões, manuais, entre outros) ou informações acessíveis através da Internet. A justificativa para o fato decorre da própria natureza destes documentos. Dados encontrados na Web apresentam uma organização bastante heterogênea, que pode variar de um texto sem nenhuma formatação até um conjunto de registros bem estruturado. Além disso, o volume destes dados pode ser grande e com muitos relacionamentos. Considerando dados referentes a um curriculum vitae, por exemplo, pode-se ter um pequeno texto informal descrevendo dados pessoais e experiência profissional ou, oposto a isso, um documento organizado em seções e subseções, com referências para dados das empresas e instituições onde a pessoa trabalhou.

Com a evolução da Internet, a World Wide Web (Web) tornou-se um vasto repositório de dados dos mais variados tipos e formatos. Além disso, teve sua estrutura pautada na construção de documentos de textos e imagens, com o simples objetivo de

informar, sem interação com os usuários. Eram informações em texto livre, som e imagem estáticos, desenvolvidos e disponibilizados apenas para acesso (PELLEGRINO et al, 2008).

Entretanto, os dados não estruturados disponíveis na Web são, em geral, difíceis de serem utilizados e manipulados pela maioria dos usuários da Internet. A principal dificuldade deve-se ao fato de que esses dados não podem ser consultados e manipulados através de técnicas de indexação e consulta como as encontradas em ambientes tradicionais de banco de dados. Outra grande dificuldade dos usuários na Web é que a maioria dos sites de recuperação de dados não possuem métodos sofisticados para consultar e manipular as informações solicitadas pelo usuário. Dessa forma, o resultado de uma busca geralmente não traz somente os dados mais relevantes, gerando insatisfação no seu uso, tornando o processo de busca bastante exaustivo.

Nos últimos tempos, a necessidade por ferramentas de pesquisa, filtragem e manipulação de informações relevantes, disponibilizadas em meio eletrônico, tornou-se essencial. Esta necessidade está presente em diversos contextos, desde ambientes empresariais, para a manipulação de dados internos, até instituições de ensino e pesquisa através da troca de produção científica.

Baseado nas dificuldades e problemas demonstrados na forma atual de tratamento dos dados, este trabalho apresenta como proposta o estudo e o desenvolvimento de uma ferramenta para realizar a extração semi-automática de dados não estruturados na web baseado em algoritmos de similaridade para o tratamento e armazenamento em um banco de dados objeto-relacional.

1.1 OBJETIVOS

1.1.1 Objetivos Gerais

Desenvolver e demonstrar um método para extração semi-automática de dados não estruturados na web, utilizando algoritmos de similaridade para o armazenamento em um banco de dados objeto-relacional.

1.1.2 Objetivos Específicos

Os objetivos específicos são:

a) descrever os tipos de dados existentes na Internet;

b) entender os métodos, regras e o funcionamento da extração de dados semi-automática em ambiente Web;

c) identificar as características de algumas ferramentas que realizam a extração de dados estruturados e semi-estruturados;

d) compreender o funcionamento dos algoritmos de comparação baseados na similaridade;

e) entender o funcionamento dos Sistemas Gerenciadores de Banco de Dados Objeto-Relacional;

f) desenvolver um protótipo de uma ferramenta para extração semi-automática de dados não estruturados baseado em algoritmos de similaridade para posterior armazenamento em banco de dados objeto-relacional.

1.2 JUSTIFICATIVA

Existe uma necessidade constante de pesquisar, filtrar e manipular informações disponíveis em diversos formatos irregulares armazenados em meios eletrônicos e convencionais, bem como o uso de ferramentas que possam pesquisar dados não estruturados e devolver respostas mais objetivas e precisas às consultas do usuário, da mesma forma como ocorre com os bancos de dados tradicionais.

Diferentemente dos dados armazenados em bancos de dados relacionais ou orientados a objetos, os dados não estruturados necessitam de uma etapa prévia ao processo de busca da informação, que consiste na recuperação de sua estrutura implícita. Nesse contexto, a World Wide Web (WWW) aparece como o principal veículo para difusão de dados e informação sem uma estrutura previamente definida.

Tendo em vista a dificuldade na recuperação dessas informações que, em geral, são imprecisas, é necessário um método de recuperação de dados mais completo. Neste contexto, os algoritmos baseados na similaridade são uma solução para a extração de dados e informações de forma mais inteligente e precisa em um ambiente Web. Segundo Santos Filho (2001), esta técnica tem sido aplicada na área da Medicina em pesquisas por similaridade entre imagens de diagnóstico (mamografia, tomografia, ressonância magnética, dentre outras).

Esta necessidade tem estimulado a realização de várias pesquisas com o intuito de extrair informações de dados semi-estruturados. (LAENDER et al, 2002), (SILVEIRA; HEUSER, 2001). A ferramenta Debye (LAENDER et al, 2002), (LAENDER; RIBEIRO NETO; SILVA, 2002), por exemplo, funciona de forma interativa, recebendo como entrada uma série de exemplos criados por usuários a partir de uma página de amostra. Obtidos estes

exemplos, o sistema gera padrões que permitem a extração de objetos de páginas similares.

De uma forma genérica, os trabalhos citados recuperam a estrutura implícita do documento, extraem as informações e as convertem em algum formato estruturado de dados para que seja possível o armazenamento e posterior utilização na extração de novas informações.

Tendo como base os problemas e as dificuldades expostas e, ainda analisando a abrangência e a importância do tema, tanto no ambiente acadêmico quanto social, este trabalho apresenta como proposta o estudo e o desenvolvimento de um protótipo para realizar a extração semi-automática de dados não estruturados na web baseado em algoritmos de similaridade para posterior armazenamento em um banco de dados objeto-relacional.

O protótipo da ferramenta proposta é dito semi-automático pois o usuário deve exemplificar uma primeira definição da estrutura do documento. Em dados não estruturados é difícil a utilização de um processo de extração totalmente automático, pelo fato de não existir uma pré-definição da estrutura dos dados. Dessa forma, é necessária a interferência humana para recuperação da estrutura que não está explícita no documento.

Com relação ao ambiente para realização da extração dos dados, a Web foi a escolhida, pois sua estrutura está em constante crescimento, além de possuir um vasto repositório de informações em diversos formatos, sendo mais comuns os dados não estruturados, que serão objeto de estudos neste trabalho.

O posterior armazenamento é importante, uma vez que o usuário pode necessitar de uma informação já extraída. Neste caso, a realização da consulta será mais rápida e precisa. O banco de dados será do tipo Objeto-Relacional, pois armazenará objetos que conterão a

estrutura dos documentos extraídos.

O estudo e o desenvolvimento de uma ferramenta que faça a extração de dados não estruturados em um ambiente Web baseado em algoritmos de similaridade e armazene-os em um Banco de dados é de extrema importância, visto que é crescente uso de dados sem estrutura na Internet, além disso, a recuperação destas informações, atualmente, não são totalmente precisas.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está dividido em sete capítulos relacionados aos métodos, tecnologias e técnicas envolvidas na elaboração do sistema desenvolvido.

Inicialmente é apresentada a estrutura geral do trabalho, com a introdução, os objetivos gerais e específicos, a justificativa da pesquisa e a organização do mesmo.

No capítulo dois é abordado o conceito inerente aos dados no ambiente Web, a apresentação das divisões de classe de dados existentes, como: dados não estruturados, dados semi-estruturados e dados estruturados. Ainda, neste capítulo é abordado o conceito de abstração de dados.

O capítulo três foi indispensável para que trabalho se desenvolvesse de forma eficiente, pois trata da extração de dados e os principais métodos utilizados na extração de dados da Web.

O capítulo quatro trata da recuperação e integração das informações, apresentado algumas técnicas para identificação das informações que serão extraídas, como as funções de similaridade.

Os conceitos de banco de dados objeto-relacional, objetos, classes, atributos, bem como os métodos para armazenamento de objetos em banco de dados relacionais estão descritos no capítulo cinco. São apresentadas também algumas formas e técnicas para fazer o mapeamento de objetos em banco de dados relacionais.

O capítulo seis apresenta alguns trabalhos correlatos, que foram utilizados para estudo e elaboração deste projeto.

O capítulo sete apresenta o sistema desenvolvido, bem como a metodologia, as ferramentas que serviram como base no desenvolvimento, os detalhes, as características do protótipo desenvolvido e os resultados obtidos com este trabalho.

Os demais capítulos apresentam a conclusão e as referências utilizadas na fundamentação do trabalho.

2 DADOS NO AMBIENTE WEB

Um sistema de gerenciamento de banco de dados é uma coleção de dados inter-relacionados e um conjunto de programas para acessar estes dados. A coleção de dados, normalmente chamada de banco de dados, contem informações relevantes a uma organização. O principal objetivo de um SGBD é fornecer uma maneira de recuperar informações de banco de dados que seja tanto conveniente quanto eficiente (PELLEGRINO et al, 2008).

Os sistemas de banco de dados são projetados para gerenciar grandes blocos de informação. O gerenciamento de dados envolve definir estruturas para armazenamento de informações e fornecer mecanismos para as suas manipulações. Além disso, o sistema de banco de dados precisa garantir a segurança das informações armazenadas, apesar das falhas de sistema ou tentativas de acesso não autorizado.

Porém, a maioria dos dados encontrados no ambiente Web não possuem estrutura bem definida, como as existentes em um banco de dados. O próximo Capitulo 2.1 descreve os tipos de dados existentes e suas características.

2.1 CLASSE DE DADOS

Maior parte da informação se movimenta online e contém dados de texto não-estruturados. Até alguns anos, a publicação de dados eletrônicos estava limitada a algumas poucas áreas científicas e técnicas. Agora, isto está se tornando universal. A maioria das pessoas vê estes dados como documentos da Web, mas estes documentos, em vez de serem manualmente compostos, são cada vez mais gerados automaticamente a partir de bancos de dados. Os documentos, portanto, tem alguma regularidade ou alguma estrutura subjacente que pode ser ou não entendida pelo usuário. É possível publicar enormes volumes de dados desse

modo, e existe a preocupação com a recuperação destas informações armazenadas de forma aleatória. De uma perspectiva de um documento, questões como recuperação eficiente, controle de versão, gerenciamento de alterações e métodos sofisticados de consultas de documentos, que foram anteriormente a esfera de ação da tecnologia de banco de dados, são de grande importância. Da perspectiva de um banco de dados, a Web gerou uma enorme demanda por arquiteturas de banco de dados recentemente desenvolvidas, tais como data warehouses e sistemas de mediação para integração de dados, e isto levou ao desenvolvimento de diversos modelos de classes de dados, com linguagens adaptadas a estes modelos (ABITEBOUL; BUNEMAN; SUCIU, 2000).

Abaixo são citadas as principais classes de dados:

2.1.1 Dados Não-Estruturados

Dados não-estruturados são frequentemente explicados como “sem esquema” ou “auto descritivos”, termos que indicam que não há descrição separada do tipo ou estrutura dos dados. Normalmente, quando armazenados ou programados com dados, primeiro descrevemos a estrutura (tipo, esquema) dos dados e então criamos instâncias daquele tipo (ou povoamos) o esquema. Em dados não-estruturados são descritos diretamente os dados, não utilizando qualquer sintaxe ou formatação.

Estes tipo de dados, se armazenados de forma desorganizada, desperdiçam espaço, já que é preciso repetir as descrições para cada item de dados, mas existe a vantagem ao ganhar interoperabilidade, o que é crucial no contexto da Web (HEUSER; DORNELES; NORONHA, 1998).

2.1.2 Dados Semi-Estruturados

Dados semi-estruturados apresentam uma representação estrutural heterogênea, não sendo nem completamente não-estruturados nem estritamente tipados. Dados Web se enquadram na seguinte definição: em alguns casos os dados possuem uma descrição uniforme (um catálogo de produtos), em outros, algum padrão estrutural pode ser identificado (um conjunto de documentos no formato de artigo), ou então, praticamente não existem informações descritivas associadas (um arquivo de imagem) (ABITEBOUL, 1997). Afirma-se também que dados semi-estruturados são dados nos quais o esquema de representação está presente (de forma explícita ou implícita) juntamente com o dado, ou seja, o mesmo é auto-descritivo. Isto significa que uma análise do dado deve ser feita para que a sua estrutura possa ser identificada e extraída (BUNEMAN, 1997).

As características principais de dados semi-estruturados são (FLORESCU; LEVY; MELDELZON, 1998):

a) estrutura evolucionária: a estrutura dos dados modifica-se tão frequentemente quanto os seus valores. Dados Web apresentam este comportamento, uma vez que existe o interesse em manter dados sempre atualizados;

b) estrutura descritiva e não prescritiva: dada a natureza irregular e evolucionária dos dados semi-estruturados, as estruturas de representação implícitas ou explícitas normalmente se restringem a descrever o estado corrente de poucas ocorrências de dados similares. Desta forma, não é possível prescrever esquemas fechados e muitas restrições de integridade com relação à semântica dos atributos. Um sinônimo para estrutura descritiva é estrutura indicativa;

c) distinção entre estrutura e dados não é clara: como a estrutura está embutida na descrição dos dados, muitas vezes a distinção lógica entre estrutura e valor não é clara. Pode-se ter, por exemplo, um endereço representado como um valor atômico em uma ocorrência de dado (string) ou como um tipo definido pelo usuário (com atributos rua, número e complemento) em outra ocorrência. Esta característica torna mais complicado o projeto de um BD para tais dados.

As características de dados semi-estruturados diferem bastante das características de dados mantidos em banco de dados tradicionais, como os modelos relacionais. BDs tradicionais apresentam um esquema predefinido e uma estrutura homogênea a nível de atributos e tipos. Em se tratando de dados semi-estruturados, cada ocorrência de dado pode ser heterogênea nos dois aspectos mencionados anteriormente. Dada essa heterogeneidade, em geral a estrutura de um dado semi-estruturado está presente na própria descrição do dado, necessitando ser identificada e extraída. Estas tarefas são complexas, uma vez que a distinção entre esquema e dados nem sempre é clara, se compararmos ocorrências de dados semanticamente iguais (FLORESCU; LEVY; MENDELZON, 1998).

Como cada dado pode ter uma organização própria, uma estrutura de representação para um conjunto de dados tende a ser extensa, de forma a refletir as particularidades de cada um deles. Já em um banco de dados tradicional, todas as ocorrências de um mesmo tipo de dado estão descritas uma única vez em um estrutura independente, mais estável e mais reduzida. Como existe regularidade de representação, é mais fácil prescrever restrições de integridade semânticas aplicáveis a todas as ocorrências de dados. ABITEBOUL et al (1997) apresenta outras características importantes dos dados semi-estruturados:

a) definição à *posteriori*: esquemas para dados semi-estruturados são usualmente definidos após a existência dos dados, com base em uma investigação de suas estruturas particulares e da análise de similaridades e diferenças. Isto não significa que sempre existe um esquema associado a um dado semi-estruturado;

b) estrutura irregular: coleções extensas de dados semanticamente similares estão organizados de maneiras diferentes, podendo algumas ocorrências terem informações incompletas ou adicionais em relação a outras. Em suma, não existe um esquema padrão para esses dados. O exemplo do *curriculum vitae* se enquadra nesta característica;

c) estrutura implícita: muitas vezes existe uma estrutura básica para os dados, porém, essa estrutura está implícita na forma como os dados são apresentados. É necessário realizar uma computação para obter essa estrutura;

d) estrutura parcial: apenas parte dos dados disponíveis pode ter alguma estrutura, seja implícita ou explícita. Por exemplo, componentes de objetos que são arquivos bitmaps não-estruturados. Já dados pessoais podem ter uma estrutura básica implícita ou explícita. Como consequência, um esquema para estes dados nem sempre é completo do ponto de vista semântico e nem sempre todas as informações esperadas estão presentes;

e) estrutura extensa: a ordem de magnitude de uma estrutura para estes dados é grande, uma vez que os mesmos são muito heterogêneos. Supondo diferentes formatos para um *curriculum vitae,* uma união de atributos significativos em cada formato pode produzir um esquema extenso.

Outra característica própria de dados semi-estruturados é que neste tipo de dado normalmente a noção de esquema é criada a *posteriori*, após a existência dos dados, com base em uma investigação de suas estruturas particulares e da análise de similaridades e diferenças. Em bancos de dados tradicionais, utiliza-se a noção de esquema a *priori*, o qual é definido antes do armazenamento de qualquer dado no banco de dados.

2.1.3 Dados Estruturados

Entende-se por dados estruturados, aqueles que são armazenados em estruturas bem definidas (esquemas), como é o caso dos bancos de dados relacionais, onde um esquema composto por nomes de tabelas e atributos é especificado e armazenado logo na criação de qualquer banco de dados (STEPHEN, 1999).

Esta estrutura rígida está evoluindo para os chamados dados semi-estruturados (documentos XML), a fim de permitir uma representação mais natural de objetos complexos do mundo real, tais como livros, artigos, filmes, componentes de aviões, projetos de chips, entre outros, sem que o projetista da aplicação em questão tenha que se contorcer para poder encontrar uma representação adequada para os dados (STEPHEN, 1999).

De forma resumida, esta classe de dados possui um esquema fixo, ou seja, possui uma representação onde é possível identificar seus atributos e valores, sem qualquer reestruturação.

2.2 ABSTRAÇÃO DE DADOS

Para que o sistema seja funcional, ele precisa recuperar dados de maneira eficiente. A necessidade de eficiência tem levado projetistas a usar estruturas de dados

complexas para representar dados no banco de dados. Como muitos usuários de sistemas de banco de dados não são treinados em computador, os desenvolvedores ocultam a complexidade dos usuários sob vários níveis de abstração conforme (DATE, 2000):

a) nível físico: o nível de abstração mais baixo descreve como os dados são realmente armazenados. O nível físico descreve em detalhes estruturas complexas de baixo nível;

b) nível lógico: o próximo nível de abstração mais alto descreve que dados estão armazenados no banco de dados e que relações existem entre eles. O nível Lógico portanto, descreve o banco de dados inteiro em termos de um pequeno número de estruturas relativamente simples. Embora a implementação das estruturas simples no nível lógico possa envolver estruturas em nível físico complexas, o usuário do nível lógico não precisa estar consciente desta complexidade;

c) nível de view: o nível de abstração mais alto descreve apenas parte do banco de dados. Mesmo que o nível lógico use estruturas mais simples, a complexidade permanece devido a variedade de informações armazenadas em um grande banco de dados. Muitos usuários do sistema de banco de dados não precisam de toda essa informação, em vês disso, eles precisam acessar apenas uma parte do banco de dados. O nível de view existe para simplificar sua interação com o sistema. O sistema pode fornecer muitas visões para o mesmo banco de dados.

2.3 MODELAGEM DE DADOS

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

A forma mais usual de modelar dados não estruturados e semi-estruturados é através de grafos direcionados rotulados, onde os vértices representam objetos identificáveis e as arestas são arcos para outros objetos que fazem parte da sua estrutura, que é hierárquica. Um rótulo presente em um arco indica o nome do atributo do objeto de onde o arco parte (objeto origem), podendo ainda informar o tipo de dado do objeto alcançável através do arco (objeto destino). Como cada ocorrência de dado pode ter uma estrutura diferente, não deve haver restrição no número de arcos que partem de um objeto origem. Todo objeto tem um vértice raiz que é o ponto de partida para a investigação da sua estrutura hierárquica (BUNEMAN 1997).

O modelo de grafo é apropriado para representar dados não estruturados e semi-estruturados pois a estrutura particular implícita de cada dado já está embutida no próprio grafo. Neste sentido, os arcos têm uma importância fundamental, uma vez que nomeiam os atributos e tipos que fazem parte da estrutura do dado (SUCIU, 1997).

A Figura 1 ilustra um exemplo de um objeto semi-estruturado representado no grafo direcionado rotulado (MELLO, 2000):

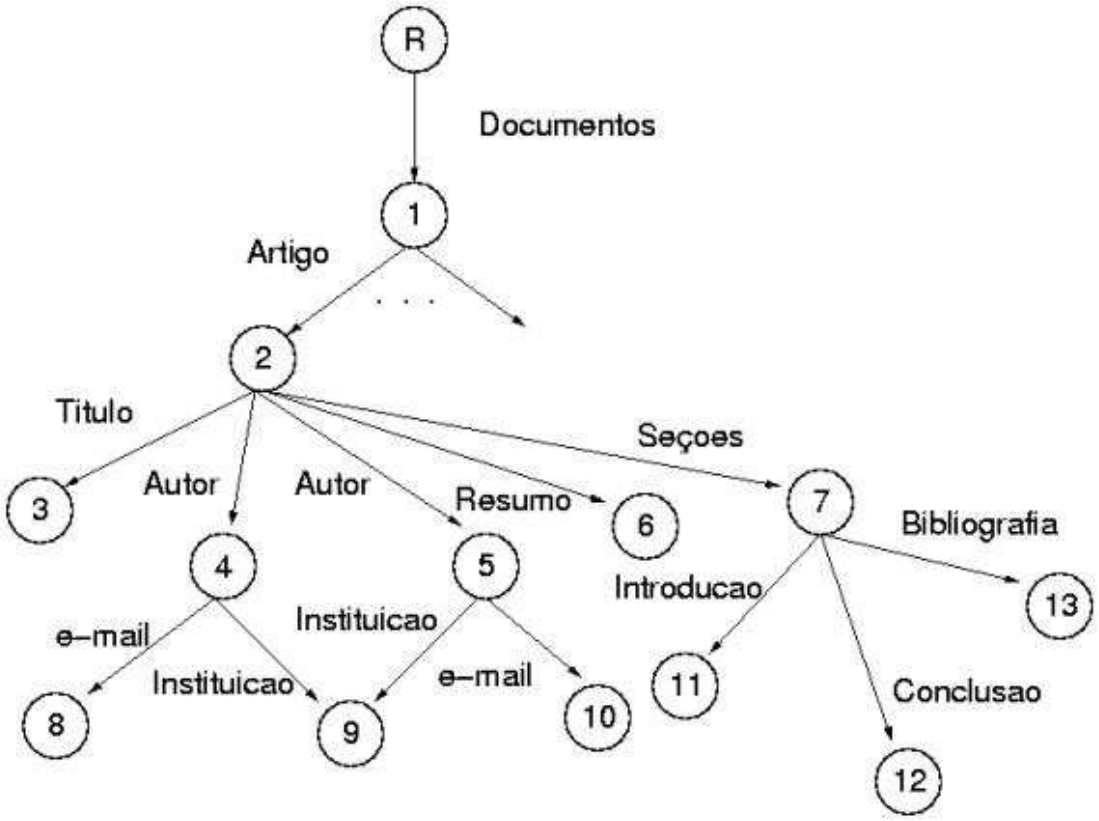


Figura 1. Grafo Direcionado Rotulado.

Fonte: PIMENTA (2003).

Segundo MELLO (2000), este é um modelo flexível, pois não considera a existência de um esquema fixo para um conjunto de dados. Toda informação esquemática está incluída nos rótulos, que podem mudar dinamicamente. É considerado, também, um modelo para instâncias de dados semi-estruturados, pois representa valores de dados e associa rótulos a cada valor para descrever o seu significado.

Após a análise e modelagem dos dados através de um grafo rotulado é possível configurar e parametrizar a extração dos dados.

3 EXTRAÇÃO DE DADOS

O interesse em extrair dados da Web, e armazená-los, tem aumentado significativamente devido a ineficiência das ferramentas de consulta disponíveis, como *browsers* e *search engines*. Segundo HAMMER et al, (1997) alguns mecanismos de extração de dados vêm sendo pesquisados e desenvolvidos para atender aos seguintes objetivos:

a) extrair informações mais precisas e em número razoavelmente menor;

b) possibilitar que os resultados sejam armazenados de uma forma mais estruturada;

c) facilitar o processamento de consultas posterior ao processo de extração;

d) fornecer resultados que estejam de acordo com a intenção do usuário.

No gerenciamento de dados não estruturados, a extração de dados faz parte do processamento de uma consulta, além de outras etapas executadas, como *parsing* e otimização (BUNEMAN, 1997). A etapa de extração é responsável pela recuperação de informações contidas nas fontes de dados, porém, sem retornar os documentos completos que contenham estas informações. Apenas as informações solicitadas pelo usuário ou relevantes para o processamento da consulta devem ser extraídas, como ocorre em consultas aos bancos de dados convencionais.

Um processo de extração normalmente transforma dados não estruturados de uma fonte para dados adequados a um modelo de dados, seja ele estruturado ou semi-estruturado. Este resultado é posteriormente processado de alguma forma: no caso de um dado estruturado, por exemplo, o resultado pode ser já armazenado em um BD para fins de consulta; no caso de

um conjunto de dados semi-estruturados ou não estruturados, por exemplo, pode ser necessário um processo de modelagem e estruturação que verifica se cada ocorrência deste conjunto é uma resposta válida para a consulta.

Mediadores e *wrappers* são importantes neste contexto, uma vez que atuam como módulos intermediários entre aplicações e fontes de dados, sendo responsáveis pelas transformações dos dados. Mediadores são módulos que integram conhecimento presente em fontes heterogêneas de dados, produzindo informação útil para camadas superiores de um sistema de gerenciamento de dados (WIEDERHOLD, 1992). Um serviço de mediação apresenta uma regra para transformação de dados e contém estruturas de conhecimento que guiam estas transformações. Uma ampla gama de funcionalidades pode estar associada a um mediador, como reorganização lógica de dados, aplicação de métodos para obtenção de informações derivadas e aplicação de padrões de apresentação de dados.

Os *wrappers* têm a função de encapsular o acesso a uma fonte de dados, tornando-a utilizável de uma maneira mais conveniente que a sua representação original. Os *wrappers* possuem características diferentes de um mediador. Um mediador integra dados de várias fontes de dados, enquanto um *wrapper* provê uma interface simplificada para uma única fonte de dados ou mais de uma fonte que tenham uma interface comum de acesso. Um *wrapper* pode ainda adicionar funcionalidade a uma fonte de dados, como por exemplo, prover uma camada de acesso para a um conjunto de documentos, para facilitar a manipulação dos dados (THIRAN, 2001).

Neste caso, o *wrapper* atua como um extrator de dados, identificando e adequando valores de dados com o esquema presente na aplicação.

Sendo a web vista como um grande banco de dados, semelhante a um gráfico, é normal que as pesquisas realizadas vão além do paradigma básico de busca das informações atualmente suportadas pelos mecanismos de pesquisas hoje existentes. Deve-se, entretanto, levar em consideração a estrutura implícita das páginas, bem como a estrutura externa das ligações que as interconectam.

As primeiras ferramentas de pesquisa na Internet baseavam-se na busca por palavras ou frases presentes em documentos descobertos pelos *web crawlers*. Atualmente vários estudos e pesquisas são realizadas para superar este paradigma, utilizando largamente as estruturas de ligações da web nas consultas, a fim de melhorar o desempenho de busca na rede.

Devido à presença maciça dos dados, chamados de não estruturados, existe a necessidade de modificar ou remodelar as linguagens tradicionais, como a SQL, pois estas não são dirigidas a estes tipos de dados. Entretanto, várias linguagens estão sendo propostas para consultas sobre estes dados. Algumas são extensões da linguagem OQL (em bases de dados orientados a objetos), como a LOREL (ABITEBOUL et al, 1997) e a StruQL (FERNANDEZ et al, 1997), que utilizam gráficos rotulados como um modelo de dados flexíveis, enfatizando a característica de consultar o esquema dos dados, tratando suas irregularidades, como por exemplo, a falta ou repetição de campos e registros heterogêneos.

Outras propostas abordam linguagens de consulta na Internet, como a W3QL e a WebSQL, que podem combinar condições de padrões de texto que aparecem no conteúdo das páginas com os padrões de gráficos que descrevem as estruturas de ligações das páginas. A WebSQL tem como proposta modelar a web como sendo um banco de dados relacional,

composto de duas relações virtuais. A primeira relação apresenta uma tupla para cada documento na web, e a segunda, apresenta uma tupla para cada âncora em cada documento da Web (MENDELZON; MILO, 1997).

3.1 INTEGRAÇÃO DE INFORMAÇÕES

A web possui um número crescente de fontes de informações que podem ser vistas como grande conjuntos de tuplas, que estão contidas em páginas HTML ou implícitas atrás de interfaces de formulários. Programas chamados *wrappers* podem ser escritos, dando a impressão que o web site está servindo a conjuntos de tuplas. Chama-se web source a combinação entre o site e o *wrapper* a ele associado.

Os sistemas de integração de dados na web tem por objetivo principal responder consultas que requeiram extração e combinação de dados a partir de múltiplas fontes dispersas na web. Vários são os problemas detectados na construção destes sistemas, sendo semelhantes aos encontrados em banco de dados heterogêneos, como grande número de web sources em constantes evoluções, escassez de metadados sobre aspectos das fontes e um maior grau de autonomias destas fontes (EIKVIL, 1999).

Na construção destes sistemas, deve-se assumir duas abordagens distintas: A primeira refere-se ao fato de que os dados são movidos para um único local e as consultas realizadas sobre eles, sendo que estes devem ser constantemente atualizados, obtendo, entretanto, consultas mais velozes. Na segunda, os dados permanecem em seus locais originais, sendo as consultas realizadas localmente, garantindo a atualização dos dados, embora diminuindo a velocidade. Esta abordagem é mais apropriada devido ao número de fontes que são numerosas e autônomas, exigindo com isso métodos sofisticados para

otimização e execução das consultas.

Há duas características que diferem estes sistemas dos banco de dados tradicionais:

Primeiramente, os dados não são obtidos diretamente, mas sim através do uso de wrappers. Estes têm a tarefa de traduzir os dados do web site para uma forma que possa ser processada pelo sistema externo que realiza a pesquisa. Segundo, o usuário não especifica pesquisas de acordo com o esquema, segundo o qual os dados estão armazenados. Ao invés disso, são utilizados esquemas mediadores que são específicos para cada aplicação de integração, evitando assim, que o usuário tenha que conhecer o esquema de cada uma das fontes. O sistema deve possuir informações sobre as fontes, uma vez que tem que traduzir as pesquisas do usuário para os esquemas das fontes, precisando com isso de dados como por exemplo, conteúdos, atributos, restrições e capacidades de processamento de pesquisas. (FREITAG, 1998).

Outra informação necessária diz respeito a otimização das pesquisas, onde há escassez de dados estatísticos referentes aos que se encontram nas fontes, resultando pouca informação para avaliação de custos e planos de execução das consultas. Há ainda a questão da construção dos *wrappers*. Conforme descrito, sua função é extrair dados de um site para permitir que estes possam ser manipulados pelo sistema que irá integrar os dados. O problema é que esta extração é normalmente realizada sobre páginas HTML, onde os dados estão em linguagem natural ou sob a forma de gráficos. Para contornar tal questão, deve-se utilizar uma técnica que prevê o desenvolvimento de gramáticas especializadas, para especificar como os dados devem ser colocados nas páginas. Outra técnica seria o desenvolvimento de

mecanismos de aprendizagem indutiva para *wrappers* de aprendizagem automática. Um algoritmo seria usado para gerar uma gramática de extração de dados para as páginas subseqüentes, a partir de páginas cujos dados são rotulados e então, usados como exemplo (ALLEN, 1995).

A extração de dados não estruturados, normalmente, possui o suporte de modelos conceituais ou ontologias (WESSMAN; LIDDLE; EMBLEY, 2005), para que o processo funcione. Neste caso, torna-se indispensável que a ontologia, ou o modelo conceitual, seja projetado de forma bastante completa para que, de fato, o processo seja executado de forma funcional.

A extração de informação sobre textos não estruturados geralmente baseia-se em técnicas de linguagem natural, que usa a relação sintática entre as palavras para extrair o significado de uma expressão. O texto semi-estruturado, por sua vez, possui marcadores que buscam explicitar e identificar estruturas no documento. Contudo, textos desse tipo apresentam irregularidades como campos ausentes ou com valor nulo, variações na ordem dos dados e ausência de delimitadores entre as informações a serem extraídas (APPELT, 1999).

Existem duas estratégias para construção de *wrappers* que irão extrair informações destes textos. A primeira é baseada na engenharia de conhecimento e outra na geração automática de regras (APPELT, 1999).

Regras dos sistemas baseados em engenharia de conhecimento podem ser definidas através de regras de produção. As regras de produção são utilizadas predominantemente para representar um conhecimento heurístico, especificando um conjunto de ações que devem ser realizadas para uma dada situação. Uma regra de produção é

composta por uma parte de condições e uma parte de ações. A ação da regra somente será executada se a condição for verdadeira. As regras de produção permitem uma representação modular do conhecimento, ou seja, cada regra representa uma parte do conhecimento de forma independente. Além disso, são fáceis de compreender e de modificar. Novas regras podem ser inseridas, enriquecendo assim a base de conhecimento do sistema (SODERLAND, 1999).

Já nos sistemas com geração automática de regras, diversos autores aplicam algoritmos de aprendizagem de máquina baseados em casamento de padrões (CALIFF; MOONEY, 1999) ou baseados em autômatos finitos, como Freitag (1998). O uso das Cadeias Escondidas de Markov também têm apresentado bons resultados (RAY; CRAVEN, 2001). Essas técnicas utilizam uma base de treinamento para identificar padrões e gerar regras de forma automática. Outras técnicas de aprendizagem de máquina como aprendizado simbólico, métodos estatísticos, gramática indutiva e programação lógica indutiva também têm sido aplicadas no problema de extração de informação (EIKVIL, 1999).

A etapa de extração dos dados do ambiente web permite que os dados extraídos sejam recuperados posteriormente por meio de algumas técnicas. No Capitulo 4 estão descritas os principais métodos para recuperação de informações e suas características.

4 RECUPERAÇÃO DE INFORMAÇÃO

A quantidade de documentos utilizados em uma empresa aumenta conforme a demanda e isto aumenta a dificuldade em recuperar a informação correta dentro do contexto de busca desejado pelos usuários. Um dos motivos desta dificuldade é a inexistência de sistemas capazes de realizar esta tarefa. Dessa forma, é de extrema importância que a área de Recuperação de Informação (*Information Retrieval*) seja aprimorada e desenvolvida para realizar as tarefas básicas e necessárias dentro do contexto de recuperação de dados.

Segundo Baeza (1999), recuperação de informação é a área da computação que trabalha na tarefa de encontrar itens de informação relevantes para uma determinada necessidade de informação expressa pela requisição de um usuário (consulta) e disponibilizá-los de uma forma adequada ao propósito da busca de informação por ele submetida. Neste cenário, os sistemas de recuperação de informação (SRI) visam dar acesso às informações potencialmente contidas em documentos registrados, organizados e processados, a fim de garantir a eficácia do processo de busca e maximizar o uso.

A recuperação de informação é dividida em quatro etapas, que podem ser resumidas como: Representar cada documento em uma forma que possa ser “compreendido” pelo computador; Interpretar as consultas fornecidas; Comparar as consultas interpretadas com o conjunto de documentos indexados; e, Apresentar os resultados de forma adequada ao propósito do usuário.

A recuperação da informação possui algumas técnicas para alcançar seus

objetivos, que são citadas abaixo:

4.1 INDEXAÇÃO

A indexação é o processo pelo qual as palavras (ou termos) que compõem um documento são extraídas e armazenadas em uma base de dados, formando uma estrutura denominada índice. Como a indexação de um documento pode resultar em uma grande relação de termos, periodicamente, um usuário especialista pode fazer a manutenção dos termos incluídos nos índices para que estes se resumam apenas às palavras-chaves que identificam o documento ou o assunto nele tratado. A avaliação dos termos indexados é importante para que o processo de recuperação seja mais preciso e retorne, de forma mais ágil, apenas resultados que se enquadram no escopo desejado. Conforme Wives (1997) depois de gerado, o índice pode ser usado em mais de uma consulta.

O conceito de busca indexada consiste em submeter consultas às bases de documentos cujo conteúdo tenha sido previamente processado e indexado, gerando estruturas de índices que determinam a relevância de determinadas palavras no contexto do documento. Neste cenário, os índices gerados funcionam como filtros com a função de selecionar os objetos que pertencem ao contexto geral da busca submetida. Para controlar os termos que serão utilizados durante o processo de indexação é aconselhável a criação de uma listagem de *stopwords*, que são termos sem relevância potencial ao documento indexado. Esta listagem pode ser definida com um número inicial de termos e, durante o processo de análise dos documentos, sofrer alterações com a inclusão de novas *stopwords*, aumentando a abrangência da lista (BAEZA, 1999).

4.2 PRECISÃO

A busca de informações de interesse particular, seja na web ou em um banco de dados específico, não é um objetivo fácil de ser alcançado. Para amenizar esta dificuldade, a tecnologia de busca semântica vem fornecendo, atualmente, algumas soluções para que o acesso à informação se torne cada vez mais satisfatório.

As medidas mais comuns para avaliar a efetividade de um Sistema de Recuperação de Informação são a rechamada e precisão (*Recall* e *Precision*). A rechamada ou revocação é definido como a proporção dos elementos relevantes recuperados em relação a todos os elementos relevantes na fonte de informação, enquanto que precisão é a proporção entre os elementos relevantes recuperados em relação a todos os elementos recuperados. Ambas estão baseadas nas avaliações de relevância do usuário que seguem o processo de recuperação. *Recall* mede a integridade da saída, isto é, a habilidade do sistema para recuperar informações relevantes. A técnica de Precisão mede a relevância de saída, significado dos dados contidos em uma página. Tal limitação dificulta e até inviabiliza o tratamento mais efetivo e a utilização de forma inteligível das informações pelos motores de busca (BAEZA, 1999).

Baeza (1999) cita, também, que precisão e rechamada são medidas baseadas na noção de documentos relevantes de acordo com uma determinada necessidade de informação. A rechamada é utilizada para medir a habilidade do sistema de encontrar todos os documentos relevantes, já a precisão mede a habilidade de recuperar documentos que são em sua maioria relevantes.

Existem outras formas de medir a eficiência de um Sistema de Recuperação de

Informação, mas geralmente estas medidas são mais difíceis de serem interpretadas, além de exigirem informações que não são obtidas sem uma análise mais detalhada dos documentos, o que demanda um esforço computacional maior. Devido a estes fatores, sugere-se a utilização da abrangência e a precisão por serem mais facilmente interpretadas (WIVES, 1997).

4.3 TESAUROS

Tesauro é uma lista estruturada de termos associada e empregada por analistas de informação e indexadores, para descrever um documento com a desejada especificidade, em nível de entrada, e para permitir aos pesquisadores a recuperação da informação que procura. Em resumo, um Tesauro é uma coleção de termos selecionados, que tem como objetivos a indexação e a recuperação de documentos e dados numa determinada área do conhecimento (CAVALCANTI, 1978).

Buscadores como o Google teriam grandes dificuldades para implementar seu mecanismo de busca baseados em estruturas de Tesauros. Isto se dá pelo fato de que existe uma grande variação de contexto entre a concepção de domínio corporativo de empresa para empresa, bem como de pessoa para pessoa. Por exemplo, o termo “bola” pode ter um significado para “Esportes” e outro significado para “Geometria”.

Neste sentido, um modelo de Tesauro pré-definido por terceiros – no caso, os mantenedores dos mecanismos de buscas disponíveis na web – tende a não ser adaptável, reconhecido ou estar completamente distante da realidade de relacionamento de termos da empresa ou pessoa, considerando que cada indivíduo ou corporação tem a sua idéia sobre o seu negócio ou processo de trabalho (WIVES, 1997).

4.4 FUNÇÕES DE SIMILARIDADE

As funções de similaridade permitem ao usuário à recuperação de informações de forma mais dinâmica, uma vez que o motor de busca implementado procurará por cadeias de caracteres que representam a mesma idéia contida nos termos submetidos à pesquisa. As funções de similaridade têm sido amplamente utilizadas pela área de Recuperação de Informações para gerar um ranking contendo todos os objetos armazenados, ou seja, uma lista de objetos ordenados de acordo com o seu escore em relação ao objeto consultado.

O resultado das funções de similaridade é uma pontuação, normalmente entre zero e um (0 e 1), que indica o valor de similaridade entre o objeto consultado e um elemento ou documento armazenado em algum lugar, como em um Banco de Dados. O valor 0 representa que não há relação entre os objetos comparados enquanto o valor 1 representa a igualdade.

Para limitar a quantidade de resultados retornados pelas consultas por similaridade define-se um coeficiente mínimo aceitável (*threshold*) que é responsável por separar as instâncias que, efetivamente, são relevantes à consulta dos demais resultados. Este valor implica diretamente na qualidade dos resultados retornados, visto que, caso seja definido um limiar muito baixo, podem ser retornados elementos que não correspondem ao objeto consultado. Ao contrário, podem ser retornados falsos negativos, ou seja, a consulta não retorna elementos relevantes. Uma solução adotada por diversos estudos é solicitar que o usuário especifique o valor de *threshold* a ser utilizado em suas consultas (FAGUNDES; KROTH, S.d.).

4.4.1 Algoritmos De Comparação Baseados Na Similaridade

A maioria dos Sistemas de Gerenciamento de Bases de Dados (SGBD) disponíveis atualmente foram desenvolvidos para manipular dados em domínios numéricos ou de pequenas seqüências de caracteres, aproveitando a relação de ordem total que existe entre os elementos desses domínios de dados. Entretanto, o volume de dados armazenado e manipulado nos SGBD cresce constantemente e, incluindo-se outros tipos de dados mais complexos, os quais não possuem a relação de ordem total. Para esses tipos de dados mais complexos tanto as operações de busca quanto as estruturas de indexação existentes nos SGBD Relacionais (SGBDR) têm pouca utilidade. Para domínios de dados complexos, como por exemplo imagens, áudio, vídeo, dados espaciais e temporais, entre outros, o grau de similaridade é o fator mais importante (FALOUTSOS, 1997), fazendo surgir uma nova classe de operadores que melhor se adequam para manipular esses dados, chamados de operadores por similaridade. Para que eles possam ser empregados, o domínio de dados em questão deve estar associado a uma função de dissimilaridade, também chamada de função de distância ou métrica (CHÁVEZ, 2001) .

Existem basicamente dois operadores de seleção por similaridade para domínios métricos: os operadores de consulta por abrangência (*Range Query*) e consulta por vizinhos mais próximos (*k-Nearest Neighbor Query*) (KORN et al, 1996). Uma consulta por abrangência recupera objetos armazenados que diferem-se no máximo até um dado grau de similaridade do objeto central da consulta. Uma consulta por vizinhos mais próximos recupera os k objetos armazenados que sejam mais semelhantes ao objeto central da consulta, onde k é um valor inteiro que especifica a quantidade de objetos.

O algoritmo utilizado no protótipo desenvolvido é o editDistance, suas características e funcionamento serão abortados no Capitulo 4.4.1.1.

4.4.1.1 Algoritmo EditDistance

O algoritmo para cálculo de distância chamado editDistance compara duas cadeias de caracteres, e determina o número mínimo de operações necessárias para transformar uma cadeia na outra. As operações realizadas nesse cálculo são: inserção, remoção e substituição de um caractere. A partir do resultado das operações é possível calcular-se um valor de similaridade entre as cadeias de caracteres.

O algoritmo para calcular a distância entre uma cadeia de caracteres A e uma cadeia de caracteres B, respectivamente com i e j caracteres, utilizando apenas as operações, é apresentado na figura 2.

Inicialmente, a matriz M é criada no passo 2 e tem a sua primeira linha e primeira coluna inicializada no passo 3. Os valores associados às demais células são calculados pelo mínimo entre três valores, que representam o custo das três operações. Os valores calculados pelos passos 4(a) e 4(b) são responsáveis por computar os custos das operações de inclusão e remoção de caracteres, enquanto que o valor calculado pelo passo 4(c) computa o custo da operação de substituição de caracteres. O cálculo dessa última operação é auxiliado pela função sub(x, y) que retorna 0, se x = y, e 1, caso contrário.

1. Sejam A e B duas cadeias de caracteres tais que |A|= i e |B|= j
2. Criar uma matriz M com $i + 1$ linhas e $j + 1$ colunas
3. Inicializar a primeira linha e a primeira coluna
   (a) Primeira linha $M[0, q] = q$, para $0 \leq q \leq j$
   (b) Primeira coluna. $M[p, 0] = p$, para $1 \leq p \leq i$
4. Calcular o valor associado às demais células $M[p, q]$ da matriz através do mínimo entre:
   (a) A posição acima mais um. $(M[p - 1, q] + 1)$
   (b) A posição à esquerda mais um. $(M[p, q - 1] + 1)$
   (c) A posição na diagonal esquerda superior mais o custo de substituição do caractere $A[p]$ por $B[q]$
   $(M[p - 1, q - 1] + sub(A[p], B[q]))$

Figura 2. Passos para cálculo da similaridade. Fonte: VINSON, 2007.

Ao final da execução do passo 4, a matriz está completamente preenchida. Cada célula da matriz M[x, y] armazena o número mínimo de operações necessárias para a seqüência dos x primeiros caracteres de A transformar-se na sequência dos y primeiros caracteres de B, ou vice-versa. Portanto, o valor contido na célula M[i, j], reflete o número mínimo de operações necessárias para a cadeia A transformar-se na cadeia B.

Após obter o número mínimo de operações é necessário calcular a medida da similaridade. A fórmula a seguir (figura 3) calcula a similaridade final pela normalização do número de operações que não são necessárias para efetuar a transformação (máximo possível menos o mínimo necessário) no intervalo [0, 1]. A normalização é feita com divisão do número de operações que não são necessárias pelo máximo número de operações possíveis.

$$editDistance(A, B) = \frac{Max(i, j) - M[i, j]}{Max(i, j)}$$

Figura 3. Fórmula para cálculo da distância. Fonte: VINSON, 2007.

A utilização de um algoritmo de similaridade permite ao usuário encontrar os dados que são relevantes às suas necessidades. Tendo em vista este ponto e encontrados os dados que são relevantes, é importante armazená-los de forma que possam ser recuperados em outro momento. Para isso, é necessário possuir um SGBD que tenha capacidade de armazenar dados complexos. Neste escopo, para armazenamento de dados não estruturados os SGBDOR são indicados, pois utilizam os benefícios dos bancos tradicionais e os benefícios da orientação a objetos.

5 SISTEMAS GERENCIADORES DE BANCO DE DADOS OBJETO-RELACIONAL.

A maior parte dos sistemas de banco de dados desenvolvidos até hoje foram os relacionais (SGBDR). Porém, com o surgimento de sistemas mais avançados como: banco de dados hipertexto, banco de dados multimídia, sistemas de automação, entre outros, houve a necessidade de assimilar estruturas que suportassem estes sistemas. Desta forma, o padrão para bancos de dados se lançou para a tecnologia objeto relacional e orientados a objetos, onde, os SGBDOR oferecem suporte aos tradicionais modelos relacionais com estrutura de orientação à objetos e os SGBDOO se caracterizam principalmente pela herança existente entre as entidades. Atualmente, não é possível a existência de um só padrão de mercado pela natureza das aplicações que ambos estão destinados (BOND et al, 2005).

Com o surgimento dos SGBDOO, acreditava-se que este seria o substituto do modelo relacional, uma vez que, em orientação a objetos se tinha uma maneira mais natural para se especificar qualquer objeto do mundo real.

Essa impressão começou a ser modificada quando se viu que os SGBDOO, por serem diferentes dos SGBDR, estavam muito aquém dos Bancos Relacionais em características importantes, tais como: performance, segurança, facilidades de consultas, capacidade de armazenamento, entre outras.

Para as organizações acostumadas com todas as facilidades e potencialidades dos grandes bancos relacionais essa mudança brusca de paradigma não foi nada agradável, já que teriam que começar tudo de novo.

Neste contexto é que surge o paradigma objeto-relacional, que veio com a proposta de suprir as carências do Modelo Relacional, proporcionando aos usuários toda a naturalidade para modelar objetos complexos e suas características peculiares que os banco de dados orientados a objetos propunham, e ao mesmo tempo, não abrindo mão de toda a carga tecnológica adquirida para os Bancos Relacionais em anos de desenvolvimento (HOLMES; SCHILDT, 2003).

A proposta é oferecer ao cliente uma interface orientada a objetos de forma transparente para que esse possa modelar tudo, usando o paradigma Relacional.

Os SGBDOR além de, continuarem oferecendo uma interface relacional para as aplicações que foram desenvolvidas neste paradigma, também dispensam a necessidade de dois bancos de dados, um para aplicações relacionais e outro para aplicações orientadas a objetos.

5.1 OBJETOS

Conforme Meyer et al (1996) os objetos podem ser qualquer coisa do mundo real que faça parte do contexto do sistema. Um objeto pode ser uma pessoa, um carro, um departamento de uma empresa, uma conta bancária, um parágrafo em um texto, entre outros. Os objetos facilitam a compreensão do mundo real e oferecem uma base real para a implementação em computador. É muito mais fácil para as pessoas trabalharem com sistemas se estes estiverem mais próximos de nossa visão do mundo real. É mais fácil entender o que a propriedade cor de um objeto carro significa, do que uma variável com nomenclatura obscura em um bloco do sistema. Os objetos são entidades que encapsulam informação de estado ou dados e têm um conjunto de operações associadas que manipulam

estes dados.

Cada objeto do sistema é único. Isto define a identidade de objetos, que se distingue pela sua existência, e não pelos valores que suas propriedades possam assumir (MEYER, et al, 1996).

5.2 CLASSES

As classes segundo Meyer et al (1996) são responsáveis pela utilização de abstração em orientação a objetos. Para simplificar a criação de objetos em um sistema, hierarquias de classes são utilizadas. As classes de objetos descrevem as propriedades, métodos e relacionamentos de um grupo de objetos similares. Quando objetos são agrupados em conjuntos são denominados Classes.

5.3 ATRIBUTOS

Os atributos segundo Rumbaugh (1994) são os dados, características de um objeto. Os atributos guardam os valores do objeto. O conjunto de atributos de um objeto define o seu estado.

Cada atributo possui um valor para cada instância de objeto. Por exemplo, o atributo Nome tem o valor "Pedro" no objeto Pedro, e "José" no objeto José. Os nomes que identificam atributos (ex:Nome, Data_Nascimento) são únicos dentro de uma classe. Classes diferentes podem conter nomes de atributos iguais. Por exemplo, a classe Pessoa e a classe Empresa podem ter o mesmo atributo chamado "endereço".

Os bancos de dados objeto relacionais são capazes de suportar grande parte dos conceitos de orientação a objetos descritos anteriormente. Diferente dos

SGBDOO, os SGBDOR são desenvolvidos com base em SGBDR, desta forma, conforme Rumbaugh (1998) é possível unir conceitos de Orientação a Objetos como: definição de super tabelas; supertipos; herança; reutilização de códigos de criação e encapsulamento, com algumas características de controle dos SGBDOO: controle de identidade de objetos, referência a objetos, etc. E ainda características dos SGBDR como a capacidade de consulta avançada e a alta proteção a dados, devido a natureza declarativa da SQL.

5.4 ARMAZENAMENTO DE OBJETOS EM BANCO DE DADOS.

5.4.1 Mapeando Objetos em Banco de Dados Relacionais

A adoção das metodologias de desenvolvimento Orientadas a Objetos como um padrão de mercado levou a uma mudança radical na estruturação e organização de informação. Contudo, a utilização de bancos de dados relacionais ainda é uma prática muito comum. Devido à necessidade de se trabalhar com estas bases de dados relacionais para o armazenamento persistente de dados, é comum a adaptação dos modelos de objetos na tentativa de compatibilizá-los com o modelo relacional. Ainda assim, é notório o esforço aplicado no processo de persistência manual dos objetos no banco de dados – o que força os desenvolvedores de aplicações a ter que dominar a linguagem SQL e utilizá-la para realizar acessos ao banco de dados. Estas questões levam a uma redução considerável na qualidade do produto final, construção de uma modelagem "orientada a objetos" inconsistente e a um desperdício considerável de tempo na implementação manual da persistência (HOLMES; SCHILDT, 2003).

Mesmo com o trabalho manual, não é possível ignorar a força e confiabilidade dos SGBDR, pois, após um grande período de desenvolvimento e ajustes fazem dos SGBDR a

opção mais eficiente.

Para realizar um processo de mapeamento entre sistemas que utiliza objetos e SGBDR, são propostas algumas idéias que convergiram para o conceito de Camada de Persistência.

5.4.2 Evolução Dos Modelos De Persistência De Objetos

Em todo o processo de desenvolvimento de aplicações Orientadas a Objetos, nem sempre é dada a devida atenção à forma como a persistência de objetos será administrada e implantada. Na maioria dos sistemas, incluir código SQL para acesso ao SGBD em meio ao restante da lógica do sistema é a solução adotada, graças à rapidez de implementação. Esta solução não é a ideal: sua adoção implica, muitas vezes, no acoplamento do sistema ao SGBD utilizado, o que dificulta o processo de manutenção de código.

Além disso, todas as mudanças nas tabelas existentes no Banco de Dados dificultam o gerenciamento da aplicação. O código SQL codificado na aplicação, muitas vezes, tem que ser reescrito, recompilado e testado novamente.

Para diminuir este acoplamento, existe uma segunda opção: separar o código SQL das classes da aplicação, de forma que as alterações no modelo requerem modificações apenas nas classes de acesso a dados, minimizando o impacto das alterações no sistema como um todo. Esta forma de desenvolvimento, trás um controle maior quanto ao escopo dos possíveis erros gerados por mudanças no esquema de dados do sistema. Entretanto, apesar da melhor divisão de responsabilidades trazidas pela adoção das Casses de Dados, a solução ainda não é a ideal, pois mantém os objetos e dados relacionais intimamente ligados (BOND et al, 2005).

A característica principal de uma camada de acesso a dados é garantir a independência entre o esquema de dados do banco e o modelo de objetos, permitindo, desta forma, que a banco ou detalhes do esquema de dados sejam substituídos sem muito impacto na aplicação. Além disso, uma Camada de Persistência permite o armazenamento dos dados em outros tipos de bases de dados diferentes dos relacionais, incluindo os SGBDOO, SGBDOR e arquivos XML, por exemplo (HOLMES; SCHILDT, 2003).

5.4.3 Camadas De Persistência

Camada de Persistência de Objetos é uma biblioteca que permite a realização do processo de persistência de forma transparente. Graças à independência entre a camada de persistência e o repositório utilizado, também é possível gerenciar a persistência de um modelo de objetos em diversos tipos de repositórios. A utilização deste conceito permite ao desenvolvedor trabalhar como se estivesse em um sistema completamente orientado a objetos – utilizando métodos para incluir, alterar e remover objetos e uma linguagem de consulta para SGBD Orientados a Objetos (BOND et al, 2005).

O uso de uma Camada de Persistência no desenvolvimento de aplicações gera vantagens evidentes: a sua utilização isola os acessos realizados diretamente ao banco de dados na aplicação, centraliza os processos de construção de consultas e operações de manipulação de dados em uma camada de objetos inacessível ao programador. Esta divisão de responsabilidades garante maior confiabilidade às aplicações e permite que, em alguns casos, o próprio SGBD ou a estrutura de suas tabelas possam ser modificados, sem trazer impacto à aplicação nem forçar a revisão e recompilação de códigos (HOLMES; SCHILDT, 2003).

5.4.3.1 Requisitos de uma Camada de Persistência

Conforme Holmes e Schildt (2003) uma Camada de Persistência real deve implementar as seguintes características:

a) dar suporte a diversos tipos de mecanismos de persistência: um mecanismo de persistência pode ser definido como a estrutura que armazenará os dados – seja ela um SGBD relacional, um arquivo XML ou um SGBDOO, por exemplo. Uma Camada de Persistência deve suportar a substituição deste mecanismo livremente e permitir a gravação de estado de objetos em qualquer um destes meios;

b) encapsulamento completo da camada de dados: o usuário do sistema de persistência de dados deve utilizar-se, no máximo, de mensagens de alto nível como *save* ou *delete* para lidar com a persistência dos objetos, deixando o tratamento destas mensagens para a camada de persistência em si;

c) ações com multi-objetos: Suportar listas de objetos sendo instanciadas e retornadas da base de dados deve ser um item comum para qualquer implementação, tendo em vista a freqüência desta situação;

d) transações: ao utilizar-se da Camada de Persistência, o programador deve ser capaz de controlar o fluxo da transação ou ter garantias sobre o mesmo;

e) extensibilidade: A Camada de Persistência deve permitir a adição de novas classes ao esquema e a modificação fácil do mecanismo de persistência;

f) identificadores de objetos: A implementação de algoritmos de geração de chaves de identificação garante que a aplicação trabalhará com objetos com identidade única e sincronizada entre o banco de dados e a aplicação;

g) cursores e *proxies*: As implementações de serviços de persistência devem ter ciência de que, em muitos casos, os objetos armazenados são muito grandes – e recuperá-los por completo a cada consulta não é uma boa idéia. Técnicas como o *lazy loading* (carregamento tardio) utilizam-se dos *proxies* para garantir que atributos só serão carregados à medida que forem importantes para o cliente e do conceito de cursores para manter registro da posição dos objetos no banco de dados (e em suas tabelas específicas);

h) registros: apesar da idéia de trabalhar-se apenas com objetos, as camadas de persistência devem, no geral, dispor de um mecanismo de recuperação de registros - conjuntos de colunas não encapsuladas na forma de objetos, como resultado de suas consultas. Isto permite integrar as camadas de persistências a mecanismos de geração de relatórios que não trabalham com objetos, por exemplo, além de permitir a recuperação de atributos de diversos objetos relacionados com uma só consulta;

i) diversas versões de banco de dados e fabricantes: a Camada de Persistência deve tratar de reconhecer diferenças de recursos, sintaxe e outras particularidades existentes no acesso aos bancos de dados suportados, isolando isto do usuário do mecanismo e garantindo portabilidade entre plataformas;

j) múltiplas conexões: um gerenciamento de conexões (pool de conexões) é uma técnica que garante que vários usuários utilizarão o sistema simultaneamente sem quedas de desempenho;

k) consultas SQL: Apesar do poder trazido pela abstração em objetos, este

mecanismo não é funcional em todos os casos. Para os casos extremos, a Camada de Persistência deve prover um mecanismo de consultas que permita o acesso direto aos dados;

l) controle de concorrência: acesso concorrente a dados pode levar a inconsistências. Para prever e evitar problemas decorrentes do acesso simultâneo, a Camada de Persistência deve prover algum tipo de mecanismo de controle de acesso.

5.4.3.2 Identificadores dos Objetos (*Object Ids*)

Os *Object Ids* (OIDs) são identificadores únicos de elementos do esquema de dados. Eles funcionam como um tipo de chave primária nos sistemas Orientados a Objetos, mantendo relacionamentos entre objetos e garantindo a unicidade dos objetos em todo o esquema (BOND et al, 2005).

Para que identificação única seja garantida, também nos esquemas de dados relacionais, as Camadas de Persistência adotam o conceito de OIDs, de forma que as chaves primárias, identificadoras das linhas de cada tabela, são geradas mediante um algoritmo de geração de chaves, que garante a unicidade dos elementos na aplicação e no banco de dados.

Os níveis de unicidade admitidos para elementos em um esquema de dados podem variar. No paradigma relacional puro, é comum identificar objetos unicamente através de uma chave primária, única dentro de cada tabela de dados. O nível de identificação de objetos dos OIDs pode ir além: para gerar identificadores que garantem uma unicidade real, é comum utilizar-se de números de identificação únicos entre todas as tabelas do esquema de dados ou mesmo únicos em relação a quaisquer bases de dados. Esta preocupação em garantir a

identidade de registros de banco de dados é justificada quando têm-se em mente cenários de bancos de dados distribuídos ou dados sincronizados entre bases distintas, por exemplo. O OID é o identificador que garante a identidade do elemento entre os bancos de dados distintos.

Alguns mecanismos mais conhecidos são utilizados para garantir a geração de OIDs independente da base. Estes mecanismos são comumente oferecidos pelas Camadas de Persistência para permitir a geração automática de chaves. São eles (HOLMES; SCHILDT, 2003):

a) utilizar, em uma tabela, o operador SQL *Max()* com o objetivo de obter o maior valor utilizado como identificador do objeto e utilizá-lo, acrescido de um, como sendo a próxima chave para um objeto armazenado. Esta técnica garante unicidade a nível de tabela;

b) adotar uma tabela de valores-chave, na qual o sistema mantém o último valor utilizado como chave por qualquer elemento no banco de dados. Sempre que um novo elemento é inserido, este valor deve ser acrescido de um. Esta técnica permite a unicidade ao nível do banco de dados;

c) utilizar o algoritmo *High/Low* – semelhante ao modelo de tabelas de valores-chave. Neste caso, o sistema acessa a tabela que armazena valores-chave apenas uma vez, mantém o valor encontrado (*high*) na memória e cria identificadores para os objetos acoplado a um contador auxiliar (*low*) gerado em memória – no momento de fornecer à aplicação um valor de chave único os dois valores são concatenados, para inclusão no banco de dados. Esta técnica também garante

unicidade ao nível do banco de dados, e o *overhead* conseqüente da geração de chaves é o menor dentre as três técnicas explicitadas, já que o banco de dados é raramente acessado para gerar valores de chave.

5.4.4 Mapeando Atributos

Ao realizar a transferência de um objeto para uma tabela em um SGBDR, os atributos do objeto são mapeados em colunas da tabela. Alguns fatores como os tipos dos dados e o comprimento máximo dos campos devem levar em consideração no processo de mapeamento, já que os SGBD podem se comportar de maneira diferente. Outro fator relevante é que, atributos de um objeto não possuem obrigatoriamente uma coluna em uma tabela que os referencie. Como exemplo, pode-se citar o valor total de uma compra: este dado poderia ser armazenado para fins de consulta, mas mantê-lo no banco de dados não é uma idéia interessante, pois este valor pode ser obtido através de consultas. Além destes fatores, existem casos em que um atributo pode ser mapeado para diversas colunas, como exemplo: endereços completos, nome dividido em “nome” e “sobrenome” no banco de dados. Da mesma forma, vários atributos podem ser mapeados para uma mesma coluna, como: número de telefone, por exemplo. As implementações de Camadas de Persistência provêem, em alguns casos, suporte a este tipo de situação (FERNANDEZ, 1997).

5.4.5 Mapeamento de Classes em Tabelas

O processo de mapeamento de classes em tabelas de um banco de dados relacional nem sempre é um processo simples. As principais técnicas para mapeamento de objetos mais comumente implementadas são detalhadas a seguir. É notória a adoção de uma destas técnicas, mesmo quando nenhum tipo de mecanismo de persistência automático é adotado no

desenvolvimento.

5.4.5.1 Mapeamento por Hierarquia

Neste tipo de mapeamento, a hierarquia de classes é representada por uma mesma tabela no banco de dados. A principal desvantagem desta estratégia é a ausência de normalização dos dados, ferindo as regras da teoria de modelagem de dados. Além disso, para hierarquias de classes com muitas especializações, a proliferação de campos com valores nulos na maioria das linhas da tabela se torna também um grande problema.

5.4.5.2 Mapeamento por Classe Concreta

No mapeamento por classe concreta, uma tabela no banco de dados representa uma classe presente no sistema. A tabela é um espelho da classe e de todos os elementos contidos na mesma. Esta técnica leva à redundância de dados, pois, quaisquer atributos definidos em uma classe abstrata devem ser criados nas tabelas que representam classes-filhas da mesma. Ainda, generalizar ou especializar um objeto pode tornar-se um problema, já que é preciso transferir os seus dados de uma tabela para outra quando acontecer uma atualização.

5.4.5.3 Mapeamento por Classe

Nesta estratégia, cria-se uma tabela para cada classe da hierarquia, sendo que o relacionamento entre as tabelas é feito pelas chaves estrangeiras. Esta modalidade de mapeamento, procura-se manter a normalização de dados, de forma que, no final, a estrutura das tabelas fique parecida com a hierarquia das classes representada pelos diagramas de classe UML. A identificação de uma classe na sua classe pai, com a utilização das chaves estrangeiras, permite obter o tipo de um objeto armazenado nas tabelas do sistema sem

precisar utilizar junções entre as tabelas, melhorando a performance, e é uma estratégia muito utilizada. Esta técnica realiza de forma mais natural o mapeamento dos objetos para bancos de dados relacionais.

5.4.6 Mapeamento de Relacionamentos

Os relacionamentos entre objetos são uma das características mais facilmente mapeadas. Conceitualmente, existem apenas três tipos de relacionamentos possíveis (um para um, um para muitos e muitos para muitos.

Relacionamentos um para um necessitam que uma chave estrangeira seja colocada em uma das duas tabelas, realizando o relacionamento com o elemento associado na outra tabela. Dependendo da disposição desta chave estrangeira, pode-se navegar no relacionamento, o qual, se dá sempre da tabela que possui a chave estrangeira para a tabela referenciada. Para os relacionamentos um para muitos, a mesma técnica pode ser adotada: uma referência na forma de chave estrangeira deve ser posta na tabela que contém os objetos múltiplos. No caso de relacionamentos muitos para muitos, convenciona-se criar uma tabela intermediária que armazene pares de chaves, identificando os dois lados do relacionamento.

5.4.7 Camadas de Persistência e Linguagens de Programação

Na Web estão disponíveis diversas implementações de camadas de persistência. Desenvolvidas geralmente nas linguagens Java, Smalltalk e C++, estes *frameworks* tratam da automatização para geração dos esquemas de dados e podem até efetuar engenharia reversa, criando um conjunto de classes a partir de tabelas em um banco de dados. As Camadas de Persistência implementam vários recursos que serão apresentados abaixo, sendo que geralmente trabalham com apenas um esquema de mapeamento de classes para tabelas,

estratégias de geração de identificadores, suporte a quaisquer tipos de relacionamentos e geração de código SQL automatizada.

Na linguagem Java, são encontradas algumas das bibliotecas de persistência (BOND et al, 2005):

a) *Hibernate*: uma implementação que permite a persistência de objetos em bases de dados utilizando o mapeamento de classes para XML e JDBC. Trata-se de um serviço de persistência e recuperação de objetos, já que, ao contrário dos frameworks de persistência, não é necessário estender nenhuma classe especial para que um objeto possa ser armazenado. Projetado para permitir integração com ambientes J2EE, o *Hibernate* utiliza reflexão para tratar a persistência, gerando código SQL à medida que for necessário. Atualmente ele é compatível com vários SGBDs comerciais (PostgreSQL, Oracle, DB2, MySQL, Sybase, HypersonicSQL, SAP DB, Progress, Microsoft SQL Server, Mckoi SQL, Pointbase e Interbase), o *Hibernate* é distribuído segundo a licença LGPL e suporta uma API baseada no padrão ODMG 3.0 para construção de SGBDs Orientados a Objetos. Dentre outros recursos avançados, o *Hibernate* suporta gerenciamento remoto através da API JMX e possibilita a geração de esquemas de dados para representar hierarquias de classes;

b) Castor: um framework de ligação de dados (*databinding*), o Castor propõe-se a ser a menor distância entre objetos Java, documentos XML, diretórios LDAP e dados SQL, promovendo mapeamentos entre todas estas estruturas de representação de objetos. A API do pacote Castor específica para a persistência em

bancos de dados relacionais é a JDO – uma implementação inspirada no padrão Java Data Objects da Sun. A API provê integração com ambientes J2EE. Atualmente em sua versão 0.9, o Castor suporta os SGBDs Oracle, Sybase, SQL Server, DB2, PostgreSQL, Informix, InstantDB, Hypersonic SQL, Interbase, MySQL e SAP DB. A distribuição segue a licença LGPL;

c) *Object-Relational Java Bridge* (OJB): um projeto do grupo Apache para prover uma implementação open-source dos padrões de mapeamento de objetos ODMG e JDO, o OJB possibilita a manipulação de objetos sem ser necessário a implementação de uma interface ou estender alguma classe específica. A biblioteca provê suporte a aplicações cliente-servidor ou locais, de forma que é possível utilizar a API OJB para persistência de objetos, também, em ambientes J2EE. Além disso, este projeto possui integração com o sistema de geração de logs Log4j. Os SGBDs suportados pela implementação atual incluem PostgreSQL, DB2, Hypersonic SQL, Sybase, Informix, MS-SQL Server, MySQL, MS-Access, Oracle e SAP DB. A distribuição é feita segundo a licença Apache;

d) Torque – um framework de persistência desenvolvido como subprojeto do projeto Apache Turbine, a API gera toda a estrutura de banco de dados, classes e código SQL para acesso aos dados relativos a um esquema pré-configurado. O esquema é escrito na forma de um arquivo XML, que é interpretado pela biblioteca utilizando o Ant, uma ferramenta de compilação de código do projeto Apache. A API Torque é distribuída segundo a licença Apache.

Apesar da resistência de alguns desenvolvedores em adotar esquemas de

persistência, a sua utilização minimiza o tempo investido na implementação de um sistema e eleva a qualidade do produto final, à medida que diminui a possibilidade de erros de codificação. Além de fornecer um acesso mais natural aos dados, as camadas de persistência executam controle transacional, otimização de consultas e transformação automática de dados entre formatos distintos (BOND et al, 2005).

Desta forma, as Camadas de Persistência funcionam como a principal ponte de ligação entre sistemas Orientados a Objetos e repositórios de dados diversos: um conceito poderoso, com implementações estáveis e comprovadamente eficientes.

6 TRABALHOS CORRELATOS

A Web cresce a cada dia, e com ela também cresce de forma exponencial a busca por respostas rápidas, aprimoradas, que retornem resultados não só quantitativos, mas também qualitativos. Neste âmbito, visto que a web é um grande banco de dados, os quais possuem pouca ou nenhuma estrutura, é crescente também a procura de soluções, não só para extração de dados na Web, mas também para posterior armazenamento e estruturação dos dados. Abaixo estão descritas algumas soluções para a extração de dados da Web que condizem com o trabalho proposto.

6.1 EXTRAÇÃO AUTOMÁTICA DE CONTEÚDO SEMI-ESTRUTURADO NA WEB

Este trabalho descreve a implementação de um sistema de extração automática de informação semi-estruturada na Web orientada a domínio. O sistema utiliza regras de produção baseada em objetos que produzem instâncias de classes que representam o domínio considerado. O sistema faz uso da API JEOPS, um motor de inferência de primeira ordem com encadeamento progressivo integrado à linguagem Java. Como estudo de caso, foi definido classes que representam o Campeonato Brasileiro de Futebol. O sistema recebe como entrada o endereço eletrônico de um portal Web e, fazendo uso de fatos e regras de sua base de conhecimento relacionada ao Campeonato Brasileiro, identifica links relacionados e navega no portal a fim de localizar a tabela de classificação do campeonato e extrair dados da tabela, produzindo de forma automática instâncias das classes especificadas (MELLO; MACEDO, 2009).

6.2 EXTRAÇÃO SEMÂNTICA DE DADOS SEMI-ESTRUTURADOS ATRAVÉS DE EXEMPLOS E FERRAMENTAS VISUAIS.

Este trabalho apresenta como proposta o projeto de uma ferramenta para realizar a extração semântica e semi-automática de dados semi-estruturados. O usuário especifica, através de uma interface visual, um exemplo de estrutura hierárquica do documento e de seu relacionamento com os conceitos de ontologia, gerando uma gramática descritiva da estrutura implícita do mesmo. A partir desta gramática, a ferramenta realiza a extração dos próximos documentos de forma automática, reestruturando o resultado em um formato regular de dados, neste caso, XML (*eXtensible Markup Language*) (SILVEIRA, 2001).

6.3 UM PROCESSO AUTOMÁTICO PARA EXTRAÇÃO DE METADADOS DE DOCUMENTOS PDF USANDO UM TEMPLATE XML

Este trabalho descreve uma proposta para extração de metadados de documentos PDF. O extrator desenvolvido utiliza um modelo *(template),* representado por um arquivo XML, que possui a descrição dos metadados a serem extraídos do documento desejado. A partir do XML, o extrator executa uma série de passos para a localização dos metadados no documento indicado,sendo todo o processo realizado de forma automática, onde a única intervenção por parte do usuário é a construção do *template* XML (MANICA;CERVI;GALANTE, 2008).

6.4 DEBYE

A ferramenta Debye (*Data Extraction by Example*) faz parte de um projeto desenvolvido pela Universidade Federal de Minas Gerais. A extração realizada pela ferramenta Debye é feita em duas etapas: a especificação dos objetos que servem de modelo, feita pelo usuário, e a extração propriamente dita.

Para especificar alguns objetos de exemplo e descrever a estrutura implícita do

documento, o usuário interage através de uma interface gráfica escrita na linguagem Java (LAENDER; SILVA; ALTIGRAN, 1999).

7 SISTEMA DE EXTRAÇÃO, CONSULTA E ARMAZENAGEM DE DADOS NÃO ESTRUTURADOS

Tendo em vista os aspectos demonstrados no trabalho, constata-se que a Web é um grande repositório de dados, que normalmente são livres ou não estruturados, como: textos, e-mails, imagens entre outras mídias de diversos formatos. Desta forma, torna difícil a busca de informações precisas neste ambiente. O trabalho desenvolvido vem de encontro a esta deficiência e propõe uma melhoria na forma de tratar dados não estruturados.

O protótipo desenvolvido é dito semi-automático, pois se tratando de dados não estruturados, existe uma complexidade em realizar a extração de dados de forma automática, uma vez que os dados são completamente livres de formatação e sem estrutura definida. Desta forma, a solução encontrada para superar esta barreira, é fazer com que o usuário especifique a estrutura do documento que deseja extrair.

O objetivo principal do protótipo desenvolvido é extrair estas informações do ambiente Web, através de um extrator de dados, e por meio da aplicação de um algoritmo de similaridade no documento extraído, encontrar a informação desejada.

Alcançado o objetivo supracitado, o sistema reterá, modelará a informação encontrada a armazenará em um Banco de Dados. Com o propósito de recuperar esta informação em outro momento.

7.1 METODOLOGIA

Para alcançar o objetivo e desenvolver o protótipo, foram seguidas algumas etapas primordiais, tais como: Compreender o funcionamento das ferramentas de extração de Dados,

observar o funcionamento das ferramentas existentes de extração de dados semi-estruturados, entender o funcionamento dos algoritmos de similaridade, estudar um algoritmo específico, testar bibliotecas que disponibilizam o uso dos algoritmos, entender a linguagem XML, bem como expressões regulares.

7.2 FERRAMENTA PARA EXTRAÇÃO DE DADOS

A idéia por trás de um processo de extração de dados da Web consiste em usar conteúdos não estruturados, disponíveis sob a forma de páginas HTML, de forma a construir grandes repositórios de informação. Isto pode ser feito com base na conversão dos dados disponíveis Web em registros estruturados, através de wrappers. Este processo também é denominado por Web Scraping ou Web Data Mining.

Os dados na Web encontram-se normalmente misturados com instruções de formatação. De fato, as páginas HTML são desenhadas para ser interpretadas por humanos e não por processos automáticos de tratamento de informação. No entanto, uma vez que cada página Web está construída de acordo com alguma lógica subjacente, é possível descrever um processo de extração de dados que permita recuperar a informação, separando-a das instruções de formatação.

A extração de dados foi o primeiro passo para implementação do protótipo desenvolvido, uma vez que, através destes dados extraídos, serão aplicadas as demais técnicas afim de alcançar o objetivo.

Foram realizadas pesquisas a cerca de sistemas e ferramentas que tem por objetivo a extração de dados no ambiente Web. Abaixo serão descritas as principais características do projeto Web Harvest, um sistema de código aberto e que disponibiliza uma API para a

realização da extração de dados, a qual foi utilizada no desenvolvimento do protótipo.

7.2.1 Web Harvest (http://web-harvest.sourceforge.net/)

Web Harvest é uma ferramenta de extração de dados na Web, *Open Source*, desenvolvida em Java. A ferramenta oferece um caminho para coletar dados de uma página Web e extrair as informações desejadas. Para isso, usa tecnologias para manipulação de Texto e XML como: XSLT, XQuery, XPath e Expressões regulares. Web Harvest mantêm o foco principalmente nos dados HTML e XML, pois são os tipos de dados mais encontrados na Web.

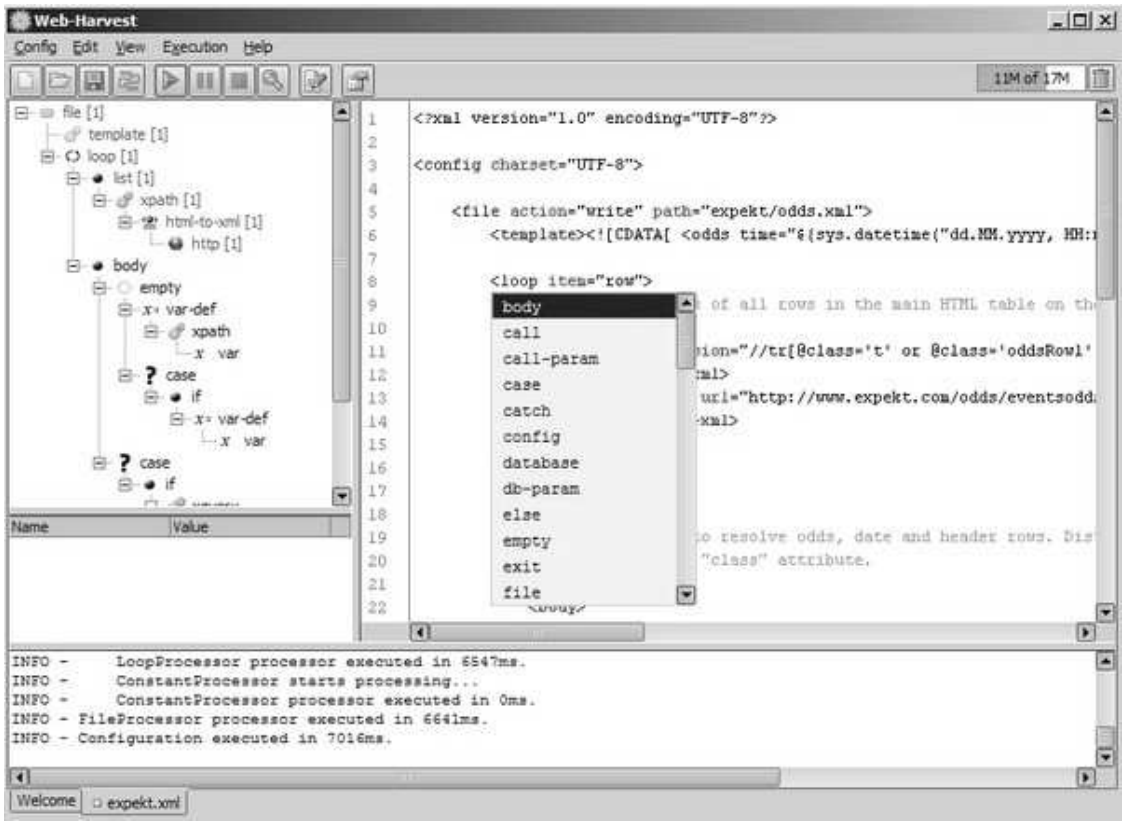


Figura 4. Interface do projeto Web Harvest.

Nesta ferramenta, cada processo de extração é definido por um arquivo de configuração XML. Este arquivo descreve uma sequência de processos de extração, os quais executam tarefas específicas de forma a obter um dado objetivo. Estes processos são executados como um *pipeline*. A saída de um processo é usada como entrada do processo seguinte (um processo faz a extração dos links de uma dada página web, enquanto que o processo seguinte baixa os conteúdos a partir desses URLs).

A figura 5 ilustra um exemplo de arquivo XML, utilizado para configuração da extração de dados no Web Harvest:

```
<xpath expression="//a[@shape='rect']/@href">
    <html-to-xml>
        <http url="http://www.somesite.com/"/>
    </html-to-xml>
</xpath>
```

Figura 5. Exemplo de arquivo de configuração.

Quando o Web Harvest executa esta configuração, acontecem as seguintes etapas:

a) a tag <HTTP> processa o *download* o conteúdo do site especificado;

b) a tag <HTML-TO-XML> “limpa” o código HTML da página e transforma em XHTML;

c) <XPATH> faz uma varredura na página XHTML buscando links, baseado na expressão regular informada no atributo *expression*.

Esta ferramenta também suporta um conjunto de funcionalidades, tais como: manipulação de variáveis, estruturas condicionais, laços de repetição, processamento de dados HTML e XML, funções e operações com arquivos.

O Projeto Web Harvest também disponibiliza uma completa API para que outros projetos de extração de dados utilizem suas funcionalidades, sendo este o fator principal para a escolha desta ferramenta ser utilizada no protótipo.

7.3 PROTÓTIPO DESENVOLVIDO

O protótipo desenvolvido neste trabalho, foi implementado na linguagem Java, por diversos fatores, entre eles:

a) plataforma independente: usada em vários sistemas operacionais;

b) linguagem orientada a objetos;

c) robusto: controle de alocação de memória automático e um seguro controle de exceções;

d) gratuidade;

e) grande procura em pesquisas e no mercado de trabalho.

A explanação sobre o sistema desenvolvido será dividida em 3 capítulos: Extrator de dados, Calculo da Similaridade e o Armazenamento das Informações. Estes capítulos descrevem as principais etapas no desenvolvimento do projeto.

7.3.1 Extrator de Dados

Esta etapa é responsável pela recuperação de informações contidas nas diversas fontes de dados, onde, neste trabalho, foi escolhido o ambiente Web como sendo o repositório de informações. Os principais motivos para a escolha do ambiente Web foram: o vasto repositório de dados não estruturados que ele possui, e pela alta velocidade que cresce este

ambiente. Diante destes fatores, existe uma grande necessidade de se possuir melhor organização dos dados ali encontrados.

Contudo, a recuperação de informações não deve retornar documentos completos, como o código HTML de uma página por exemplo. Ela deve ser capaz de retornar apenas as informações solicitadas pelo usuário ou relevantes para o processamento da consulta, como ocorre em consultas aos bancos de dados convencionais.

O processo de extração de dados, neste projeto, utiliza a API disponibilizada pelo projeto open source Web Harvest, e será responsável por transformar os dados não estruturados ou sem formatação, de forma que se adéquem a um modelo de dados, também chamado de espelho de dados. Esta modelagem será definida em um arquivo de configuração XML, que será utilizado para que o extrator de dados tenha definido quais as etapas, quais dados ele deve recuperar e como será a modelagem dos dados.

7.3.1.1 Arquivo de Configuração

Para realização da extração dos dados é necessário informar, antecipadamente, os parâmetros da consulta, onde será realizada a extração, quais tipos de dados serão recuperados, a estrutura do arquivo recuperado, entre outras informações. Neste item, será demonstrado um exemplo de configuração e a descrição dos métodos contidos neste arquivo. Para criar o arquivo de configuração é necessário possuir, previamente, o conhecimento da linguagem XML e do ambiente de onde serão extraídas as informações, como analisar o código-fonte HTML da página Web.

O arquivo de configuração deve possuir extensão XML, e conter as sequências de processos que o extrator deve seguir. Cada método especificado no arquivo é responsável por

uma etapa. Abaixo são demonstrados alguns elementos que podem ser usados no arquivo de configuração:

a) text – conversão e representação textual do conteúdo;

b) file - entrada/saída de arquivos;

c) http – envio de pedidos HTTP e download dos conteúdos;

d) html-to-xml - limpeza de conteúdos HTML e conversão para XHTML;

e) regexp - pesquisas e substituições com base em expressões regulares;

f) xpath - pesquisas sobre conteúdos XML com base em XPath;

g) xquery - pesquisas sobre conteúdos XML com base em XQuery;

h) xslt - transformações XSLT sobre conteúdos XML;

i) script - aplicação de scripts simples "Java" com lógica específica.

Para exemplificar a utilização de alguns elementos citados, a figura 6 mostra um exemplo de arquivo de configuração. O arquivo demonstrado tem o objetivo de extrair as notícias contidas em uma página do site Terra (http://www.terra.com.br). O resultado da extração será um novo arquivo que contém as notícias, distinguindo o título da notícia, a data da notícia e o conteúdo da notícia.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<config charset="ISO-8859-1">
    <var-def name="inicioUrl">http://noticias.terra.com.br/ciencia/ultimasnoticias/0,,EI238,00.html</var-def>

    <file action="write" path="terra/terra${sys.date()}.xml" charset="UTF-8">
        <template>
            <![CDATA[ <terra_noticias data_extracao="${sys.datetime("dd.MM.yyyy")}"> ]]>
        </template>
        <loop item="noticiaUrl" index="i">
            <!-- Coleciona as URLs de todas as notícias encontradas na página -->
            <list>
                <xpath expression="//div[@class='list articles']/ol/li/a/@href">
                    <html-to-xml>
                        <http url="${inicioUrl}"/>
                    </html-to-xml>
                </xpath>
            </list>
            <!-- Faz o Download de cada artigo e extrai os dados solicitados -->
            <body>
                <xquery>
                    <xq-param name="doc">
                        <html-to-xml>
                            <http url="${noticiaUrl}"/>
                        </html-to-xml>
                    </xq-param>
                    <xq-expression><![CDATA[
                        declare variable $doc as node() external;
                        let $conteudo := data($doc//div[@class='page fontsize pl printing'])
                        let $titulo := data($doc//h1)
                        let $data := data($doc//h1/em)
                            return
                                <noticia>
                                    <titulo>{data($titulo)}</titulo>
                                    <conteudo>{data($conteudo)}</conteudo>
                                    <data>{data($data)}</data>
                                </noticia>
                    ]]></xq-expression>
                </xquery>
            </body>
        </loop>
        <![CDATA[ </terra_noticias> ]]>
    </file>
</config>
```

Figura 6. Arquivo de configuração XML.

A figura 7 demonstra a tela do protótipo onde será informado o caminho do arquivo XML(figura 6):

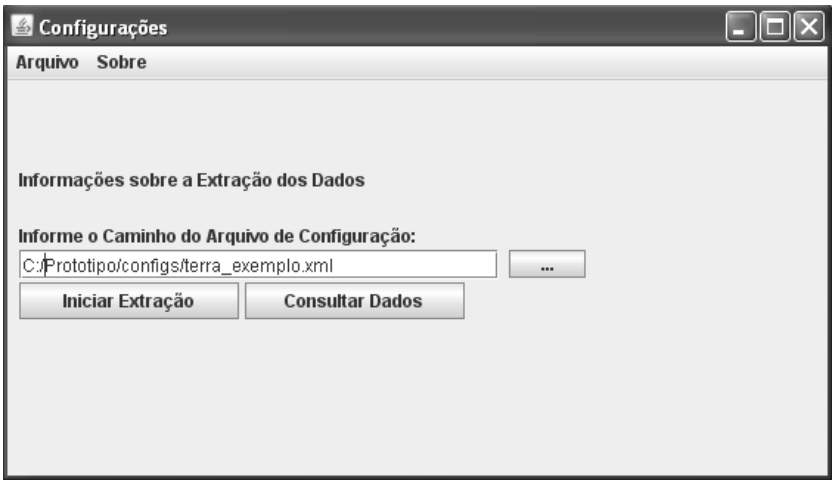


Figura 7. Tela de configuração do protótipo.

O arquivo de configuração (figura 6) possui dois conjuntos de processos bem definidos, também chamados de *pipelines*. O primeiro *pipeline* executa as seguintes etapas:

a) linha 3: a variável *inicioUrl* é definida com o endereço onde serão extraídas as notícias;

b) linha 5: o elemento *file* define a criação do arquivo de saída através do atributo *action*;

c) linha 11 a 17: o elemento *http*(linha 14) baixa o conteúdo do endereço especificado. O elemento *html to xml*(linha 13) converte o conteúdo em um arquivo *xhtml*, que contém as *URLs* das notícias através das instruções contidas no elemento *xpath*(linha 12).

O endereço especificado na variável *inicioUrl* contém apenas o título das notícias e seus respectivos *links* para acesso as notícias completas. Desta forma, para alcançar o objetivo e recuperar o conteúdo da matéria, é necessário realizar uma varredura em cada *URL* coletada no primeiro *pipeline*. O segundo processo (linha 19 a 39) contido no arquivo alcança

este objetivo com as seguintes etapas:

a) linha 23: é feito o *download* da página de cada matéria através das URLs obtidas no processo anterior;

b) linha 22: Convente o arquivo *html* em *xhtml*, setando este conteúdo na variável *doc*(linha 21);

c) através dos elementos *xquery* e sub-elemento *xq-expression*(linha 26 a 37) é criado o conteúdo do arquivo *XML* de saída. Através das expressões regulares e a linguagem *xquery* são encontrados os valores titulo, conteúdo e data da notícia;

d) enfim, na linha 41, é encerrado o conteúdo do arquivo de saída com a *tag* nó “</terra_noticias>”.

Após o processamento destas instruções é gerado o seguinte arquivo:

```
<terra_noticias data_extracao="30.05.2010">
    <noticia>
        <titulo> Descoberta científica abre portas para tratamento de lúpus     </titulo>
        <conteudo>
        Cientistas dos Institutos Nacionais de Saúde dos Estados Unidos     descobriram que a ativação
de basófilos, um tipo de leucócito do sistema imunológico, causa     danos aos rins de ratos, em um
experimento que pode ajudar a desenvolver novos tratamentos para o lúpus.O lúpus é uma doença que afeta
vários órgãos do corpo e para a qual não existe cura.A pesquisa dos institutos foi divulgada neste
domingo pela revista "Nature Medicine" e foi dirigida pelo doutor Juan Rivera, diretor do Instituto
Nacional de Artrites e Doenças Músculo- Esqueléticas  e da Pele dos EUA (Niams, na sigla em
inglês).Rivera declarou que cerca de 1,5 milhão de americanos sofrem de lúpus e aproximadamente entre
60% e 70% apresentam sua forma mais grave.O doutor afirmou que as estatísticas sobre mortalidade são
menos exatas, mas calculou que o número de mortes causadas pelo tipo mais grave chegue a "milhares" por
ano..</conteudo>
        <data>30 de maio de 2010 </data>
    </noticia>
    <noticia>
        <titulo> Nem leão, nem elefante, rei da savana é o cupim, diz estudo</titulo>
        <conteudo>
        Um estudo da Universidade de Harvard, nos Estados Unidos, indica que os verdadeiros reis da
savana africana não são os leões, mas sim os cupins. Segundo o pesquisador Robert Pringle, a rede de
colônias criadas pelas colônias do inseto influencia mais na população de animais que os grandes
predadores ou os gigantes da região, como os elefantes e as girafas. As informações são da
Agência Fapesp.Segundo a pesquisa, a ação do cupim contribui enormemente para a produtividade do solo,
que acaba por estimular a produção vegetal e, por consequência, animal. Os cientistas afirmam que a
distribuição dos cupinzeiros por uma área maior maximiza a produtividade de todo o ecossistema."Não são
os predadores carismáticos - como leões e leopardos - que exercem os maiores controles em populações.
Em muitos aspectos, são os pequenos personagens que controlam o cenário. No caso da savana,
aparentemente os cupins têm uma tremenda influência e são fundamentais para o funcionamento do
ecossistema", diz Robert Pringle.Os pesquisadores estudaram cupinzeiros na região do Quênia central.
        </conteudo>
        <data>30 de maio de 2010</data>
    </noticia>
</terra_noticias>
```

Figura 8. Arquivo XML dos dados extraídos.

A figura 8 ilustra o arquivo extraído, onde é perceptível uma organização dos dados. O nó <terra_noticias> indica o início do arquivo, ele possui a data da extração dos dados. O nó principal contém o sub-elemento <noticia>, que indicará cada notícia contida no arquivo. Da mesma forma o nó <noticia> possui os sub-elementos <titulo> que corresponde ao título da notícia, <conteúdo> contém a matéria na íntegra e a *tag* <data> representa a data em que a notícia foi apresentada no site.

Após a etapa de extração efetiva dos dados, é possível encontrar um arquivo que contém as informações de maneira formatada ou estruturada, mesmo que seja uma pequena estrutura. Desta forma, é possível distinguir do conteúdo não estruturado, os principais sub-conjuntos e processar estes dados de uma forma lógica.

A próxima etapa que o protótipo realiza é a comparação das informações extraídas com as informações que o usuário deseja encontrar. Para a leitura e identificação do documento XML gerado pela aplicação, foi utilizada a biblioteca Java XSTREAM, que possibilita a manipulação e transformação do conteúdo XML em Objetos Java, que neste caso se chama “Noticia”, e possui os seguintes atributos: Título, Conteúdo e Data.

A partir deste momento, recuperado o conteúdo do arquivo XML, o protótipo pode realizar a comparação utilizando cálculos de distância baseado em um algoritmo de similaridade.

7.3.2 Cálculo da Similaridade

Uma função de similaridade é um algoritmo que compara dois ou mais itens, e realiza comparações e cálculos para medir numericamente a semelhança existente entre eles. A maioria das funções de similaridade recebe como entrada dois itens e retorna como resultado um valor real normalizado no intervalo [0, 1], onde 0 representa a completa desigualdade entre as entradas e 1 significa que os itens são equivalentes para a função.

Para determinar o valor final de similaridade, um algoritmo analisa certas características dos itens e computa um valor baseado nessa comparação. Dessa forma, duas propriedades podem ser associadas aos algoritmos de similaridade:

a) reflexão: o resultado da função de similaridade comparando um item com ele mesmo é sempre o valor máximo definido pela função, ou seja, será sempre o valor 1.

b) simetria: o valor calculado da similaridade entre um item A e um item B é

igual ao valor calculado de similaridade entre o item B e o item A.

O protótipo desenvolvido utiliza o algoritmo de similaridade para comparar o que o usuário necessita, com as informações extraídas. Com base neste escopo, o sistema desenvolvido utiliza o algoritmo editDistance (LEVENSHTEIN, 1966) para calcular a similaridade entre os dados. Os fatores determinantes para a escolha da utilização deste algoritmo no protótipo foram sua fácil implementação e sua eficiência na consulta de dados não estruturados.

Abaixo são apresentados dois exemplos de preenchimento de matriz utilizando o algoritmo EditDistance. O primeiro compara as palavra “computacao” e “computador”, o segundo compara duas palavras iguais:

		c	o	m	p	u	t	a	c	a	o
	0	1	2	3	4	5	6	7	8	9	10
c	1	0	1	2	3	4	5	6	7	8	9
o	2	1	0	1	2	3	4	5	6	7	8
m	3	2	1	0	1	2	3	4	5	6	7
p	4	3	2	1	0	1	2	3	4	5	6
u	5	4	3	2	1	0	1	2	3	4	5
t	6	5	4	3	2	1	0	1	2	3	4
a	7	6	5	4	3	2	1	0	1	2	3
d	8	7	6	5	4	3	2	1	1	2	3
o	9	8	7	6	5	4	3	2	2	2	3
r	10	9	8	7	6	5	4	3	2	1	3

Figura 9. Exemplo 1, matriz do algoritmo editDistance.

Na figura 9, está sendo calculada a distância de edição entre as cadeias “computacao” e “computador”. A distância de edição calculada pelo algoritmo editDistance

entre as cadeias de caracteres apresentada na célula M[10, 10] são três operações . Então, aplicando-se a fórmula, observa-se que a similaridade entre as duas palavras é 0,7.

$$Sim(computacao, computador) = \frac{10 - 3}{10} = 0,7$$

Figura 10. Exemplo 1, aplicação da fórmula de similaridade.

O segundo exemplo faz a comparação entre as palavras “trabalho” e “trabalho” exibidas na figura 11:

		t	r	a	b	a	l	h	o
	0	1	2	3	4	5	6	7	8
t	1	0	1	2	3	4	5	6	7
r	2	1	0	1	2	3	4	5	6
a	3	2	1	0	1	2	3	4	5
b	4	3	2	1	0	1	2	3	4
a	5	4	3	2	1	0	1	2	3
l	6	5	4	3	2	1	0	1	2
h	7	6	5	4	3	2	1	0	1
o	8	7	6	5	4	3	2	1	0

Figura 11. Exemplo 2, matriz do algoritmo editDistance.

Aplicando-se os conceitos apresentados e a fórmula para obtenção da similaridade entre as palavras iguais, o resultado é um 1. A figura 12 demonstra o cálculo:

$$Sim(trabalho, trabalho) = \frac{8 - 0}{8} = 1$$

Figura 12. Exemplo 2, aplicação da fórmula de similaridade.

A função de distância editDistance e o algoritmo de similaridade associado a ela são amplamente utilizados e apresentam excelentes resultados em aplicações que comparam

cadeias de caracteres.

O sistema desenvolvido utiliza este algoritmo para comparar a cadeia de caracteres do conteúdo extraído com os caracteres que o usuário digitar. Como o sistema trabalhará com uma grande cadeia de caracteres extraídos, a comparação não pode se feita de uma vez somente, pois dependendo da diferença entre os tamanhos das cadeias comparadas, o resultado pode apresentar uma similaridade baixa, mesmo que existam palavras iguais sendo comparadas. Desta forma, a comparação será feita separando as cadeias pelos espaços em branco existentes. Exemplo:

Levando em consideração o exemplo do conteúdo extraído, Capítulo 9.4.1, se o usuário digitar a palavra “lúpus” no sistema, este encontrará a noticia com o título “Descoberta científica abre portas para tratamento de lúpus”, pois o protótipo faz a comparação entre as palavras separadamente. Caso contrário, se fosse comparada a palavra “lúpus” com a cadeia inteira do título “Descoberta científica abre portas para tratamento de lúpus” o sistema não encontraria a notícia, pois a similaridade é muito baixa. Abaixo está ilustrada a diferença entre as duas formas de comparação:

Comparando-se as cadeias separadamente o resultado é encontrado facilmente, pois foi encontrada a palavra “lúpus”, a qual possui máxima similaridade (1,0) com a mesma palavra do título:

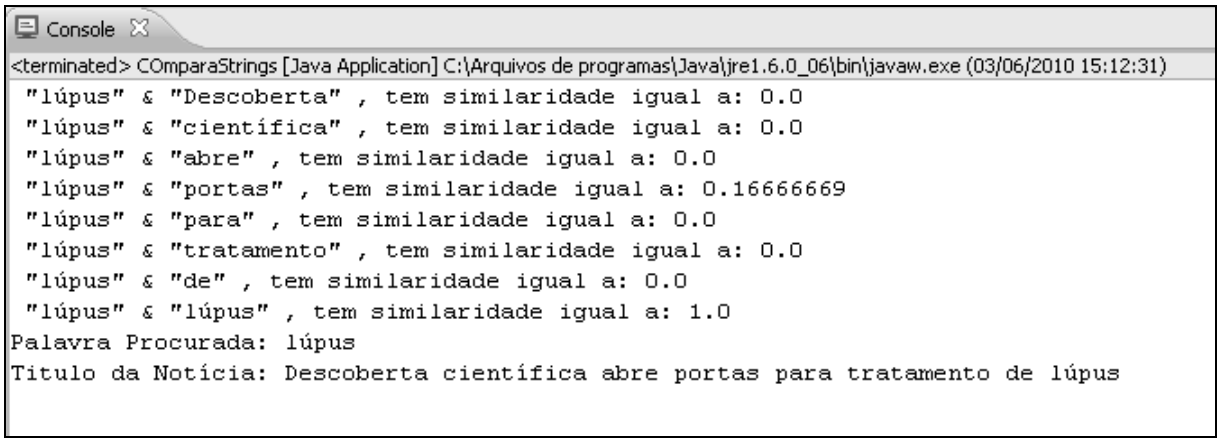


Figura 13. Exemplo de comparação entre cadeias separadas.

Já, na comparação por cadeias completas, a similaridade encontrada (0,0862) é irrisória, ou seja, muito difícil encontrar-se alguma notícia, mesmo que, a palavra buscada se encontre no título da notícia:

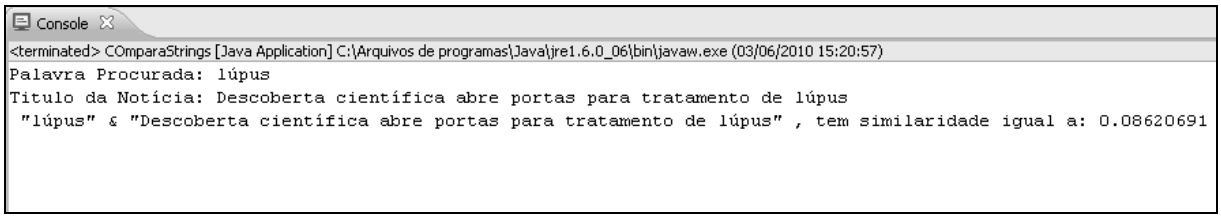


Figura 14. Exemplo de comparação entre cadeias inteiras.

Na implementação para comparação entre os dados extraídos, o protótipo utiliza a biblioteca SimMetrics (http://www.dcs.shef.ac.uk/~sam/simmetrics.html), que se trata de um projeto open source, onde são disponibilizados vários algoritmos de similaridades já implementados, com a documentação completa, possibilitando o maior agilidade no desenvolvimento do projeto.

O protótipo desenvolvido permite, ainda, o usuário escolher a similaridade mínima entre o que foi digitado pelo o usuário e o conteúdo extraído da Web. Na figura 15 está a tela do sistema desenvolvido, onde será feita a consulta das noticias:

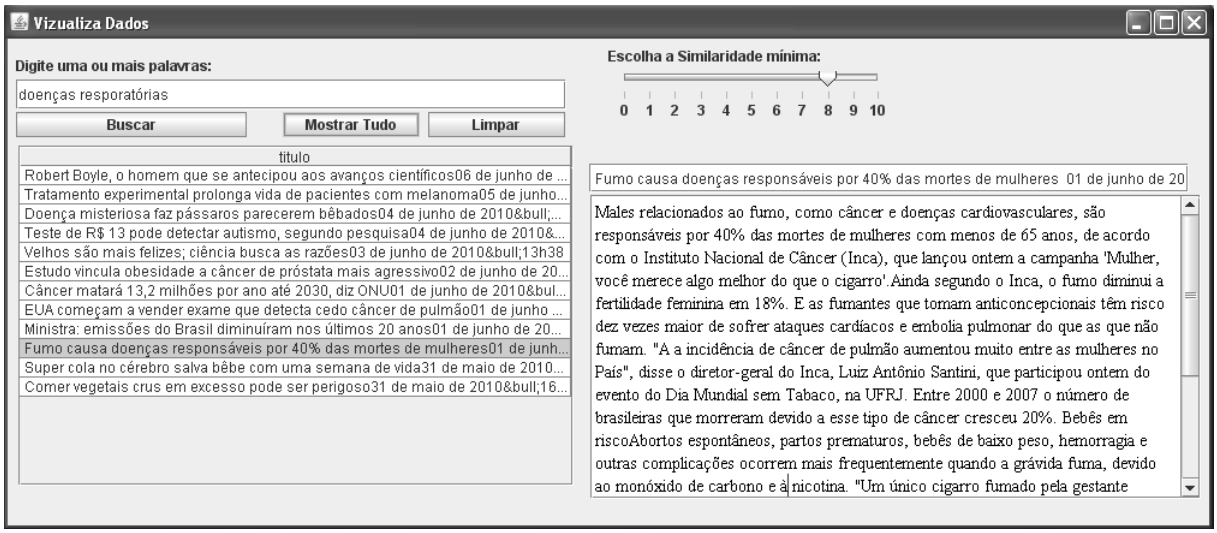


Figura 15. Tela do protótipo para visualização dos dados.

No exemplo da figura 15, o usuário digitou “doenças respiratórias”, escolheu a similaridade 8, entende-se “0,8” e clicou no botão Buscar. Neste momento o sistema começa a comparar as palavras que o usuário digitou com as palavras que estão no conteúdo das notícias. A notícia selecionada na figura foi encontrada pois possui em seu conteúdo a palavra “doenças”, dessa forma, calculando a similaridade, encontra-se o coeficiente 1 entre a palavra “doenças” digitada pelo usuário e a palavra “doenças” contida na notícia. Como o usuário escolheu buscar notícias onde a similaridade entre as palavras é maior que “0,8”, esta notícia foi encontrada com êxito.

Após o processo de extração dos dados, recuperação dos dados e a comparação feita através do algoritmo de similaridade editDistance, o usuário tem a possibilidade de armazenar a notícia em um banco de dados.

7.3.3 Armazenamento das informações

Com o objetivo de recuperar as informações que o usuário necessita de forma rápida e eficiente, o protótipo possui a funcionalidade de armazenar o conteúdo extraído,

neste caso a notícia, em um banco de dados. Para realização deste propósito, o sistema desenvolvido utiliza o SGBDOR PostgreSQL 8.4. Entre os fatores determinantes para a escolha estão:

a) Open Source;

b) possui suporte a modelagem Objeto Relacional;

c) apóia um número grande de interfaces de programação, como ODBC e Java (JDBC);

d) sua licença é BSD, portanto pode ser utilizado, modificado e distribuído por qualquer pessoa ou empresa para qualquer finalidade, sem encargo, em quaisquer dos sistemas operacionais suportados.

De posse de um SGBDOR completo e funcional, e também dos dados extraídos e classificados de acordo com a similaridade e o texto digitado pelo usuário, o sistema possibilita o armazenamento destas informações.

Com o objetivo de dinamizar o sistema e diminuir o acoplamento entre classes do sistema e a conexão com o banco de dados, este protótipo utiliza o *Hibernate*. O *Hibernate* é um *framework* para o mapeamento objeto-relacional escrito na linguagem Java, este *framework* facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de arquivos (XML) para estabelecer esta relação.

O objetivo deste *framework* é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do

modelo relacional. Em especial, no desenvolvimento de consultas e atualizações dos dados.

Sua principal característica é a transformação das classes em Java para tabelas de dados e o inverso. O *Hibernate* gera as chamadas SQL e libera o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados SQL.

Para realizar as operações no banco de dados o *Hibernate* disponibiliza a HQL (*Hibernate Query Language*), que é um dialeto SQL para o *Hibernate*. Ela é uma poderosa linguagem de consulta que se parece muito com a SQL, mas a HQL é totalmente orientada a objeto, incluindo os paradigmas de herança, polimorfismo e encapsulamento.

Desta forma, para realizar o armazenamento dos dados no Banco de dados, primeiramente foi necessário criar uma tabela no banco de dados chamada noticia. Após a criação da tabela e suas colunas foi necessário fazer o mapeamento da classe Java para a tabela do Banco de dados utilizando um arquivo XML, como mostra a figura 16:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
   "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="net">
   <class name="noticia">
      <id name="id" column="id" type="integer">
      <generator class="increment" ></generator>
      </id>
      <property name="titulo" />
      <property name="conteudo" />
      <property name="data" />
   </class>
</hibernate-mapping>
```

Figura 16. XML de configuração do Hibernate.

Este mapeamento é necessário para o *Hibernate* identificar qual é a ligação entre a

classe Java e a tabela no banco de dados, pois como foi citado anteriormente, este framework trabalha com persistência de objetos, ou seja, na prática, quando o sistema for salvar os dados no banco é necessário informar um Objeto. Após isso o *hibernate* vai identificar, através desta configuração, em quais colunas da tabela do banco de dados os atributos do objeto serão armazenados.

Este framework também possibilita a integração com diversos banco de dados, como neste projeto foi utilizado o SGBDOR PostgreSQL, a configuração foi realizada conforme a classe ilustrada na figura 17:

```
package DB;

import net.noticia;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class conexao {

    public static SessionFactory factory(Object classe){

        Configuration  config = new Configuration().
        setProperty("hibernate.dialect", "org.hibernate.dialect.PostgreSQLDialect").
        setProperty("hibernate.connection.driver_class", "org.postgresql.Driver").
        setProperty("hibernate.connection.url", "jdbc:postgresql://localhost:5432/prototipo").
        setProperty("hibernate.connection.username", "postgres").
        setProperty("hibernate.connection.password", "admin").
        setProperty("hibernate.show_sql", "true");
        config.addClass(((noticia)classe).getClass());
        return config.buildSessionFactory();
    }
}
```

Figura 17. Classe de configuração para conexão com o banco de dados.

Está classe é responsável pela conexão ao banco de dados, onde é informado o dialeto da linguagem SQL, o driver de conexão, o endereço da conexão, usuário e senha, entre outras propriedades que podem ser adicionadas de acordo com a necessidade.

Após todas as configurações realizadas, o *hibernate* pode realizar o armazenamento dos dados. Desta forma, o sistema possibilita consultar estes dados utilizando

a linguagem HQL para recuperar os objetos do banco de dados. De posse dos objetos recuperados é possível retornar ao usuário àquela informação que ele solicitou, de acordo com a palavra digitada e a similaridade mínima escolhida.

7.4 RESULTADOS OBTIDOS

Os resultados foram obtidos por meio do levantamento bibliográfico realizado e pela utilização de ferramentas e bibliotecas disponibilizadas em projetos *open source*. Estas facilitaram a compreensão no momento da implementação do protótipo, ao passo que o levantamento bibliográfico permitiu o entendimento dos procedimentos que devem ser realizados, bem como o conhecimento da teoria e dos diversos métodos para realizar cada etapa deste trabalho.

O protótipo desenvolvido realizou a extração das notícias de forma eficiente utilizando o mecanismo de extração do projeto Web Harvest. Este projeto é open source, possui uma gama de recursos para extração de dados e disponibiliza uma biblioteca Java contendo o mecanismo de extração de dados, a qual foi utilizada no desenvolvimento do protótipo. Para realização da extração foi desenvolvido um arquivo de configuração no formato XML, que contém o endereço do site onde será realizada a extração, as parametrizações necessárias, e o formato de como é o formato do arquivo XML de saída. Após a execução é retornado um arquivo XML com todas as notícias disponíveis no site, contendo data da extração, título, data da notícia e o conteúdo da notícia.

Com os dados extraídos, o protótipo permite o usuário pesquisar as notícias através da interface implementada, utilizando o algoritmo editDistance. Este algoritmo realiza o cálculo do número mínimo de operações necessárias para transformar a palavra digitada

pelo usuário nas palavras que estão no conteúdo da noticia. EditDistance é muito útil para aplicações que precisam determinar quão semelhantes são duas palavras ou strings, como os verificadores ortográficos.

Na mesma interface desenvolvida, além de todas as notícias que já foram armazenadas automaticamente, o usuário pode armazenar as notícias favoritas no banco de dados. Com as notícias favoritas é armazenado também o conteúdo que o usuário digitou para buscar a notícia, caso utilizado. Isto, com objetivo de manter um histórico das pesquisas e abrir opções para trabalhos futuros poderem implementar novos mecanismos de buscas.

Para o mapeamento das noticias e a armazenagem dos dados foi utilizado o *framework Hibernate,* que disponibiliza uma forma eficiente e flexível para persistência no banco de dados escolhido, neste trabalho o PostgreSQL 8.4.

Este trabalho diferencia-se por ser um dos poucos trabalhos existentes relacionados aos dados não estruturados, e também por concentrar três etapas em um único sistema, são elas: a extração dos dados encontrados na web, a consulta utilizando um algoritmo de similaridade e a possibilidade de armazenar as informações de forma flexível. Desta forma, contribuindo para o ambiente acadêmico com uma inovadora opção para a realização de pesquisas.

CONCLUSÃO

Neste trabalho foi possível compreender o funcionamento dos métodos de extração de dados, em especial a extração de dados na WEB, e entender as características envolvidas no processo, a partir de testes feitos com as ferramentas disponibilizadas em projetos open-source. Além de ter sido validado e entendida a fundamentação teórica do mesmo, a partir da pesquisa e implementação desta etapa no protótipo.

Outro grande propósito atingido, foi a compreensão de como é realizado o cálculo de similaridade entre palavras, utilizando o algoritmo editDistance. Com o estudo feito acerca dos algoritmos de similaridade, foi possível entender, na teoria e na prática, cada passo para montar uma matriz, fazer a comparação entre as palavras e aplicar a fórmula para a obtenção do coeficiente de similaridade.

Sem menos importância, a pesquisa feita sobre banco de dados e mapeamentos Objeto-Relacional proporcionou o desenvolvimento do protótipo que pode ser flexível a qualquer banco de dados suportado pelo framework *Hibernate*, bastando a configuração em uma única classe. Além disso, proporcionou melhor entendimento sobre orientação a objetos, bem como suas características e aplicações na prática.

Uma grande dificuldade encontrada durante a elaboração deste trabalho, foi a escassez de livros sobre dados não estruturados, sendo que esta dificuldade foi, em sua parcialidade, suprida com artigos, publicações e monografias que continham bastante referência sobre dados semi-estruturados.

Outra grande dificuldade foi encontrada na implementação, devido as freqüentes mudanças do ambiente WEB, como exemplo pode se citar o site www.globo.com: O

protótipo estava sendo desenvolvido para se adequar às noticias do site Globo.com, porém após a feita configuração para extração dos dados e toda a documentação da implementação, o site foi totalmente remodelado, de forma que todo o trabalho teve que ser desfeito. Após isto acontecer, o site www.terra.com.br foi o escolhido para exemplificar o funcionamento do protótipo. Outro fator que dificultou o desenvolvimento do trabalho foi a falta de padronização no desenvolvimento Web.

De forma geral, pode-se concluir que o ambiente Web dispõe de um imenso conteúdo de dados não estruturados. Desta forma, o estudo e a implementação do protótipo desenvolvido contribuiu para o entendimento do processo como um todo, desde a extração de uma informação contida na web, à comparação realizada por um algoritmo de similaridade, até o armazenamento da informação de forma estruturada em um banco de dados.

Assim, sugere-se como trabalhos futuros:

a) aplicar outros algoritmos de similaridade para a comparação dos dados;

b) implementação de uma ferramenta para geração automática do arquivo XML utilizado para extração dos dados;

c) implementar a integração com redes sociais, como o Twitter, capturando dados baseados nos algoritmos de similaridade;

d) utilização de mecanismos de aprendizagem automática em tarefas de extração de dados;

e) implementar extração de dados utilizando Yahoo! Pipes. (http://pipes.yahoo.com).

REFERÊNCIAS

ABITEBOUL, S.; BUNEMAN, P.; SUCIU, D. Data On the Web – from relations to semistructured data and XML. San Francisco: Morgan Kaufman, 2000. 257 p.

ABITEBOUL, Serge; QUASS, Dallan; MCHUGH, Jason; WIDOM, Jennifer; WIENER, Janet. The Lorel query language for semistructured data. International Journal on Digital Libraries, Abril 1997.

ALLEN, J.F..Natural Language Understanding. 2ª edition, 1995. 625p. Disponível em: <http://www.uni-giessen.de/~g91062/Seminare/gk-cl/Allen95/al1995co.htm>. Acessado em setembro de 2009.

APPELT, Douglas, ISRAEL, David. Introduction of Information Extraction Technology. International Joint Conference on Artificial Inteligence, 41 p. Disponível em: <http://www.ai.sri.com/~appelt/ie-tutorial/IJCAI99.pdf>. Acessado em novembro de 2009.

BAEZA, Ricardo Yates. Modern Information Retrieval. New York: ACM Press Book, 1999.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. **Scientific American**. v. 284, n. 5, p. 28-37, 2001.

BOND, Martin; HAYWOOD, Dan; LAW Debbie; LONGSHAW, Andy; ROXBURGH, Peter. Aprenda J2EE- **Com EJB, JSP, Servlets, JNDI, JDBC, XML**. Makron Books. São Paulo, SP. Tradução de João Eduardo Nóbrega Tortello. 2005.

BUNEMAN, Peter. Semistructured Data. In: SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS (ACM), 16, 1997. Tutorial in Proceeding, 1997. p. 5.

CALIFF, M.E.; MOONEY, R.J. Relational learning of pattern-match rules for information extraction, Proceedings of the 16th National Conf. on AI, 1999.

CARLAN, Eliana. **Ontologia e Web Semântica.** 2006. 61 f. Monografia (Bacharel) - Universidade de Brasilia, Brasília, 2006.

CARNEIRO, Raquel Elias; BRITO, Parcilene Fernandes de. **Definição de uma Ontologia em OWL para Representação de Conteúdos Educacionais.** In: VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS, 2005, Palmas. **Anais…** Palmas: 2005.

CAVALCANTI, Cordelia R. Indexação & tesauro: metodologia e técnicas. Brasília: Associação de Bibliotecários do Distrito Federal, 1978. 89 p.

CHÁVEZ, E.; NAVARRO, G.; BAEZA, R.; MARROQUÍN, J. L.. Searching in metric spaces. ACM Computing Surveys (CSUR), 2001.

DATE, C. J., Introdução a sistemas de banco de dados. 7ª Ed. Editora Campus, 2000. P 803.

DINIZ, Danielle Dornellas. **A interação no ensino a distância sob a ótica dos estilos de aprendizagem.** Escola de Engenharia de São Carlos, São Carlos – SP. 2008. disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18140/tde-20122007-141314/>. Acesso em: 04/04/2009.

EIKVIL, Line. Information Extraction from World Wide Web - A Survey. Norwegian Computing Center, 1999.

FAGUNDES, Ricardo Cassiano; KROTH, Eduardo. Aplicação de consultas baseadas em similaridade em ambientes de conhecimento definidos por tesauros. Universidade de Santa Cruz do Sul (UNISC).

FALOUTSOS, C.. Indexing of multimedia data. **In Multimedia Databases in Perspective**, p. 219–245, 1997.

FERNANDEZ, Mary; FLORESCU, Daniela; LEVY, Alon; SUCIU, Dan. A query language for a web-site management system. SIGMOD Record, Setembro, 1997.

FLORESCU, Daniela; LEVY, Alon. MENDELZON, Alberto. Database Techniques for the World Wide Web: **A Survey**. SIGMOD Record, Setembro de 1998.

FREITAG, D. Information extraction from HTML: **Application of general machine learning approach**. In Proc. of the AAAI, 1998.

HAMMER,J.; GARCIA-MOLINA, H.; CHO J.; ARANHA, R. and CRESPO, A.. "Extracting Semistructured Information from the Web". In Proceedings of the Workshop on Management of Semistructured Data. Tucson, Arizona, Maio 1997.

HEUSER, Carlos Alberto; DORNELES, Carina; NORONHA, Marilene. Ligando a Tecnologia de Bando de Dados com a Gestão de Documentos. UFRGS, 1998.

HOLMES, James; SCHILDT, Herbert. A arte do Java. Editora Campus. Rio de Janeiro, RJ. Tradução de Edson Frumankiewcz. 2003.

LAENDER, Alberto H. F; SILVA, Elaine E; ALTIGRAN S. da Silva. DEByE - uma ferramenta para extração de dados semi-estruturados, 1999.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, 1966.

LIRA, Heremita Brasileiro. **Uma ontologia para a descrição da semântica da IMML.** 2006. 195 f. Dissertação (Pós) - Universidade Federal do, Natal, 2006.

LONGMIRE, W. **A Primer On Learning Objects.** American Society for Training & Development. Virginia. USA. 2001.

KORN, F.; SIDIROPOULOS, N.; FALOUTSOS, C.; SIEGEL, E.,; PROTOPAPAS, Z.. Fast nearest neighbor search in medical image databases. In Vijayaraman, T. M., Buchmann, A. P., Mohan, C., & Sarda, N. L., editors, Proceedings of the 22th International Conference on Very

Large Data Bases (VLDB), p. 215–226, 1996.

MANICA, Edimar; CERVI, Cristiano Roberto; GALANTE, Renata de Matos. Um Processo Automático para Extração de Metadados de Documentos PDF Usando um Template XML.

MELLO, Ronaldo; DORNELES, Carina; KADE, Adrovane; BRAGANHOLO, Vanessa; HEUSER, Carlos. Dados Semi-Estruturados. In: Simpósio brasileiro de banco de dados. 2000, João Pessoa, PB. Anais...João Pessoa: [s.n.], 2000.

MELLO, Alexandre S.; MACEDO, Hendrik T.Extração Automática de Conteúdo Semi-Estruturado na Web:Estudo de Caso do Futebol Brasileiro. 2009.

MENDELZON, G. Mihaila; MILO , T.. Querying the world wide web. International Journal on Digital Libraries, Abril 1997.

MEYER, B.; COLEMAN, D.; ARNOLD, P.; BODOFF, S.; DOLLIN, C. ; GILCHRIST, H.; HAYES, F. ; JEREMAES, P. Desenvolvimento Orientado a Objetos o Método Fusion , 1ª edição, Rio de Janeiro, Campus, 1996.

MOLOSSI, Sinara. **Inserção da Biblioteca Digital de Teses e Dissertações no Contexto da Web Semântica::** Construção e Uso da Ontologia. 2008. 228 f. Dissertação (Mestre) - Universidade Federal de Santa Catarina, Florianópolis, 2008.

MURRAY, B. H.; MOORE, A. **Sizing the Internet.** Jul. 2000. Disponível em: <http://www.cyveillance.com/web/us/downloads/Sizing_the_Internet.pdf>. Acesso em: 01/05/2009.

NETO, Wilson Castello Branco. **Web Semântica na Construção de Sistemas de Aprendizagem Adaptativos**. Florianópolis – SC. 2006. p. 237.

OLIVEIRA, Ismênia Ribeiro de; NETO, Rafael de Sousa; GIRARDI, Rosário.**Uma Ontologia para a Especificação de Sistemas de Padrões.** Universidade Federal do Maranhão. São Luiz – MA.

PELLEGRINO, Grazieno et al. XML e Banco de Dados para WEB. Instituto de Matemática – Universidade Federal da Bahia. 2008.

PERÉZ, A. G.: CORCHO, O. **Ontology Languages for the Semantic Web. IEEE Intelligent Systems**. v. 13, n. 1, p. 54-60. Jan. 2002.

RUMBAUGH, J. Modelagem e Projetos baseados em Objetos; tradução de Dalton Conde de Alencar, Rio de Janeiro, Campus, 1994.

RAY, S.; CRAVEN, M. Representing sentence structure in hidden markov models for information extraction, In Proc. of the Int. Joint Conf. on Artificial Intelligence, 2001.

SANTOS FILHO, J. Camilo dos. Pesquisa quantitativa versus pesquisa qualitativa: o desafio paradigmático. In: SANTOS FILHO, J. Camilo dos; GAMBOA, Silvio Sánchez. Pesquisa educacional: quantidade-qualidade. 4. ed. São Paulo: Cortez, 2001.

SILVEIRA, Fábio F. Técnicas para Banco de Dados para World Wide Web. Disponível em: <http://www.alternet.com.br/users/ffs/download/Art_BD_ffs.pdf>. Acessado em outubro de 2009.

SILVEIRA, Iraci Cristina da. Extração Semântica de Dados Semi-Estruturados através de Exemplos e Ferramentas Visuais. Porto Alegre, 2001.

SODERLAND, S., Learning information extraction rules for semi-structured and free text, Vol. 34(1-3), 1999.

STEPHEN, J. DeRose. What do those weird XML types want, anyway ? Proceedings of the 25th VLDB Conference, Edinburgh, 1999.

SUCIU, D. Management of Semistructured Data. SIGMOD Record, 4ª ed., p.4-7, Dezembro, 1997.

THIRAN, Hainaut. Wrapper development for legacy data reuse. Reverse Engineering, 2001. p. 198-207.

VINSON, Alexander Richard. PathSim: um algoritmo para calcular a similaridade entre caminhos XML. Porto Alegre, 2007.

WESSMAN, Alan; LIDDLE , Stephen W.; EMBLEY , David W.. A Generalized Framework for an Ontology-Based Data-Extraction System. Brigham Young University, EUA. Março, 2005.

WIEDERHOLD, G. Mediators in the Architecture of Future Information Systems. Computer, v. 25,n.3, Março, 1992, p.38-49.

WIVES, Leandro K. Um Estudo Sobre Técnicas de Recuperação de Informações com ênfase em Informações Textuais. Porto Alegre: UFRGS, 1997.

Extração Semi-Automática de Dados Não Estruturados na Web baseada em Algoritmos de Similaridade para Armazenamento em Banco De Dados Objeto-Relacional

Mário Luís Scarpari Citadin

Departamento de Ciência da Computação
Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC – Brasil

marioluiscitadin@gmail.com

Abstract. *This article describes the types of data in the web and demonstrates a way to manipulate them and store them in a database. The main steps to accomplish the goal and manipulate the data, especially unstructured data, are: the analysis of the desired data, web data extraction using wrappers, data classification using the algorithm editDistance and storage in the database using the framework Hibernate for object-relational mapping.*

Resumo. *Este artigo descreve os tipos de dados existentes na Web e demonstra uma forma de manipulá-los e armazená-los em um banco de dados. As principais etapas para realizar o objetivo e manipular os dados, principalmente os dados não estruturados, são a análise dos dados desejados, extração dos dados na Web utilizando wrappers, classificação dos dados através do algoritmo editDistance e o armazenamento no banco de dados utilizando o framework Hibernate para o mapeamento objeto-relacional.*

1. Dados no Ambiente Web

Segundo ABITEBOUL (2000), as informações ou dados eletrônicos podem ser classificados em três tipos: dados não estruturados, como arquivos de textos; dados estruturados, como dados em um bancos de dados; e dados semi-estruturados que formam uma classe intermediária entre esses dois tipos.

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

Porém, nota-se atualmente que boa parte dos dados disponíveis para acesso eletrônico não estão mantidos em Banco de Dados. Alguns exemplos são diretórios de arquivos de documentos de uma organização (Atas de reuniões, manuais, entre outros) ou informações acessíveis através da Internet. A justificativa para o fato decorre da própria natureza destes documentos. Dados encontrados na Web apresentam uma organização bastante heterogênea, que pode variar de um texto sem nenhuma formatação até um conjunto de registros bem estruturado. Além disso, o volume destes dados pode ser grande e com muitos relacionamentos. Considerando dados referentes a um curriculum vitae, por exemplo, pode-se ter um pequeno texto informal descrevendo dados pessoais e experiência profissional ou,

oposto a isso, um documento organizado em seções e subseções, com referências para dados das empresas e instituições onde a pessoa trabalhou.

Com a evolução da Internet, a World Wide Web (Web) tornou-se um vasto repositório de dados dos mais variados tipos e formatos. Além disso, teve sua estrutura pautada na construção de documentos de textos e imagens, com o simples objetivo de informar, sem interação com os usuários. Eram informações em texto livre, som e imagem estáticos, desenvolvidos e disponibilizados apenas para acesso (PELLEGRINO et al, 2008).

Entretanto, os dados não estruturados disponíveis na Web são, em geral, difíceis de serem utilizados e manipulados pela maioria dos usuários da Internet. A principal dificuldade deve-se ao fato de que esses dados não podem ser consultados e manipulados através de técnicas de indexação e consulta como as encontradas em ambientes tradicionais de banco de dados. Outra grande dificuldade dos usuários na Web é que a maioria dos sites de recuperação de dados não possuem métodos sofisticados para consultar e manipular as informações solicitadas pelo usuário. Dessa forma, o resultado de uma busca geralmente não traz somente os dados mais relevantes, gerando insatisfação no seu uso, tornando o processo de busca bastante exaustivo.

Nos últimos tempos, a necessidade por ferramentas de pesquisa, filtragem e manipulação de informações relevantes, disponibilizadas em meio eletrônico, tornou-se essencial. Esta necessidade está presente em diversos contextos, desde ambientes empresariais, para a manipulação de dados internos, até instituições de ensino e pesquisa através da troca de produção científica.

Baseado nas dificuldades e problemas demonstrados na forma atual de tratamento dos dados, este trabalho apresenta uma ferramenta realizar a extração semi-automática de dados não estruturados na web, utilizando o algoritmo de similaridade editDistance para a consulta aos dados extraídos, e o armazenamento em um banco de dados utilizando o mapeamento objeto-relacional.

2. Análise dos Dados

A primeira etapa para poder manipular os dados não estruturados consiste em definir quais dados se deseja se obter e onde estão localizados estes dados. Esta etapa é definida como análise dos dados e tem o objetivo de modelar e delimitar as características das informações desejadas.

A forma mais usual de modelar dados não estruturados e semi-estruturados é através de grafos direcionados rotulados, onde os vértices representam objetos identificáveis e as arestas são arcos para outros objetos que fazem parte da sua estrutura, que é hierárquica. Um rótulo presente em um arco indica o nome do atributo do objeto de onde o arco parte (objeto origem), podendo ainda informar o tipo de dado do objeto alcançável através do arco (objeto destino). Como cada ocorrência de dado pode ter uma estrutura diferente, não deve haver restrição no número de arcos que partem de um objeto origem. Todo objeto tem um vértice raiz que é o ponto de partida para a investigação da sua estrutura hierárquica (BUNEMAN 1997).

O modelo de grafo é apropriado para representar dados não estruturados e semi-estruturados pois a estrutura particular implícita de cada dado já está embutida no próprio grafo. Neste

sentido, os arcos têm uma importância fundamental, uma vez que nomeiam os atributos e tipos que fazem parte da estrutura do dado (SUCIU, 1997).

A Figura 1 ilustra um exemplo de um objeto semi-estruturado representado no grafo direcionado rotulado (MELLO, 2000):

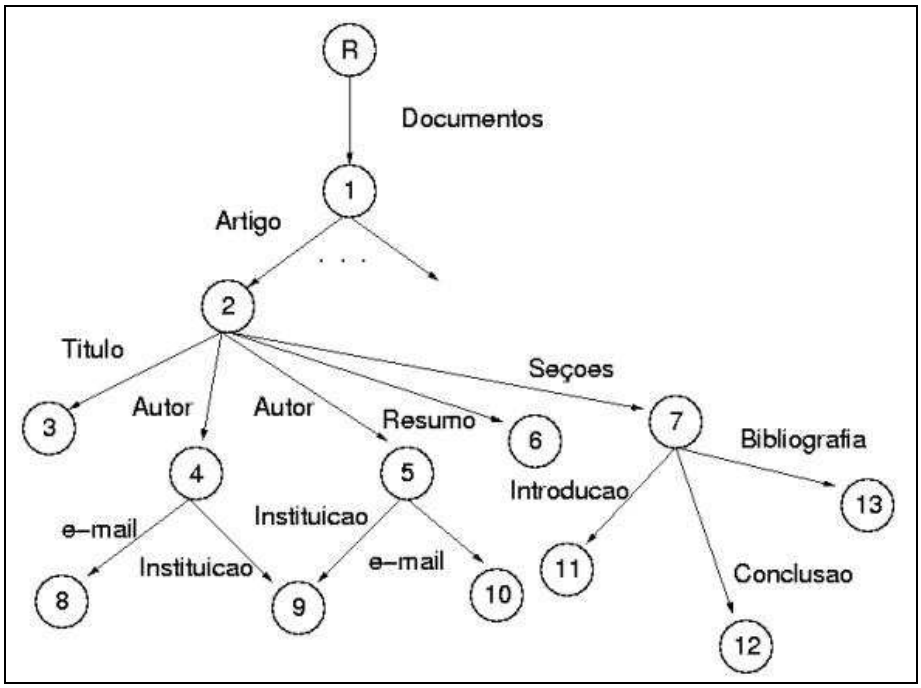


Figura 18. Grafo Direcionado Rotulado.

Fonte: PIMENTA (2003).

Segundo MELLO (2000), este é um modelo flexível, pois não considera a existência de um esquema fixo para um conjunto de dados. Toda informação esquemática está incluída nos rótulos, que podem mudar dinamicamente. É considerado, também, um modelo para instâncias de dados semi-estruturados, pois representa valores de dados e associa rótulos a cada valor para descrever o seu significado.

Neste trabalho o local escolhido para análise e coleta dos dados foi uma página de notícias do site Terra (HTTP://www.terra.com.br). A figura 2 demonstra a escolha dos dados que se deseja obter:

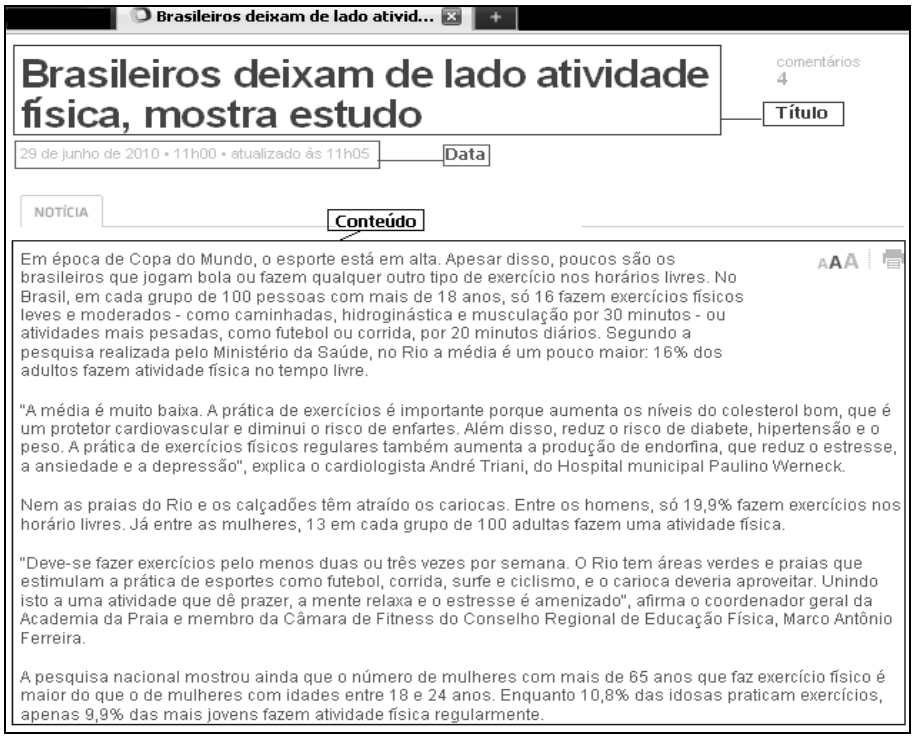


Figura 19. Página do site Terra.

A análise requer que o usuário possua conhecimento de HTML e XML. Esta etapa consiste em localizar os dados que se deseja obter contido no código-fonte da página analisada. Após encontradas as tags HTML onde estão contidos o título, a data e o conteúdo da notícia é possível configurar e parametrizar a extração dos dados.

3. Extração dos Dados

No gerenciamento de dados não estruturados, a extração de dados faz parte do processamento de uma consulta, além de outras etapas executadas, como *parsing* e otimização (BUNEMAN, 1997). A etapa de extração é responsável pela recuperação de informações contidas nas fontes de dados, porém, sem retornar os documentos completos que contenham estas informações. Apenas as informações solicitadas pelo usuário ou relevantes para o processamento da consulta devem ser extraídas, como ocorre em consultas aos bancos de dados convencionais.

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

Mediadores e *wrappers* são importantes neste contexto, uma vez que atuam como módulos intermediários entre aplicações e fontes de dados, sendo responsáveis pelas transformações dos dados. Mediadores são módulos que integram conhecimento presente em fontes

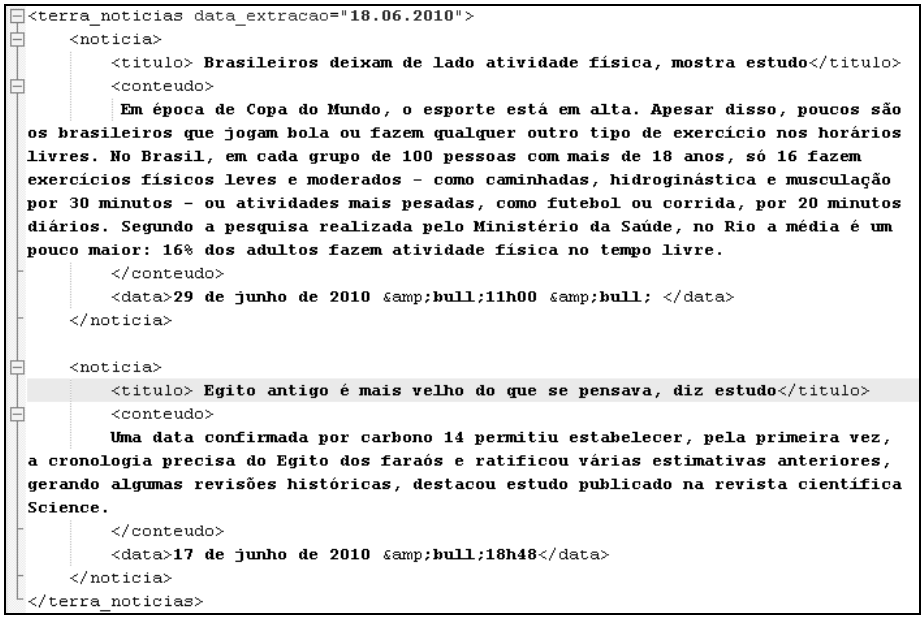
heterogêneas de dados, produzindo informação útil para camadas superiores de um sistema de gerenciamento de dados (WIEDERHOLD, 1992). Um serviço de mediação apresenta uma regra para transformação de dados e contém estruturas de conhecimento que guiam estas transformações. Uma ampla gama de funcionalidades pode estar associada a um mediador, como reorganização lógica de dados, aplicação de métodos para obtenção de informações derivadas e aplicação de padrões de apresentação de dados.

Os *wrappers* têm a função de encapsular o acesso a uma fonte de dados, tornando-a utilizável de uma maneira mais conveniente que a sua representação original. Os *wrappers* possuem características diferentes de um mediador. Um mediador integra dados de várias fontes de dados, enquanto um *wrapper* provê uma interface simplificada para uma única fonte de dados ou mais de uma fonte que tenham uma interface comum de acesso. Um *wrapper* pode ainda adicionar funcionalidade a uma fonte de dados, como por exemplo, prover uma camada de acesso para a um conjunto de documentos, para facilitar a manipulação dos dados (THIRAN, 2001).

Neste trabalho, para a extração dos dados do ambiênte Web, foi utilizada a biblioteca Java disponibilizada pelo projeto *open source* Web-Harvest (http://web-harvest.sourceforge.net/). Web Harvest é uma ferramenta de extração de dados na Web desenvolvida em Java. A ferramenta oferece um caminho para coletar dados de uma página Web e extrair as informações desejadas. Para isso, usa tecnologias para manipulação de Texto e XML como: XSLT, XQuery, XPath e Expressões regulars.

Nesta ferramenta, cada processo de extração é definido por um arquivo de configuração XML. Este arquivo descreve uma sequência de processos de extração, os quais executam tarefas específicas de forma a obter um dado objetivo, baseado na análise feita onde se deseja obter os dados (Capítulo 2).

O resultado do processo de extração é um arquivo XML com as informações desejadas de forma estruturada, onde é possível perceber na figura 3 que o objeto principal “noticia” possui o sub-elementos “titulo”, “conteudo” e “data da noticia”.



```
<terra_noticias data_extracao="18.06.2010">
    <noticia>
        <titulo> Brasileiros deixam de lado atividade física, mostra estudo</titulo>
        <conteudo>
         Em época de Copa do Mundo, o esporte está em alta. Apesar disso, poucos são
os brasileiros que jogam bola ou fazem qualquer outro tipo de exercício nos horários
livres. No Brasil, em cada grupo de 100 pessoas com mais de 18 anos, só 16 fazem
exercícios físicos leves e moderados - como caminhadas, hidroginástica e musculação
por 30 minutos - ou atividades mais pesadas, como futebol ou corrida, por 20 minutos
diários. Segundo a pesquisa realizada pelo Ministério da Saúde, no Rio a média é um
pouco maior: 16% dos adultos fazem atividade física no tempo livre.
        </conteudo>
        <data>29 de junho de 2010 &amp;bull;11h00 &amp;bull; </data>
    </noticia>

    <noticia>
        <titulo> Egito antigo é mais velho do que se pensava, diz estudo</titulo>
        <conteudo>
        Uma data confirmada por carbono 14 permitiu estabelecer, pela primeira vez,
a cronologia precisa do Egito dos faraós e ratificou várias estimativas anteriores,
gerando algumas revisões históricas, destacou estudo publicado na revista científica
Science.
        </conteudo>
        <data>17 de junho de 2010 &amp;bull;18h48</data>
    </noticia>
</terra_noticias>
```

Figura 20. Arquivo com as notícias de forma estruturada.

A próxima etapa que o protótipo realiza é a comparação das informações extraídas com as informações que o usuário deseja encontrar. Para a leitura e identificação do documento XML gerado pela aplicação, foi utilizada a biblioteca Java XSTREAM, que possibilita a manipulação e transformação do conteúdo XML em Objetos Java, que neste caso se chama “noticia”, e possui os seguintes atributos: Título, Conteúdo e Data.

A partir deste momento, recuperado o conteúdo do arquivo XML, o protótipo desenvolvido pode realizar a comparação utilizando cálculos de distância baseado em um algoritmo de similaridade.

4. Algoritmos de Similaridade

Uma função de similaridade é um algoritmo que compara dois ou mais itens, e realiza comparações e cálculos para medir numericamente a semelhança existente entre eles. A maioria das funções de similaridade recebe como entrada dois itens e retorna como resultado um valor real normalizado no intervalo [0, 1], onde 0 representa a completa desigualdade entre as entradas e 1 significa que os itens são equivalentes para a função.

O algoritmo utilizado no protótipo desenvolvido é o editDistance, suas características e funcionamento são descritas no Capítulo 4.1.

4.1. Algoritmo editDistance

O algoritmo para cálculo de distância chamado editDistance compara duas cadeias de caracteres, e determina o número mínimo de operações necessárias para transformar uma cadeia na outra. As operações realizadas nesse cálculo são: inserção, remoção e substituição de um caractere. A partir do resultado das operações é possível calcular-se um valor de similaridade entre as cadeias de caracteres.

O algoritmo para calcular a distância entre uma cadeia de caracteres A e uma cadeia de caracteres B, respectivamente com i e j caracteres, utilizando apenas as operações, é apresentado na figura 4.

Inicialmente, a matriz M é criada no passo 2 e tem a sua primeira linha e primeira coluna inicializada no passo 3. Os valores associados às demais células são calculados pelo mínimo entre três valores, que representam o custo das três operações. Os valores calculados pelos passos 4(a) e 4(b) são responsáveis por computar os custos das operações de inclusão e remoção de caracteres, enquanto que o valor calculado pelo passo 4(c) computa o custo da operação de substituição de caracteres. O cálculo dessa última operação é auxiliado pela função sub(x, y) que retorna 0, se x = y, e 1, caso contrário.

1. Sejam A e B duas cadeias de caracteres tais que |A|= i e |B|= j
2. Criar uma matriz M com $i + 1$ linhas e $j + 1$ colunas
3. Inicializar a primeira linha e a primeira coluna
   (a) Primeira linha $M[0, q] = q$, para $0 \leq q \leq j$
   (b) Primeira coluna. $M[p, 0] = p$, para $1 \leq p \leq i$
4. Calcular o valor associado às demais células $M[p, q]$ da matriz através do mínimo entre:
   (a) A posição acima mais um. $(M[p - 1, q] + 1)$
   (b) A posição à esquerda mais um. $(M[p, q - 1] + 1)$
   (c) A posição na diagonal esquerda superior mais o custo de substituição do caractere $A[p]$ por $B[q]$
   $(M[p - 1, q - 1] + sub(A[p], B[q]))$

Figura 21. Passos para cálculo da similaridade. Fonte: VINSON, 2007.

Ao final da execução do passo 4, a matriz está completamente preenchida. Cada célula da matriz M[x, y] armazena o número mínimo de operações necessárias para a seqüência dos x primeiros caracteres de A transformar-se na sequência dos y primeiros caracteres de B, ou vice-versa. Portanto, o valor contido na célula M[i, j], reflete o número mínimo de operações necessárias para a cadeia A transformar-se na cadeia B.

Após obter o número mínimo de operações é necessário calcular a medida da similaridade. A fórmula a seguir (figura 5) calcula a similaridade final pela normalização do número de operações que não são necessárias para efetuar a transformação (máximo possível menos o mínimo necessário) no intervalo [0, 1]. A normalização é feita com divisão do número de operações que não são necessárias pelo máximo número de operações possíveis.

$$editDistance(A, B) = \frac{Max(i, j) - M[i, j]}{Max(i, j)}$$

Figura 22. Fórmula para cálculo da distância. Fonte: VINSON, 2007.

A utilização de um algoritmo de similaridade permite ao usuário encontrar os dados que são relevantes às suas necessidades. Tendo em vista este ponto e encontrados os dados que são relevantes, é importante armazená-los de forma que possam ser recuperados em outro momento. Para isso, é necessário possuir um SGBD que tenha capacidade de armazenar dados complexos. Neste escopo, para armazenamento de dados não estruturados os SGBDOR são indicados, pois utilizam os benefícios dos bancos tradicionais e os benefícios da orientação a objetos.

5. Armazenamento das Informações

A maior parte dos sistemas de banco de dados desenvolvidos até hoje foram os relacionais (SGBDR). Porém, com o surgimento de sistemas mais avançados como: banco de dados hipertexto, banco de dados multimídia, sistemas de automação, entre outros, houve a necessidade de assimilar estruturas que suportassem estes sistemas. Desta forma, o padrão para bancos de dados se lançou para a tecnologia objeto relacional e orientados a objetos, onde, os SGBDOR oferecem suporte aos tradicionais modelos relacionais com estrutura de orientação à objetos e os SGBDOO se caracterizam principalmente pela herança existente entre as entidades. Atualmente, não é possível a existência de um só padrão de mercado pela natureza das aplicações que ambos estão destinados (BOND et al, 2005).

Neste contexto surge o paradigma objeto-relacional, que veio com a proposta de suprir as carências do modelo relacional, proporcionando aos usuários toda a naturalidade para modelar objetos complexos e suas características peculiares que os banco de dados orientados a objetos propunham, e ao mesmo tempo, não abrindo mão de toda a carga tecnológica adquirida para os Bancos Relacionais em anos de desenvolvimento (HOLMES; SCHILDT, 2003).

Com o objetivo de recuperar as informações que o usuário necessita, de forma rápida e eficiente, o protótipo possui a funcionalidade de armazenar o conteúdo extraído, neste caso a notícia, em um banco de dados. Para realização deste propósito, o sistema desenvolvido utiliza o SGBDOR PostgreSQL 8.4.

De posse de um SGBDOR completo e funcional, e também dos dados extraídos e classificados de acordo com a similaridade e o texto digitado pelo usuário, o sistema possibilita o armazenamento destas informações.

Com o objetivo de dinamizar o sistema e diminuir o acoplamento entre classes do sistema e a conexão com o banco de dados, este protótipo utiliza o Hibernate. O Hibernate é um framework para o mapeamento objeto-relacional escrito na linguagem Java, este framework facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de arquivos (XML) para estabelecer esta relação.

O objetivo deste framework é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do modelo relacional. Em especial, no desenvolvimento de consultas e atualizações dos dados.

A partir do momento que o *hibernate* é utilizado, pode-se realizar o armazenamento dos dados. Desta forma, em outro momento, o sistema possibilita consultar estes dados utilizando a linguagem HQL para recuperar os objetos do banco de dados. De posse dos objetos recuperados é possível retornar ao usuário àquela informação que ele solicitou, de acordo com a palavra digitada e a similaridade mínima escolhida no protótipo.

6. Conclusão

Neste trabalho foi possível compreender o funcionamento dos métodos de extração de dados, em especial a extração de dados na WEB, e entender as características envolvidas no processo, a partir de testes feitos com as ferramentas disponibilizadas em projetos open-source. Além de ter sido validado e entendida a fundamentação teórica do mesmo, a partir da pesquisa e implementação desta etapa no protótipo.

Outro grande propósito atingido, foi a compreensão de como é realizado o cálculo de similaridade entre palavras, utilizando o algoritmo editDistance. Com o estudo feito acerca dos algoritmos de similaridade, foi possível entender, na teoria e na prática, cada passo para montar uma matriz, fazer a comparação entre as palavras e aplicar a fórmula para a obtenção do coeficiente de similaridade.

Sem menos importância, a pesquisa feita sobre banco de dados e mapeamentos Objeto-Relacional proporcionou o desenvolvimento do protótipo que pode ser flexível a qualquer banco de dados suportado pelo framework *Hibernate,* bastando a configuração em uma única classe. Além disso, proporcionou melhor entendimento sobre orientação a objetos, bem como suas características e aplicações na prática.

Uma grande dificuldade encontrada durante a elaboração deste trabalho, foi a escassez de livros sobre dados não estruturados, sendo que esta dificuldade foi, em sua parcialidade, suprida com artigos, publicações e monografias que continham bastante referência sobre dados semi-estruturados.

Outra grande dificuldade foi encontrada na implementação, devido as freqüentes mudanças do ambiente WEB, como exemplo pode se citar o site www.globo.com: O protótipo estava sendo desenvolvido para se adequar às noticias do site Globo.com, porém após a feita configuração para extração dos dados e toda a documentação da implementação, o site foi totalmente remodelado, de forma que todo o trabalho teve que ser desfeito. Após isto acontecer, o site www.terra.com.br foi o escolhido para exemplificar o funcionamento do protótipo. Outro fator que dificultou o desenvolvimento do trabalho foi a falta de padronização no desenvolvimento Web.

De forma geral, pode-se concluir que o ambiente Web dispõe de um imenso conteúdo de dados não estruturados. Desta forma, o estudo e a implementação do protótipo desenvolvido contribuiu para o entendimento do processo como um todo, desde a extração de uma informação contida na web, à comparação realizada por um algoritmo de similaridade, até o armazenamento da informação de forma estruturada em um banco de dados.

References

ABITEBOUL, S.; BUNEMAN, P.; SUCIU, D. Data On the Web – from relations to semistructured data and XML. San Francisco: Morgan Kaufman, 2000. 257 p.

BUNEMAN, Peter. Semistructured Data. In: SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS (ACM), 16, 1997. Tutorial in Proceeding, 1997. p. 5.

MELLO, Ronaldo; DORNELES, Carina; KADE, Adrovane; BRAGANHOLO, Vanessa; HEUSER, Carlos. Dados Semi-Estruturados. In: Simpósio brasileiro de banco de dados. 2000, João Pessoa - PB, 2000.

PELLEGRINO, Grazieno et al. XML e Banco de Dados para WEB. Instituto de Matemática – Universidade Federal da Bahia. 2008.

SUCIU, D. Management of Semistructured Data. SIGMOD Record, 4ª ed., p.4-7, Dezembro, 1997.

THIRAN, Hainaut. Wrapper development for legacy data reuse. Reverse Engineering, 2001. p. 198-207.

VINSON, Alexander Richard. PathSim: um algoritmo para calcular a similaridade entre caminhos XML. Porto Alegre, 2007.

WIEDERHOLD, G. Mediators in the Architecture of Future Information Systems. Computer, v. 25,n.3, Março, 1992, p.38-49.