UNIVERSIDADE FEDERAL DE SANTA CATARINA

Projeto de uma Base de Dados Terminológica

Fabrício Santos da Silva

Florianópolis - SC
2005/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Projeto de uma Base de Dados Terminológica

Fabrício Santos da Silva

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Florianópolis - SC
2005/1

Fabrício Santos da Silva

Projeto de uma Base de Dados Terminológica

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Orientador: _____
Prof. Doutor Ronaldo dos Santos Mello

Banca examinadora:

_____
Prof. Doutor José Eduardo De Lucca

_____
Prof. Doutor Paulo José Ogliari

AGRADECIMENTOS

Um agradecimento ao professor Ronaldo, pela orientação deste trabalho, conselhos, pela paciência demonstrada e pela confiança que teve.

Agradeço a Professora Lígia, por sanar as minhas dúvidas no campo da terminologia. Um muito obrigado a Maristela que me ajudou com a formatação deste trabalho e pela amizade.

Gostaria de deixar um muito obrigado a todos os amigos que fiz durante a minha passada nestes anos de curso e da minha vida. Em especial ao Wellington pela ajuda no início do curso e pela camaradagem. Para a Daniela, pela grande amiga que é.

Um agradecimento especial para a minha família, por tudo.

RESUMO

A Internet proporciona a troca de qualquer tipo de informação, atualmente o intercambio de dados em documentos XML está crescendo. XML estão se tornando um padrão para a troca de informações. Devido as características fornecidas pelos documentos XML, novos mecanismos de busca estão sendo propostos. Estes trabalhos focam a integração dos esquemas XML, para extrair a informação. Esta abordagem acaba por esbarrar na variação lingüística presente na definição estrutural destes documentos. Então é necessária uma ferramenta auxiliar, que indique qual caminho a ferramenta de integração deve tomar. Este trabalho propõe a criação e construção de uma base de dados terminológica, para dar este suporte aos mecanismos de integração, durante o processo de validação semântica entre os termos. Este sistema permite que o conhecimento possa ser acessado por usuários em geral, sendo que toda a informação é gerenciada por usuários com conhecimento sobre o domínio, seja para a integração de dados ou não.

Palavras-chave: XML, terminologia, banco de dados terminológicos, semântica

ABSTRACT

The Internet provides the exchange of any type of information, currently interchanges of data in XML documents are growing. XML see if becoming a standard for the exchange of information. Because of the characteristics supplied by XML documents, new search engines are being considered. These works focus the integration of XML schema to extract the information. This boarding finishes for stopped in the limitations of variation linguistic present in the structural definition of this documents, So is necessary a tool to help and indicates which way the integration tool must take. This work considers the creation and construction of a terminological database, to give this support to the integration mechanisms, during the validation process semantics between the terms. This system allows that the knowledge can be accessed by general users, being that all the information is managed by users with knowledge on the domain, either for the integration of data or not.

KEYWORD: XML, terminological, terminological databases, semantic

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ARTEMIS – Analysis and Reconciliation Toll Environment for Multiple Information Systems

ASID – Administrador de Sistemas de Integração de Dados

BDTerm – Banco de Dados Terminológicos

BInXS – *Bottom-up Integration of XML Schemata*

DER – Diagrama Entidade Relacionamento

DTD – Document Type Definition

SGBD – Sistema de Gerenciamento de Banco de Dados

SID – Sistema de Integração de Dados

UPC – Usuário Público Colaborador

UPG – Usuário Público Gerente

UPP – Usuário Público Participativo

1 Introdução

As principais conquistas tecnológicas do século XX se deram no campo da informação (TANENBAUM, 1999). Este crescimento é conseqüência do grande número de computadores espalhados pelo mundo, ao número de usuários conectados na WWW (*World Wide* Web ou simplesmente Web) e também a facilidade que as pessoas têm para publicar, compartilhar, ler documentos nos mais diversos formatos e idiomas, com conteúdos dos mais variados domínios.

O crescimento do conteúdo disponível para consulta acabou por gerar novos problemas, no campo de armazenamento, disponibilização e é claro, o principal, a recuperação da informação. Os recursos hoje disponíveis para buscar informações, as denominadas “máquinas ou sítios de busca” não são eficientes nem suficientes para indexar todo o conteúdo disponível na Internet relativo a uma consulta desejada (BOTELHO, 2002). Somente para exemplificar a dimensão do problema, atualmente existem sítios de busca dos mais variados, que procuram em documentos em geral (são as máquinas de busca mais conhecidas) ou somente para documentos científicos (teses, dissertações, publicações, revistas cientificas, etc) que são utilizados principalmente na comunidade científica, etc.

Assim, a recuperação da informação pode se tornar uma tarefa árdua ou, às vezes, até uma grande perda de tempo para o usuário, até que este consiga encontrar um documento que satisfaça as suas necessidades. Também deve ser levado em consideração que muitos sítios não podem ser considerados confiáveis, devido à omissão de informações que dariam ao texto uma conotação mais formal sobre o tema, como por exemplo, à falta do nome do(s) autor(es) do documento, a perda das referências bibliográficas e em muitos casos a sobreposição da informação através de uma prática comum de cópia de informações de outros sítios.

Os problemas acima citados necessitam de soluções mais especializadas, como a criação de mecanismos de busca mais específicos e esta abordagem pede um tratamento diferenciado na forma de como a informação deve ser tratada. Este é um dos contextos no qual as técnicas de integração de dados surgem como alternativa.

Grande parte do conteúdo que a Web disponibiliza apresenta se na forma de dados semi-estruturados (ABITEBOU *apud* SIEDLER, 2004), dados que separam a

informação da estrutura do documento, dados semi-estruturados são apresentados no capítulo dois. Existem diversas abordagens de extração do conteúdo destes dados, como *DOM* (*Document Object Model*) que é uma interface que permite que programas acessem dinamicamente o conteúdo e a estrutura de documentos XML (*eXtensible Markup Language*), bem como *XQuery* (*XML Query*), que provê facilidades para extrair informações de documentos na Web, permitindo que documentos XML sejam acessados de forma semelhante a um banco de dados (W3C, 2004), com métodos de busca, o que não resolve se a mesma informação tiver que ser localizada em diversos documentos, onde cada um possui uma estruturação diferente. Por este motivo, vêm sendo pesquisadas formas de integração de documentos semi-estruturados.

Muitos dos problemas encontrados na construção de sistemas de integração de dados na Web são similares aos problemas encontrados em sistemas de integração de bancos de dados (BACILI *apud* SIEDLER, 2004), levando em consideração o processo e métodos mais gerais, uma vez que dados semi-estruturados requerem um tratamento diferenciado em muitos pontos. Por exemplo, dados na Web são em geral semi-estruturados e quando se deseja integrar tais dados, alguns aspectos adicionais precisam ser analisados, tais como, o grande número de fontes de informação disponível, a falta de metadados sobre estas, o alto grau de autonomia das fontes e o nível elevado de irregularidade da estrutura dos dados (SIEDLER, 2004).

Devido a grande quantidade de fontes de dados existentes na Web, uma ferramenta de integração auxilia o usuário, retirando deste a necessidade de saber como a informação está representada em cada uma dessas fontes, ou seja, o sistema de integração funciona como um intermediário entre o usuário e as fontes de dados.

Motivado pelas diversas formas de representar e divulgar a informação através da Web, faz-se necessário definir um padrão para a representação dos dados. XML é um formato bastante utilizado para armazenar textos estruturados e semi-estruturados, pretendidos para disseminação e finalmente para publicação, tais como catálogos, formulários (BRADLEY, 2000). Para realizar o acesso a estas informações, aplicações na Web devem desenvolver *mecanismos de integração de dados* que permitam consultas cujo escopo de atuação seja um conjunto de fontes XML (MELLO, 2002).

Atualmente existem várias propostas para realizar a integração de esquemas XML, *MIX* (LUDÄSCHER, *apud* MELLO, 2002)*, Xyleme* (REYNAUD, *apud* MELLO, 2002)*, LSD* (DOAN; DOMINGOS e HALEVY, *apud* MELLO, 2002)*, DIXSE* (RODRIGUES-GIANOLLI e MYLOPOULOS, *apud* MELLO, 2002) e BInXS (MELLO,2002). A proposta na qual este trabalho está inserido é a proposta BInXS, que possui as seguintes características (MELLO, 2002):

- utilização de uma representação conceitual para esquemas XML, com relacionamentos de associação e herança que modelam a intenção semântica dos dados XML. Este esquema conceitual é obtido através de um processo de conversão do esquema XML que analisa detalhadamente o modelo de dados XML;
- um processo de integração semântica destes esquemas conceituais que considera a determinação de equivalências e a resolução de conflitos entre representações semi-estruturadas;
  - em particular, são tratados casos específicos na unificação de dois tipos de elementos (texto e estruturados) e elementos com representações alternativas. Um esquema conceitual global é o resultado deste processo, funcionando como mediador entre o usuário e fontes de dados XML na Web.

O presente trabalho está relacionado ao suporte necessário à fase de integração semântica do BInXS, tendo por objetivo prover um maior suporte automático a esta fase no que se refere à determinação de equivalências semânticas entre conceitos de um mesmo domínio que estão em fontes de dados XML diferentes. Este trabalho pretende ser uma alternativa ao sistema *ARTEMIS* (*Analysis and Reconciliation Tool Environment for Multiple Information Systems*), que atualmente é o sistema que fornece esse suporte ao BInXS, ver capítulo quatro.

Especificamente, este trabalho propõe a definição de uma base de dados terminológica, bem como formas de interação com esta base, para auxiliar o BInXS no processo de integração semântica, onde os termos a serem validados sobre o aspecto de sua intenção semântica (equivalência e ou de hierarquia), estão sob um mesmo domínio, diminuindo a tarefa de um usuário especialista para resolver pequenos conflitos. Estes deixaram de ocorrer pelo fato de que a relação semântica entre os dois termos passa a estar vinculada ao domínio, eliminando-se assim o fato

de que o usuário especialista do BInXS teria para determinar entre as respostas, a que abrangia a sua necessidade.

Para diminuir a tarefa de inserção do conteúdo informativo da base, foi estendida a sua funcionalidade, ou seja, ela pode ser consultada e administrada por usuários que não tem envolvimento direto com as atividades de integração de dados. Desta forma, esta base passa também a ser um repositório para consulta de termos. Adicionalmente, existe ainda a possibilidade de usuários responsáveis pelos mecanismos de integração de dados inserirem termos específicos das fontes de dados que eles manipulam, aumentando assim o poder expressivo de consulta da ferramenta.

Este trabalho está dividido em outros quatro capítulos. O capítulo dois irá tratar sobre conceitos básicos de dados semi-estruturados e XML e o motivo pelo qual a linguagem XML é empregada para representação destes dados. O capítulo três apresenta uma definição de banco de dados terminológicos e os conceitos que estão envolvidos na sua definição, incluindo ontologias e *thesauri*. No capítulo quatro será apresentado um resumo sobre o BInXS, o processo de integração semântica e os motivos pelo qual um banco de dados terminológico será desenvolvido. O capítulo cinco apresenta o sistema proposto, com a conceituação sobre o problema, modelagens geradas, funcionalidades e interface. O último capítulo apresenta a conclusão sobre este trabalho, considerações e propostas para atividades futuras.

2 Dados Semi-Estruturados e XML

Dados semi-estruturados são dados não convencionais cuja representação pode ser altamente heterogênea mesmo para instâncias em uma mesma fonte de dados (MELLO, 2002).

Dados semi-estruturados não são restritamente tipados como em banco de dados e nem são sem estrutura (ABITEBOUL, 2004) e carregam informação sobre o seu esquema sendo referenciados como dados *autodescritivos*. Porém nem sempre um esquema externo está explicitamente associado ao dado. Desta forma, mecanismos de buscas tradicionais não têm um bom desempenho ao tratarem com definições semi-estruturadas (NEELY, 2004).

Dados semi-estruturados podem ser caracterizados por (Abiteboul *apud* MELLO et. al., 2000):

- Estrutura irregular – coleções extensas de dados semanticamente similares estão organizados de maneiras diferentes. Em suma, não existe um esquema padrão para esses dados;
- Estrutura implícita – a estrutura básica de representação dos dados pode estar de forma implícita, sendo necessário realizar uma computação para obter essa estrutura;
- Estrutura parcial – apenas parte dos dados disponíveis pode ter alguma estrutura associada, seja implícita ou explícita;
- Estrutura extensa – a ordem de magnitude de uma estrutura para estes dados é grande, uma vez que os mesmos são muito heterogêneos;
- Estrutura evolucionária – a estrutura dos dados modifica-se tão freqüentemente quanto os seus valores. Dados Web apresentam este comportamento, uma vez que existe o interesse em manter estes dados sempre atualizados;
- Estrutura descritiva e não prescritiva – dada à natureza irregular e evolucionária, 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;

- 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 (*tags* dentro de *tags*).

XML é um formato bastante utilizado atualmente para armazenar dados na forma estruturada ou semi-estrutrurada, com a intenção de disseminar e publicar, estes dados, em diversos formatos de mídia (BRADLEY, 2000). Assim uma mesma informação contida em um dado possa ser visualizada em um editor de texto ou em um navegador Web, garantido desta forma que o conteúdo não está preso a visualização. Na Web, a maior parte dos documentos veiculados podem ser compostos por dados semi-estruturados ou por dados sem estrutura alguma. Neste último caso a maior parte da informação disponível na Web no formato HTML (*HyperText Markup Language*).

Os documentos XML estão divididos em duas partes. Uma lógica, que permite que o documento seja representado por um elemento, definido pelos metadados (palavra que representa a informação) e que podem conter uma informação associada ou não (um elemento pode ser vazio). A formação de um elemento pode ser feita através de atributos, estes carregam uma informação relevante a estrutura do documento, e não sobre a informação a ser visualizada, como exemplificado na Figura 2.1, onde o atributo id é uma característica de “dados” e não da informação que este carrega. O elemento também pode ser definido pela inclusão de outros elementos, na Figura 2.1 o elemento “dado” é composto pelos sub-elementos “nome” e “contato”, desta forma a informação que “dado” carrega é o conjunto de informações que compõe os seus sub-elementos.

A segunda parte, chamada de física, permite que os componentes do documento, as *entidades*, sejam nomeadas e armazenadas separadamente (BRADLEY, 2000). Na Figura 2.1, é apresentado um exemplo da parte lógica, onde o dado XML é a informação compreendida entre as *tags* iniciais e finais <dados> e </dados> respectivamente. A informação contida em dados, está sub-definida nos elementos nome (elemento com um atributo direto) e contato (composto pelos dois sub-elementos: e-mail e telefone).

XML atende todos os pontos anteriormente citados, sobre as características de dados semi-estruturados. Podendo também representar dados estruturados. Em ambos os casos pode existir uma DTD (Document Type Definition), ou uma DTD

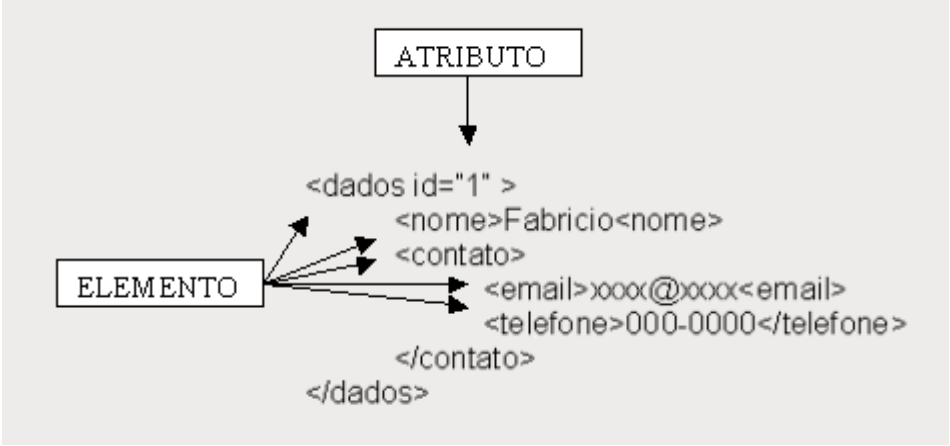

Figura 2.1 Representação de um dado simples em XML

pode ser associada a um dado XML quando esta não pré-definida. Uma DTD, pode ser vista como as especificações das regras de construção de elementos, por exemplo pode se associar uma DTD ao elemento “nome”, não permitindo que seja inserido caracteres numéricos. As DTDs possuem regras próprias para a sua construção o que não será apresentado aqui, pois foge ao escopo deste trabalho.

3 Banco de Dados Terminológicos e Conceitos Relacionados

Banco de dados terminológicos ou simplesmente BDTerm, são uma forma de armazenar o conhecimento terminológico em um de banco de dados com uma modelagem voltada às necessidades de um domínio específico (este domínio pode ser o conhecimento de Ciências da Computação ou o conhecimento sobre banco de dados, não havendo uma definição sobre especificidade). Ele utiliza os conceitos de classificação do léxico, porém através de um subconjunto - a Terminologia (BIDERMAN, 2001).

Atualmente existem várias aplicações que utilizam os conceitos e soluções propostas pelo conhecimento sobre o léxico para abordar um determinado tipo de problema, como o uso de ontologias, *thesaurus*, banco de dados léxicos, banco de dados temáticos e banco de dados terminológicos.

Pelas possibilidades de se usar o conhecimento léxico e pela subjetividade sobre como os domínios são representados, nem sempre conceitos corretos são empregados por pesquisadores no campo da informática. Esse processo é chamado como a "redescoberta da roda" por (LANCASTER, 2004), que tem trabalhos no campo de recuperação da informação. Em particular ele cita que "cientistas da computação que escrevem sobre recuperação da informação parecem só reconhecer e citar outros cientistas da computação. Processo também observado por Holmes citado por (LANCASTER, 2004), um cientista da computação, que diz que o que pensamos ser inovações são muitas vezes são meras repetições de idéias que datam de 40 ou 60 anos, e também complementa que para a nossa profissão se desenvolver de modo mais rápido e melhor, devemos construir sobre os alicerces do passado ao invés de ignorá-lo. Também há uma outra referência feita sobre este problema de definição de termos "Santini, outro cientista da computação, diz que o uso incorreto da linguagem em informática ameaça levar nossa profissão a se isolar da sociedade e tornar incompreensíveis nossas realizações" (LANCASTER, 2004).

Na seqüência é definido com mais detalhes o conceito de banco de dados terminológicos e outros conceitos relacionados (ontologias e *thesauri*). Para que fique de forma clara a diferença entre os três conceitos e que mais adiante possa ser

entendido os motivos da abordagem adotada.

3.1 Ontologias

Ontologia provê critérios para distinguir os tipos de objetos (concretos e abstratos, existência e não existência, real e ideal, independente ou dependente) e suas equivalências (relações, dependências e predicados) (CORAZZON, 2004).
No campo do compartilhamento do conhecimento, ontologia pode significar uma especificação explícita de uma conceituação (GRUBER, 2004). Já segundo (SOWA, 2000) ontologia é um catálogo de tipos de coisas que existem em um domínio de interesse ***D*** sobre a perspectiva de uma pessoa que utiliza uma linguagem ***L*** com o objetivo de falar sobre ***D***. Os tipos em ontologia representam os predicados, significado das palavras ou conceitos e tipos de relações.

Novamente, (LANCASTER, 2003) chama a atenção para o emprego incorreto do significado da palavra ontologia, como sendo uma equivalência para *classificação* ou *classificação hierárquica,* como é mais freqüentemente referenciada.

Na área de Ciência da Computação, ontologia vem sendo empregada com o significado de formular de forma exaustiva e rigorosa um esquema conceitual sobre um dado domínio (WIKIPEDIA, 2004). O foco principal é a capacidade para definir primitivas reais ou elementos básicos do conhecimento associado ao domínio (SAINT-DIZIER e VIEGAS, 1995). Uma hierarquia pode ser empregada na construção de uma ontologia, permitindo que uma nova ontologia que herde conceitos e definições de uma ontologia já existente, algo semelhante a herança de classes na programação orientada a objetos. Mas deve ficar claro que mesmo existindo esta possibilidade, as ontologias criadas serão distintas, uma vez que os domínios são diferentes e que a hierarquia é sobre o domínio e não sobre os termos que o compõe.

As ontologias são compostas, da seguinte forma (SAINT-DIZIER e VIEGAS, 1995):

- conjunto de entidades, geralmente estruturadas e tipadas;
- conjunto de relações e operações e propriedades relacionadas;
- um conjunto finito de predicados, descrevendo estados primitivos e ações do

domínio;

- um conjunto de funções que atuam sobre as entidades.

Atualmente, ontologia pode ser considerada como um campo complexo e multidisciplinar que remete ao conhecimento da organização da informação, ganhando popularidade e se tornando uma tecnologia em expansão com grande potencial para melhorar a organização da informação, bem como gerência do conhecimento desta (DING e SCHUBERT, 2001). Ontologias estão ajudando pessoas e computadores a acessar a informação e melhorando a troca do conhecimento, fazendo que um sub-campo de pesquisa comece a emergir – a engenharia de ontologias, que tem como atribuições: construir ontologias para domínios específicos e manter atualizadas essas ontologias.

A construção de ontologias pode ser feita de forma manual, o que demanda tempo, grande trabalho e propensão ao erro; de forma semi-automática ou completamente automática, este último somente em casos restritos; bem como podem ser criadas do zero, a partir de ontologias já existentes, de corpus de fontes de informações, ou uma combinação desses dois últimos métodos (DING e SCHUBERT, 2001).

Existe uma grande variedade de linguagens para realizar a representação de ontologias como (SILVA, 2003):

- Ontolingua desenvolvida para dar suporte a projetos e especificações de ontologias com semântica lógica e clara;
- LOOM linguagem e aplicação para construção de aplicações inteligentes, baseado na lógica de descrição;
- OCML (*Operacional Conceptual Modeling Language*) linguagem baseada em *frames* que provê mecanismos para expressar termos como relacionamentos, funções, regras, classes e instâncias ;
- FLogic formalismo baseado em *frames* que descreve de modo claro e declarativo a maioria dos aspectos estruturais do paradigma de linguagens orientada a objetos e baseada em *frames*;
- OKBC (*Open Knowledge Base Connectiy*) protocolo de acesso a bases de conhecimento, não chegando a ser uma linguagem e sim um complemento a linguagens que dão suporte ao compartilhamento do conhecimento;
- XOL (*XML-Based Ontology Exchange Language*), linguagem baseada na

Web para a troca de definições ontológicas entre um conjunto de sistemas de *software* heterogêneo em um domínio;

- SHOE (*Simple HTML Ontology Extentions*) extensão do HTML que permite adicionar conhecimento semântico legível para sistemas computacionais;
- OML/CKML (*Ontology Markup Language/Conceptual Knowledge Markup Language*) onde OML representa estruturas esquemáticas e ontológicas, enquanto que CKML provê um arcabouço do conhecimento conceitual;
- OIL combina representação e inferência para ontologias baseadas na Web, combinando primitivas de modelagem de linguagens baseada em *frames* com serviços de semântica formal e raciocínio provido pela lógica de descrição.

Assim como a grande variedade de linguagens que dão suporte a escrita e construção de ontologias, também existe uma série de ferramentas para o auxilio a construção. Estas ferramentas minimizam os esforços e permitem criar ontologias do zero ou a partir de ontologias reusáveis, incluindo documentação importação e exportação de ontologias de diferentes formatos, visualização gráfica e mecanismos de inferência. Exemplos destas ferramentas são: Apollo, LinkFactory, Ontolingua, Protégé-2000 (SILVA, 2003).

3.2 Thesaurus

Thesaurus vem do latim “treasure” e significa uma estrutura similar a um dicionário, que ao invés de armazenar definições, fornece sinônimos e antônimos para as palavras (KOWALSKI E MAYBUNY, 2000). Atualmente um *thesaurus* pode armazenar mais do que simples relações de equivalência ou oposição entre os termos, mantendo estruturas de armazenamento de informações úteis nos dias de hoje.

Thesaurus são listas de termos interligados semanticamente entre si (CABRÉ, 1993), que pode ser considerado um vocabulário dinâmico de termos relacionados semanticamente e genericamente, de um domínio do conhecimento em comum (ISO, 1982).

Pelas duas definições acima citadas, pode-se perceber que, assim como

ontologias, *thesauri* estão intimamente ligadas a um domínio específico.

A construção básica de um *thesaurus* envolve três características (CABRÉ, 1993):

- conteúdo - relação semântica entre as palavras, que podem ser sinonímias, hiponímias (palavras que são mais específicas que outras, como por exemplo, camundongo é hiponimo de roedor) ou associação;
- estrutura - documento formalizado composto por um vocabulário controlado e dinâmico e de relações conceituais expressas por regras formalizadas;
- função – são instrumentos de controle, que regulam o uso da linguagem natural.

A construção de um *thesaurus* não é uma tarefa simples. Devido a importância sobre o assunto existe uma recomendação, ISO 2788-1974 (ISO, 1982), de 1974, que serve como guia para a construção de *thesauri* monolingue. Por ser uma sugestão, serão apresentados os principais pontos, para que os conceitos sobre o funcionamento básico dos *thesauri* possam ser melhor compreendidos.

Os *thesauri* podem ter dois tipos de controle, um onde somente uma palavra pode ter a função de referência ao conceito denotado e o outro onde várias palavras podem denotar o mesmo conceito. Em ambos os casos estes termos de referência são chamados *descritores* (ISO, 1982), para melhor compreensão do melhor caso, quando se faz consulta ao índice de um livro, buscando as páginas ao termo “X” e ao invés de encontrar as páginas desejadas sobre este termo, aparece a expressão *ver “Y”.*

Os descritores podem ser formados por uma ou mais palavras, desde que o conceito representado seja um só e que faça parte do domínio desejado. Também deve-se ter alguns cuidados para com a forma como as palavras que compõem os descritores sejam grafadas, como a forma mais comum de escrita deve ser adotada (ISO, 1982), como no caso de farmácia e pharmácia (considerando que esta forma ainda fosse utilizada, ou que possa haver referências a documentos antigos) o descritor deve ser farmácia e o pharmácia faria referência ao descritor e não ao conceito. O mesmo deve ocorrer para termos que possuam equivalente em outra língua e este termo “estrangeiro”, seja comumente utilizado (ISO, 1982), o que é muito comum no campo da informática.

Os termos que compõem os descritores devem estar preferencialmente na forma substantiva. Quanto ao singular e o plural fica a critério do modelo. O mesmo

critério é aplicado a abreviaturas e acrônimos (ISO, 1982). Se houver homônimos em um mesmo domínio, pode ser acrescentado um qualificador entre parênteses, desta forma o descritor será formado pelo termo e pelo qualificador (ISO, 1982).

A inter-relação entre os descritores é realizada através da formação de uma rede de relações, seja hierárquica e/ou de equivalência e/ou associativa (ISO, 1982):

- equivalência:
  - sinônimos;
  - quasi-sinonimos ;
  - como sentido de *USO*, *USO PARA, USO PARA COMBINAÇÃO.*
- hierárquica:
  - relação genérica – relação entre termos superordinados (genérico) e subordinados (específico);
  - parte de um todo;
- associativa: utilizada quando a relação não se encaixa nem como de equivalência e nem como hierárquica

Desta forma *thesauri* possuem características de uma ontologia (sobre um domínio), mas com restrições e definições próprias (regras de escrita e relacionamento entre as palavras) que em termos de aplicação podem constituir ferramentas muito mais poderás.

3.3 Banco de Dados Terminológicos (BDTerm)

Antes de se dar início aos conceitos técnicos relacionados a banco de dados terminológicos, é necessário fazer uma pequena introdução sobre os conceitos que envolvem a palavra “termo”, bem como a ciência que os estuda – a Terminologia.

Desde tempos remotos os homens criam e utilizam palavras para expressar e denominar conceitos, objetos e processos nos diferentes campos do conhecimento especializado (KRIEGER e FINATTO, 2004). No campo de estudos teórico e aplicado dedicado aos termos técnico-científicos, a Terminologia compreende também uma face aplicada, sobretudo, à produção de glossários, dicionários técnico-científicos e banco de dados terminológicos.

A constituição de uma terminologia própria marca, em toda ciência, o advento ou o desenvolvimento de uma conceituação nova, assinalando assim um momento decisivo de sua história (Benveniste *apud* KRIEGER e FINATTO, 2004). O vocabulário próprio de uma ciência é o que a diferencia das demais, dando a esta uma característica de estar em contínuo desenvolvimento. Isto é o que Benvensite (KRIEGER e FINATTO, 2004) afirma, quando comenta que uma ciência só consegue se impor quando impõe os seus conceitos através de suas denominações. Esta necessidade que cada campo do conhecimento tem de caracterizar e referenciar os seus conceitos gera uma grande quantidade de informações e necessidades especiais de como tratá-las.

O léxico temático configura-se como um componente a serviço da comunicação especializada, uma vez que os termos transmitem conteúdos próprios de cada área, um valor monossêmico e monorreferencial, porque estabelecem o significado específico de cada área (KRIEGER e FINATTO, 2004). É comum imaginar situações para área cientificas distintas, como por exemplo biologia e computação ao fazerem referência ao termo “herança”, que em biologia pode estar associada ao material genético de um ser vivo e para a computação pode estar associado a programação orientada a objetos. Mas deve ficar claro que este comunicação especializada pode estar subdivida em subáreas de uma ciência, como em banco de dados e redes de Petry, onde o termo “cardinalidade” tem definições diferentes.

No que diz respeito ao campo da ciência da computação a relação com a terminologia está seguindo dois caminhos distintos, um onde a ciência da computação auxilia e muda as atividades terminológicas e metodologia. Outro, onde a terminologia ajuda na pesquisa lingüística computacional (CABRÉ, 1993). Como exemplo para o primeiro caso, a construção de banco de dados voltados para a terminologia, dá a está ciência as facilidades desejadas para que esta continue em crescimento. No segundo caso o campo de Inteligência Artificial, um dos primeiros campos da computação que vem se beneficiando e utilizando o conhecimento gerado pela Terminologia.

Bancos de dados terminológicos apresentam-se como um sistema de informações interconectados, composto por uma base principal, onde está a lista de termos, e outras bases auxiliares de caráter suplementar, conectadas por meio de informações (KRIEGER e FINATTO, 2004). As bases auxiliares, tais como:

bibliográficas e documentais; temáticas (thesuarus); textuais e conhecimento (CABRÉ, 1993), dão a Terminologia uma certa liberdade, para não ter que prever e tratar todas as formas de informações. Pode ser observado que a terminologia é um campo multidisciplinar e de certa forma um tanto quanto “discreto”, onde as suas características presentes em muitos campos de aplicação fogem aos olhos do público em geral.

Bancos de dados terminológicos foram inicialmente concebidos a serviço da tradução, formados por bancos de dados bilíngües ou plurilíngües, mas a possibilidade de armazenar uma grande quantidade de termos e suas respectivas informações, bem como manter este conhecimento atualizado e de fácil divulgação aos usuários é que estão transformando estas bases em importantes ferramentas para diversos profissionais (CABRÉ, 1993).

Na década de 1970, surgiram os primeiros bancos de dados terminológicos, permitindo assim o tratamento automatizado de termos técnico-cientificos e armazenamento de quantidades de informações, antes não imaginadas (KRIEGER e FINATTO, 2004). Surgindo assim o *Eurodicautom*, que nasceu para atender as necessidades do tradutor europeu, o *Normaterm*, da associação francesa de normalização (AFNOR). Atualmente no Brasil o projeto mais conhecido neste campo é o Termisul, que possui a tarefa de organizar terminologias em instrumentos bilíngües de modo a colaborar com a integração dos países-membro do MERCOSUL.

Um dicionário eletrônico pode abarcar um pequeno banco de dados terminológicos, entre outras ferramentas menores, mas três características particularizam a arquitetura de um banco de dados terminológicos em relação aos dicionários: (KRIEGER e FINATTO, 2004)

- Integração – suporte único de informações terminológicas variadas, procedentes de diversas fontes que podem ser acessadas pela mesma linguagem de consulta.
- Estruturação – realiza-se pelos itens que perfazem a ficha terminológica, onde ao se registrar um termo conterá seus respectivos dados em campos definidos.
- Volume da informação – um banco de dados terminológico é formado por um repertório considerável de informações sobre os termos e textos de várias áreas do conhecimento.

Os registros terminológicos geralmente compreendem os campos: termo; fonte de referência; aspectos conceituais; aspectos lingüísticos; aspectos pragmáticos; outras línguas (equivalências) e gestão da informação (CABRÉ, 1993).

Desta forma, as experiências dos últimos vinte anos, enfatizam que uma definição única sobre estas bases de dados não é viável, devido a diversidade lingüística e cultural existente, bem como as necessidades diferentes que os campos de aplicação acabam por exigir (Fortin *apud* CABRÉ, 1993). Muitos dos conceitos acima apresentados, satisfazem as necessidades do campo da Terminologia e campos correlacionados, quando o objetivo for o contrário, os BDTerm servirem como suporte a informática, alguns conceitos terão que ser revistos, podendo haver a retirada de especificações, bem como a inserção de outras.

Existe o conceito de banco de dados léxicos, que merece atenção por ser semelhante em forma de construção a um BDTerm, mas o conteúdo da informação representada é diferente. Enquanto que a terminologia se aplica ao domínio específico de uma determinada área, a lexicologia estuda todas as palavras de uma determinada língua. Desta forma, a terminologia pode ser vista como um subconjunto da lexicologia (ANDRADE, 1998).

4 BInXS – Um Processo de Integração de Esquemas XML

4.1 O Processo de Integração de Esquemas XML

A motivação para realizar a integração de esquemas XML, ou seja a partir de esquemas locais (que estão distribuídos por várias fontes de dados) gerar um esquema global, se deve ao fato de resolver as diversidades estruturais e semânticas entre eles, permitindo desta forma o acesso integrado a uma grande quantidade de fonte de dados (MELLO, 2002).

A integração de esquemas XML, ou dados semi-estruturados, não pode seguir as mesmas regras definidas para a integração de dados obtidos a partir de banco de dados. Isto porque a natureza dos dados semi-estruturados, é diferente dos dados tradicionais de banco de dados, uma vez que a instância de uma mesma informação em uma mesma fonte de dados pode ser diferente (MELLO, 2002).

A integração se dá inicio através da extração de visões, ou esquemas XML das bases de referências, estas visões serão a entrada para o sistema que realizará a integração .

> "Há duas aproximações principais de prover visões integradas a partir de múltiplas fontes de dados. A primeira, chamada de materialização, extraí dados adiantados e os armazena em um repositório. Depois as consultas do usuário são direcionadas diretamente para esse repositório e são processadas sem a necessidade de acessar diretamente a fonte de dados. Na segunda aproximação, chamada de sistema mediador, as visões não são materializadas, no entanto as consultas necessitam acessar a fonte de dados. Em ambos os casos permitem que o usuário formule as consultas através de um esquema intermediário, livrando-os da localização e heterogeneidade dos dados"
> (BELLAHSENE, 2004).

Vários sistemas de integração na Web atuais são baseados em mediadores e utilizam XML como modelo comum, os quais são denominados de *XML-Based Mediators* (BOAS, 2004). Eles estão associados a uma recente categoria de sistema de gerenciamento de dados heterogêneos, chamada de *sistemas de informação federados Baseados em Mediadores* (SIFBM), como mostrado na Figura 4.1(Busse *apud* MELLO 2000).

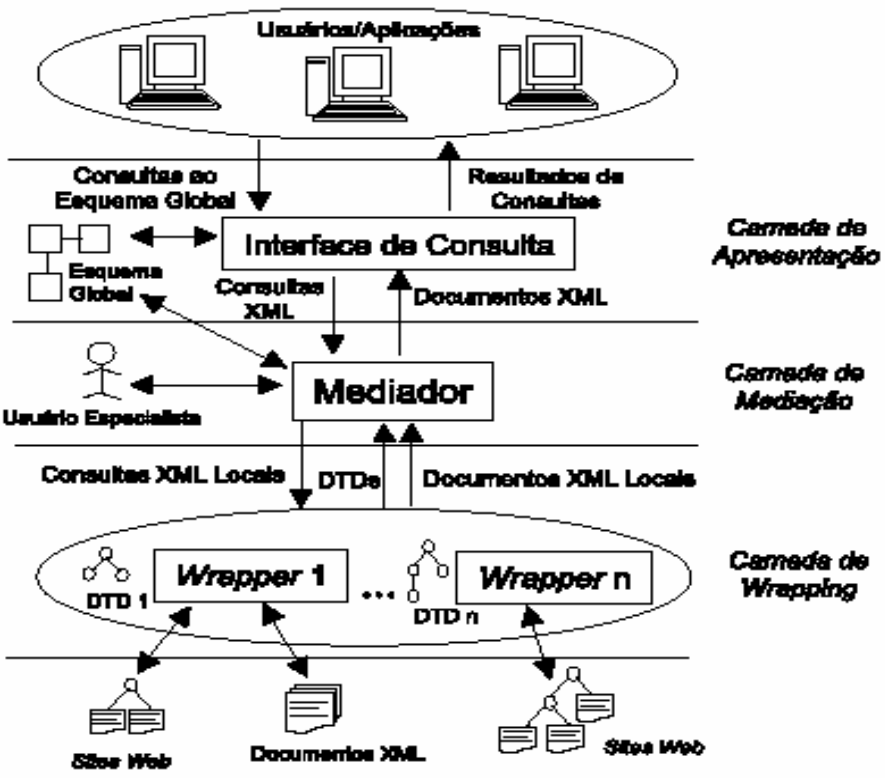

Figura 4.1 Arquitetura para um sistema baseado em mediador

Dentro das possibilidades de realizar o processo de integração de esquemas XML, será considerada a proposta apresentada como BInXS, uma vez que este trabalho tem por objetivo apresentar uma ferramenta que auxilie o BInXS no processo de integração.

4.2 BInXS

BInXS é parte integrante de uma camada de mediação para acesso a múltiplas fontes de dados XML, com o objetivo de realizar a integração semântica dos esquemas XML das fontes de dados na Web e possibilitar a consulta às mesmas através de um esquema global. As informações nesta seção foram retiradas de (MELLO, 2002).

O processo de integração do BInXS é semi-automático, pois exige a intervenção de um usuário especialista para decidir pela interpretação semântica

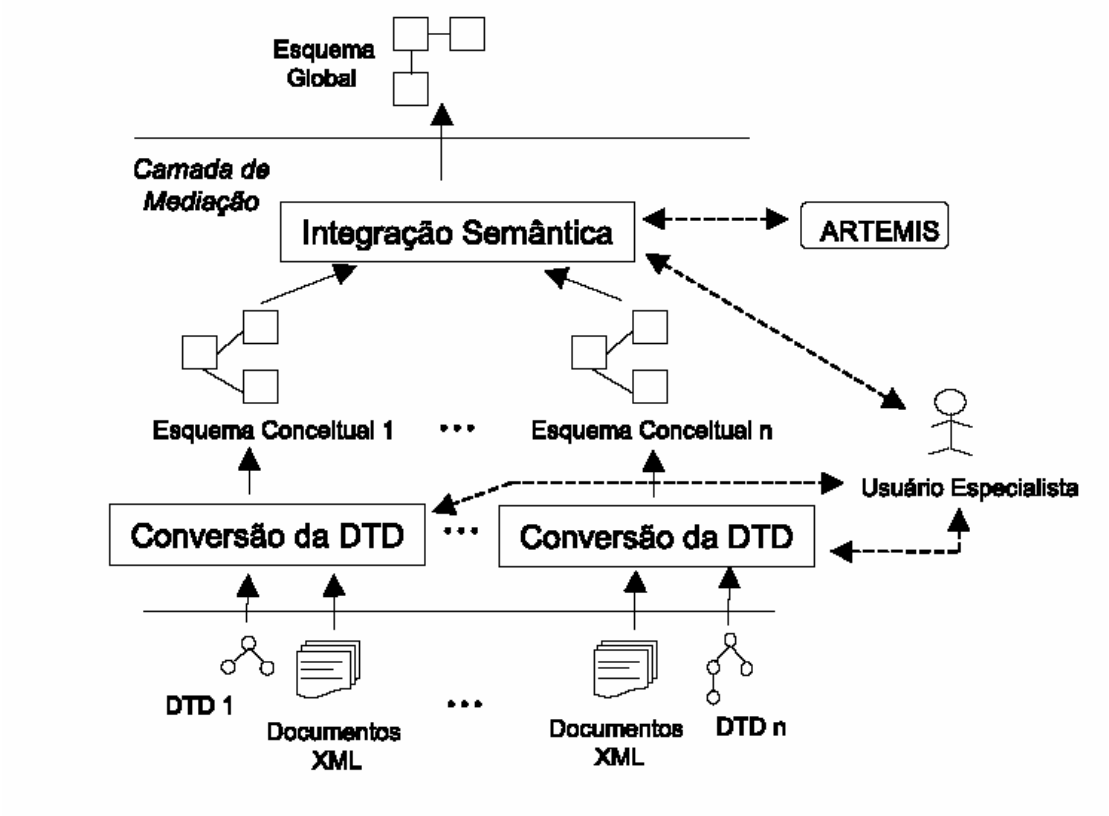

Figura 4.2 Integração de esquemas XML

durante as tarefas de abstração conceitual e unificação de esquemas XML. A característica *bottom-up* (a integração é realizada a partir de esquemas locais já existentes, para gerar um esquema global) deste processo têm o objetivo de oferecer um esquema global que sirva como vocabulário de referência para uma consulta a um conjunto de fontes XML, sendo que a manutenção deste esquema global é vantajosa deixa de forma transparente para os usuários e aplicações a alta heterogeneidade estrutural das fontes de dados. A adoção desta forma de esquema também dá a característica ao BInXS de um banco de dados federado, evitando que usuários e aplicações tenham conhecimento sobre os esquemas locais de cada fonte XML.

O processo de integração de BInXS apresenta duas etapas, conforme mostra a Figura 4.2.

A etapa de Conversão da DTD é responsável pela conversão de cada DTD associada a uma ou mais fontes XML em um esquema conceitual canônico. A utilização deste modelo permite uma abstração da DTD que leva em conta a semântica dos seus dados, facilitando a etapa posterior de integração. Como nem sempre é possível determinar a intenção semântica do dado, a intervenção do

usuário pode ser requerida. Outra característica desta etapa é que não somente informação sobre os esquemas é analisada, mas também o conteúdo dos arquivos XML para auxiliar na definição do esquema conceitual.

A etapa de integração semântica é responsável pela integração dos esquemas canônicos, gerando o esquema global e mapeamentos dos seus conceitos para elementos e atributos semanticamente equivalentes em cada DTD. O esquema global é também representado no modelo canônico. A tarefa de integração segue basicamente os passos do processo de integração de esquemas tradicionais (ver próxima seção), sendo o usuário solicitado para confirmar ou efetivar certas ações que ocorrem nas fases de comparação dos esquemas e unificação. Nesta etapa uma ferramenta chamada *ARTEMIS* é utilizada na determinação de afinidades semânticas entre conceitos de esquemas locais diferentes.

4.3 Integração Semântica

Nesta segunda etapa do BInXS, o objetivo é realizar a validação da intenção semântica dos esquemas conceituais obtidos (MELLO, 2002), ou seja dado dois termos é necessário, através de métodos de comparação semântica, dizer se estes são equivalentes, ou possuem algum outro tipo de relação. No BInXS, os esquemas conceituais obtidos passam por várias etapas, como mostrado na Figura 4.3, Este processo está dividido em quatro etapas principais:

- Agrupamento de conceitos sinônimos – responsável pela verificação de equivalência entre termos, através de sinonímias. Nesta etapa, o *ARTEMIS* é empregado para determinar atividades entre conceitos de esquemas diferentes, retornando estes conceitos agrupados em clusters de afinidade, que são ser validados pelo usuário. Na Figura 4.3 um esquema conceitual para a DTD1 e outro para a DTD2, serve como entrada para o *ARTEMIS*, e os termos são agrupados em *clusters* de afinidade;
- Unificação: realiza a unificação dos conceitos presentes em cada cluster, com a geração de um esquema global preliminar como resultado. Esta unificação resolve basicamente conflitos de nomes, relacionamentos e disjunções de

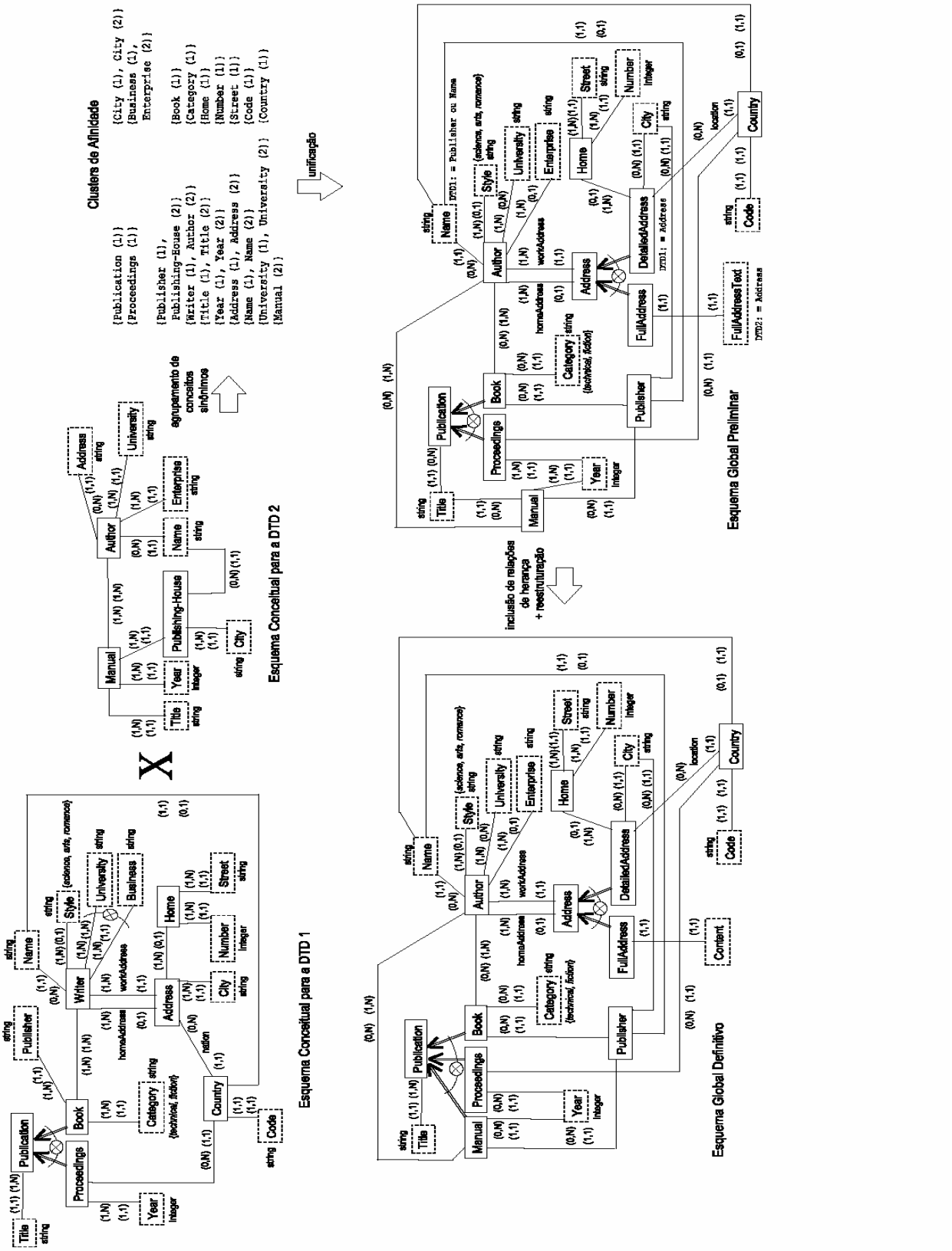

Figura 4.3 Exemplo de aplicação da etapa de integração semântica

relacionamentos, além de considerar os tipos de conceitos presentes em

cada cluster;[

- Inclusão de Relação de Herança: como o esquema global preliminar é o resultado da unificação de diversos esquemas conceituais, é possível que pares de conceitos advindos de esquemas conceituais diferentes tenham um relacionamento de herança, podendo ser este relacionamento relevante para o domínio. Para esta etapa a consulta a um *Thesaurus* se faz necessária, no caso o *WordNet* (WORDNET, 2004);
- Reestruturação: realiza ajustes automáticos e manuais no esquema global preliminar, afim de gerar um esquema global definitivo. Como por exemplo a generalização de relacionamentos, onde o relacionamento de *Title* com *Proceedings* e *Book* é generalizado para *Publication*.

4.4 Necessidade de um Banco de Bados Terminológico

Atualmente o BInXS é um processo semi-automático, pois necessita da interação de um usuário especialista para validar algumas etapas do processo. Boa parte do processo de relacionamento entre conceitos é feito com o auxilio das ferramentas *ARTEMIS* e do WordNet.

O projeto *ARTEMIS*, desenvolvido pelo laboratório ISLab do Departamento de Informática e Comunicação da Universidade de Milano (*ARTEMIS*, 2004), utiliza de *thesaurus* para geração de esquemas da intenção semântica, através de três alternativas: o usuário cria um *thesaurus* com o seu próprio domínio, usa um domínio extraído do *WordNet* e um terceiro que é um híbrido dos dois primeiros (*ARTEMIS*, 2004).

O *ARTEMIS* provê métodos e ferramentas para a integração semântica para base de dados heterogênea. Utilizando um modelo canônico de referência para os esquemas a serem integrados chamado de *ODL*, capaz de suportar esquemas de banco de dados relacionais e orientados a objetos, bem como esquemas semi-estruturados (MELLO, 2002). O BInXS, para poder interagir com o sistema *ARTEMIS*, deve converter os seus esquemas conceituais em esquemas ODL, e o *ARTEMIS* gera os *clusters de afinidade*, que são processados na etapa de unificação (MELLO, 2002).

O WordNet, desenvolvido pelo Laboratório de Ciências Cognitivas da Universidade de Princeton (L), é uma base de dados léxica da língua inglesa (não sendo um banco de dados, por não se utilizar de nenhum sistema deste tipo), onde substantivos, verbos, adjetivos e advérbios, da língua inglesa, estão organizados em conjuntos de sinônimos, e diferentes relações interligam estes conjuntos. Ele é empregado no processo de integração semântica, na etapa de inclusão da relação de herança do BInXS, para verificar se dois conceitos possuem relações de hierarquia, ou seja, verificar se os conceitos possuem uma relação de especialidade ou generalização entre si (MELLO, 2002).

O *WordNet* informa a relação que um termo tem com outros termos, com possibilidade de agrupamento por relevância. Por exemplo ao consultar a palavra *bird*¸ Figura 4.4, é retornada uma lista com todas as definições para esta palavra, bem como verificar as relações que esta possui, podendo ser:

- Sinônimos: conjunto de termos diferentes com o mesmo significado, por exemplo a Figura 4.5 mostra os sinônimos para o primeiro significado da ocorrência para "*bird*";
- Termos coordenados: são termos subordinados a um mesmo conceito, através de relações lógicas, como pode ser visto na Figura 4.6;
- Hiperônimo: quando um termo é mais genérico que o outro, mais abrangente, como exemplificado na Figura 4.7;
- "Merônimos: termos que compõe outro termo, por exemplo Figura 4.8, "wing" é parte de "*bird*";
- Hipônimos: termo mais específico que outro, como exemplo a *Figura 4.9*.

A utilização de um BDTerm, traz uma série de vantagens para o processo de integração de BInXS e contribui com a disponibilização de funcionalidades que melhoram o suporte ao processo de integração do BInXS. Primeiramente por evitar a conversão para esquemas conceituais ODL do *ARTEMIS* e elimina o acesso ao WordNet. Vale observar que o *WordNet* possui relação de equivalência, através de sinonímias, mas mesmo assim o BInXS consulta o *ARTEMIS* para esta etapa, isto se ao fato de que o *ARTEMIS* permite atribuir valores empíricos, definido pelo usuário do sistema, de quanto um termo é mais semelhante a outro. Isto se deve ao fato de que o WordNet, foi construído para ser uma base de dados da língua inglesa,

```
The noun bird has 5 senses (first 2 from tagged texts)

1. (31) bird -- (warm-blooded egg-laying vertebrates
characterized by feathers and forelimbs modified as
wings)
2. (1) bird, fowl -- (the flesh of a bird or fowl (wild
or domestic) used as food)
3. dame, doll, wench, skirt, chick, bird -- (informal
terms for a (young) woman)
4. boo, hoot, Bronx cheer, hiss, raspberry, razzing,
snort, bird -- (a cry or noise made to express
displeasure or contempt)
5. shuttlecock, bird, birdie, shuttle -- (badminton
equipment consisting of a ball of cork or rubber with a
crown of feathers)

The verb bird has 1 sense (no senses from tagged texts)

1. bird, birdwatch -- (watch and study birds in their
natural habitat)
```

Figura 4.4 Relação de significados para o substantivo "*bird*"

```
Sense 1
bird -- (warm-blooded egg-laying vertebrates
characterized by feathers and forelimbs modified as
wings)
       => vertebrate, craniate -- (animals having a bony
or cartilaginous skeleton with a segmented spinal column
and a large brain enclosed in a skull or cranium)
```

Figura 4.5 Relação de sinônimos para "*bird*"

ou seja, ele abrange termos, no ponto de vista de aplicação de uma língua, a inglesa no caso, desta forma o conceito de "estrutura" tem um significado geral, mas teria conceitos diferentes se considerarmos áreas como Ciência da Computação, Engenharia Civil, Letras, entre muitas outras possíveis. Também deve levar-se em conta que o WordNet, não utiliza mecanismos de banco de dados (pelo menos na versão *Desktop*), o que perde em qualidade e compreensão de sua estrutura final.

Como um BDTerm foi relacionado a um domínio, elimina-se o procedimento de valoração empírica, pois se dois termos são sinônimos a relação de equivalência é evidente. Um BDTerm pode incorporar ferramentas agregadas, para aumentar a sua funcionalidade, como a inclusão de um *thesaurus*. As consultas podem ser procedimentos mais simples.

```
Sense 1
bird -- (warm-blooded egg-laying vertebrates
characterized by feathers and forelimbs modified as
wings)
    -> vertebrate, craniate -- (animals having a bony or
cartilaginous skeleton with a segmented spinal column
and a large brain enclosed in a skull or cranium)
       => fetus, foetus -- (an unborn or unhatched
vertebrate in the later stages of development showing
the main recognizable features of the mature animal)
       => Amniota -- (higher vertebrates (reptiles,
birds and mammals) possessing an amnion during
development)
       => amniote -- (any member of the Amniota)
       => aquatic vertebrate -- (animal living wholly or
chiefly in or on water)
       => gnathostome -- (a vertebrate animal possessing
true jaws)
       => bird -- (warm-blooded egg-laying vertebrates
characterized by feathers and forelimbs modified as
wings)
       => amphibian -- (cold-blooded vertebrate
typically living on land but breeding in water; aquatic
larvae undergo metamorphosis into adult form)
       => reptile, reptilian -- (any cold-blooded
vertebrate of the class Reptilia including tortoises
turtles snakes lizards alligators crocodiles and extinct
forms)
       => mammal -- (any warm-blooded vertebrate having
the skin more or less covered with hair; young are born
alive except for the small subclass of monotremes and
nourished with milk)
       => tetrapod -- (a vertebrate animal having four
feet or legs or leglike appendages)
```

Figura 4.6 Relação dos termos coordenados para "*bird*"

```
Sense 1
bird -- (warm-blooded egg-laying vertebrates
characterized by feathers and forelimbs modified as
wings)
       => vertebrate, craniate -- (animals having a bony
or cartilaginous skeleton with a segmented spinal column
and a large brain enclosed in a skull or cranium)
          => chordate -- (any animal of the phylum
Chordata having a notochord or spinal column)
             => animal, animate being, beast, brute,
creature, fauna -- (a living organism characterized by
voluntary movement)
                => organism, being -- (a living thing
that has (or can develop) the ability to act or function
independently)
                   => living thing, animate thing --
(a living (or once living) entity)
                      => object, physical object --
(a tangible and visible entity; an entity that can cast
a shadow; "it was full of rackets, balls and other
objects")
                         => entity -- (that which
is perceived or known or inferred to have its own
distinct existence (living or nonliving))
```

Figura 4.7 Relação de hiperônimos para "*bird*"

```
Sense 1
bird -- (warm-blooded egg-laying vertebrates
characterized by feathers and forelimbs modified as
wings)
          HAS PART: beak, bill, neb, nib, pecker --
(horny projecting mouth of a bird)
          HAS PART: furcula -- (a forked bone formed by
the fusion of the clavicles of most birds)
          HAS PART: feather, plume, plumage -- (the
light horny waterproof structure forming the external
covering of birds)
          HAS PART: wing -- (a movable organ for flying
(one of a pair))
          HAS PART: pennon, pinion -- (wing of a bird)
          HAS PART: bird's foot -- (the foot of a bird)
          HAS PART: uropygium -- (posterior part of a
bird's body from which the tail feathers grow)
          HAS PART: air sac -- (any of the membranous
air-filled extensions of the lungs of birds)
          HAS PART: uropygial gland, preen gland --
(oil-secreting gland situated at the base of the tail in
most birds)
          HAS PART: syrinx -- (the vocal organ of a
bird)
          HAS PART: bird, fowl -- (the flesh of a bird
or fowl (wild or domestic) used as food)
```

Figura 4.8 Relação de merônimos para "*bird*"

```
bird -- (warm-blooded egg-laying vertebrates
characterized by feathers and forelimbs modified as
wings)
       => dickeybird, dickey-bird, dickybird, dicky-bird
-- (small bird; adults talking to children sometimes
call small birds dickeybirds)
       => cock -- (adult male bird)
       => hen -- (adult female bird)
       => nester -- (a bird that has built (or is
building) a nest)
       => night bird -- (any bird associated with night:
owl; nightingale; nighthawk; etc)
       => bird of passage -- (any bird that migrates
seasonally)
       => protoavis -- (most primitive avian type known;
extinct bird of the Triassic having birdlike jaw and
hollow limbs and breastbone with dinosaur-like tail and
hind limbs)
       => archaeopteryx, archeopteryx, Archaeopteryx
lithographica -- (extinct primitive toothed bird of the
Jurassic period having a long feathered tail and hollow
bones; usually considered the most primitive of all
birds)
       => Sinornis -- (sparrow-sized fossil bird of the
Jurassic period to the Cretaceous period having a keeled
breastbone and vestigial tail; found in China;
considered possibly the second most primitive of all
birds)
       => Ibero-mesornis -- (sparrow-sized fossil bird
of the Cretaceous period having a vestigial tail; found
in Spain; considered possibly the third most primitive
of all birds)
       => archaeornis -- (extinct primitive toothed bird
with a long feathered tail and three free clawed digits
on each wing)
```

Figura 4.9 Relação de hipônimos para "*bird*"

5 Sistema de Gerenciamento de Bases de Dados Terminológicas

Conforme foi justificado anteriormente, para dar suporte a etapa de integração semântica utilizada por sistemas de integração de dados, como o BInXS, a solução encontrada foi a criação de um Banco de Dados Terminológico (BDTerm) e um conjunto de funcionalidades que permita o gerenciamento destas bases, definindo assim um sistema de gerenciamento de BDTerms.

O principal motivo que leva à adoção desta solução é que termos têm significados diferenciados, conforme um domínio do conhecimento. O processo de integração de documentos XML apresentado pelo BInXS interage sempre dentro de uma mesma área afim, ou seja, dentro de um mesmo domínio. Com a disponibilização de BDTerms para domínios específicos aumenta-se a confiabilidade da etapa de determinação de equivalências semânticas em um processo de integração de dados, reduzindo o esforço de validação por parte de um usuário especialista.

O suporte necessário ao BInXS, durante a etapa de determinação de equivalências semânticas entre dados, foi o que motivou este trabalho. Porém, este trabalho pode ser utilizado por qualquer outro sistema de integração de dados que julgar importante este suporte.

Durante a etapa de projeto do sistema também se determinou que para um maior crescimento da base, as informações devem ser preenchidas pelo maior número de usuários possíveis. Desta forma, convencionou-se que uma BDTerm também deve estar disponível para consultas ao público, funcionando como um sistema de busca conceitual de termos e suas relações semânticas existentes. Mais detalhes sobre o sistema são descritos nas próximas seções.

5.1 Arquitetura

A construção de sistemas computacionais sempre deve prever a interação com usuários, sejam estes seres humanos ou sistemas informatizados. Esta

interação passa a ser um ponto de atenção na hora de definir uma interface de comunicação, pois é necessário garantir que o sistema ofereça recursos de interação aos seus usuários, sem que estes possam vir a prejudicá-lo.

Desta forma, foi definida uma arquitetura para permitir que os usuários interajam com o BDTerm de maneira adequada. Esta arquitetura é mostrada na Figura 5.1. Nesta são previstos três tipos de usuários (definidos a seguir) para interagir com o sistema através de uma camada (biblioteca) de funções.

Esta camada de funções tem por objetivo atender as necessidades de interação de um determinado usuário ou de um grupo de usuários,. atuando como uma camada intermediária entre os usuários e as BDTerms.

Este esquema possui uma outra camada de interação: a interface WEB. Esta camada visa separar a apresentação da camada de funções, realizando a troca de informações entre a camada de nível inferior e os usuários (nível superior). Também é responsabilidade desta interface tratar a informação que está sendo enviada ao sistema, bem como, os dados que são retornados pelo mesmo.

Existe um outro tipo de usuário que não aparece no esquema, este é comumente chamado de administrador do banco de dados - administrador. Como este usuário está constantemente acessando um sistema de banco de dados, ele possui ferramentas (gráficas ou não) próprias para realizar as suas atividades. Como este tipo de usuário tem plenos poderes para interagir com o sistema, não é necessário definir políticas de acesso e nem sobrecarregar a camada de interface com funções para validar as suas ações.

Os quatro tipos de usuários que interagem com o BDTerm possuem papéis bem definidos. Desta forma o conjunto de suas ações tem por objetivo manter a estabilidade, a evolução do sistema como um todo ou apenas buscar informação.

O *administrador* é o responsável por todo o sistema de BDTerms, ou seja, este usuário tem permissão para executar funções de gerenciamento sobre o sistema, como criar novos domínios (BDTerms), inserir e remover usuários. E garantir o funcionamento do sistema.

O *SID* (Sistema de Integração de Dados) é um usuário não-humano que corresponde a um sistema de integração de dados que acessa os BDTerms, como o BInXS. Ele pode executar funções de consulta às BDTerms para atender

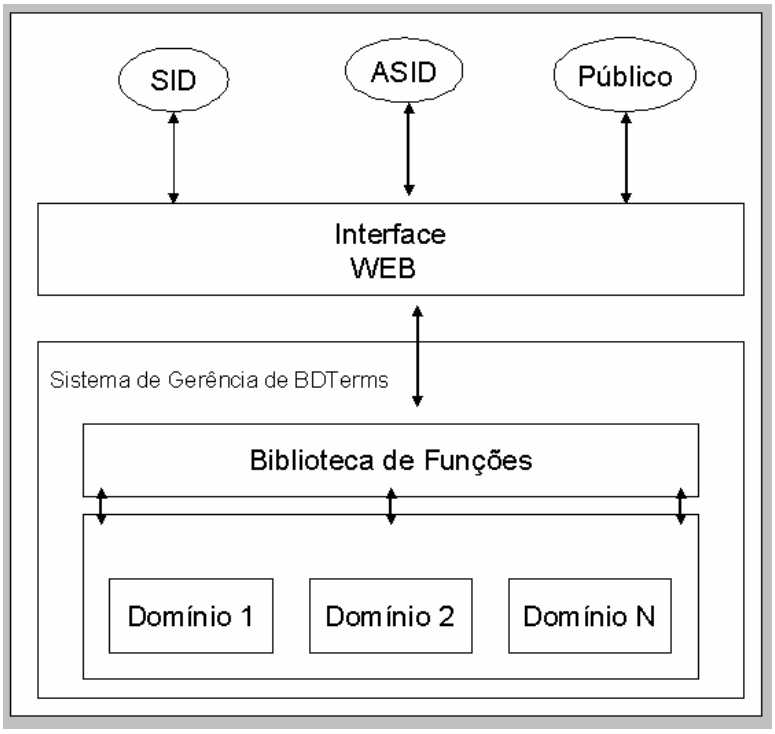

Figura 5.1 Arquitetura

as suas necessidades durante a etapa de integração semântica de dados.

Estas consultas retornam tanto termos correntes do domínio, bem como termos que fazem parte deste domínio somente ao nível de integração de dados. Esta diferença conceitual está definida na próxima seção, quando são apresentados os requisitos do sistema.

O usuário *ASID* (Administrador de Sistemas de Integração de Dados) é um especialista em sistemas de integração de dados, tendo permissão para inserir informações dentro de uma BDTerm. Estas têm a intenção de aumentar o conteúdo do domínio, melhorando assim o poder de determinação de equivalências semânticas por parte de um *SID*. Novamente, esta diferença é abordada na seção sobre os requisitos de dados do sistema (seção 5.2.1).

O usuário público é responsável pelo gerenciamento do conteúdo de um domínio de uma BDTerm, estando subdividido em três sub-categorias: *gerenciador* (UPG), *colaborador* (UPC) e *participante* (UPP). O *gerenciador* é o responsável por um determinado domínio, podendo inserir informações, validar conceitos, relações e inserir novos usuários, sejam estes, gerenciadores ou

colaboradores. O *colaborador* pode somente inserir ou validar as informações presentes no domínio de qual faz parte. Já o *participante* pode realizar consultas a um determinado domínio e realizar a inserção de informações.

5.2 Projeto do Sistema e do Banco de Dados

Esta seção apresenta os requisitos e os fatos do mundo real relevantes para o domínio do problema, bem como os modelos que são gerados ao longo da etapa de construção do banco de dados do sistema. Nas sub-seções seguintes são apresentados inicialmente os requisitos que definem o problema. Na seqüência, é gerado o modelo conceitual a partir dos requisitos e posteriormente o modelo lógico do banco de dados.

5.2.1 Descrição de Requisitos do Sistema

O conceito sobre *termo* apresentado pela terminologia nos permite deduzir que um termo possui somente uma definição dentro de um determinado domínio. Com base nesta afirmativa é que o sistema apresentado neste trabalho foi proposto.

No entanto, esta afirmativa nem sempre pode ser empregada, pois um termo dentro de uma área específica do conhecimento pode estar associado a mais de uma definição. Isto é comum quando novos termos vão surgindo e a sua conceituação ainda esteja se moldando conforme as especificações definidas pelos pesquisadores da área.

Como exemplo, podemos citar o conceito de SGBD (Sistemas de Gerenciamento de Banco de Dados), apresentados por dois autores diferentes. Segundo (Silberschatz, 1999) um SGBD consiste em uma coleção de dados inter-relacionados e em um conjunto de programas para acessá-los. Para (Ramakrishnam, 2000) SGBD é um programa para ajudar na manutenção e utilização de grandes coleções de dados. Existe uma diferença sutil nas duas definições, mas, no geral, carregam a mesma informação.

Desta forma, pode-se desejar descrever um termo que apresente duas definições aparentemente iguais, mas que possuem pontos de vista diferenciados para dois autores consagrados sobre o assunto. Para resolver o problema exemplificado e garantir a unicidade conceitual do termo, as seguintes sugestões são feitas:

- Inserir dois ou mais significados para um mesmo termo, deixando a responsabilidade para o usuário de distinguir a conceituação correta dentro do texto da definição do termo (recomendado). Exemplo:
  - Termo: SGBD
  - Conceito: Segundo Silberschatz um SGBD consiste de uma coleção de dados inter-relacionados e em um conjunto de programas para acessá-los. Para Ramakrishnam é um programa para ajudar na manutenção e utilização de grandes coleções de dados;
- Cadastrar o termo o mais abrangente e para cada conceituação diferenciada inserir novos termos com o mesmo nome, mas que apresente alguma informação adicional ao nome, mantendo assim a unicidade da informação. Depois com a utilização de relações semânticas (apresentadas mais à frente) é possível manter a inter-relação entre os termos. Exemplo:
  - Termo: SGBD
  - Conceito: sistema responsável por gerenciar banco de dados.
  - Termo: SGBD(Silberschatz)
  - Conceito: Segundo Silberschatz, um SGBD consiste de uma coleção de dados inter-relacionados e em um conjunto de programas para acessá-los.
  - Termo: SGBD(Ramakrishnam)
  - Conceito: Para Ramakrishnam, é um programa para ajudar na manutenção e utilização de grandes coleções de dados;

O objetivo do tratamento acima é permitir que um mesmo termo apresente mais de uma definição conceitual sem afetar a regra de unicidade dentro de um domínio. O recomendado é a primeira opção, pois fica a cargo dos autores do

domínio apresentar a conceituação do termo e suas variações conceituais dentro da definição deste. Em todos os casos, as duas alternativas são válidas e aceitas.

Para melhor compreender sobre os domínios que formam o sistema, convencionou-se que estes podem herdar um outro já existente. Isto foi adotado para permitir que existam dois domínios com o mesmo nome, mas que pertençam a áreas diferentes. Por exemplo, se fosse desejado criar o domínio “redes”, este domínio pode estar relacionado ao contexto de “computadores” ou de “pesca”. Desta forma, ao se criar o domínio “redes”, pode-se especificar à qual super-domínio ele se enquadra, permitindo assim uma construção hierárquica entre os domínios do sistema. Deve ficar claro que não é permitido dois ou mais domínios com o mesmo nome dentro de um mesmo super-domínio.

As relações semânticas têm por objetivo interligar termos através de relações de equivalência - *sinonímias* ou de hierarquia – *hiperonímias* (quando um termo é mais geral que outro) e *hiponímias* (quando um termo é mais específico que outro). Neste trabalho, somente são consideradas estas três possibilidades, embora existam outras.

Dois termos podem possuir ou não uma relação semântica dentro de um mesmo domínio, não sendo permitido que os mesmos possuam mais de um tipo de relação semântica dentro do mesmo domínio. Além disso, um termo não tem a capacidade de se auto-relacionar. Estas restrições evitam possíveis erros conceituais dentro de uma BDTerm, aumentando assim a semelhança do sistema modelado com a realidade.

Os usuários públicos do sistema, conforme as suas atribuições, têm por finalidade manter o sistema em funcionamento e em constante evolução. De forma que uma BDTerm possa evoluir, deve-se permitir que o máximo de informação presente em um domínio possa ser carregada para dentro do sistema.

Para realizar a inserção de informações dentro de uma BDTerm, o ideal é que um ou mais usuários especialistas sobre o assunto sejam os responsáveis por esta atividade. Entretanto, a inserção de informações dentro de um SGBD, quando feito de forma manual, não é uma tarefa agradável. Para melhor realizar esta atividade, se utiliza o princípio de que qualquer pessoa pode inserir um termo e sua definição dentro de um domínio e relacionar semanticamente este termo

inserido a um outro termo já existente, além de realizar consultas as informações disponibilizadas. Este é o papel do *UPP*.

Porém, isso pode levar ao problema de falta de confiabilidade do sistema, uma vez que um *UPP* pode adicionar uma informação incorreta. Para resolver isso, o sistema trabalha com um mecanismo de validação posterior da informação, ou seja, toda informação inserida é considerada incorreta e não pode ser visualizada, até que um *UPG* ou *UPC* a confirmem como correta. O que se pretende com esta política de trabalho é permitir que qualquer usuário possa contribuir com o crescimento de um domínio, evitando, porém, que este apresente informações incorretas, que possam vir a prejudicar buscas por informações.

Como os usuários *UPG* e *UPC* têm a responsabilidade de manter a corretude da informação presente em um domínio, deseja-se que estes tenham um grande conhecimento sobre a área que estão gerenciando. Surge aqui um problema para o *administrador* do sistema, que é definir quem pode atuar como gerenciador e ou colaborador dentro de uma área. Para resolver esta situação, convencionou-se que o *administrador* deve, ao inserir um novo domínio no sistema, inserir um usuário que terá o papel de *UPG* dentro deste, ficando a seu cargo inserir novos usuários para o auxiliarem na tarefa de controle do domínio. Esta abordagem supõe que este usuário escolhido para ser o *UPG* conhece outros membros de sua área de conhecimento que poderão lhe ajudar nesta tarefa. Este usuário, ao inserir um novo, deve especificar o papel que este terá no domínio, *UPG* ou *UPC*. O *UPC* tem a mesma funcionalidade que o *UPG*, com a diferença de não poder adicionar nenhum outro usuário ao sistema.

Existem ainda os usuários *SID* e *ASID*. Ambos os usuários estão relacionados aos sistemas de integração de dados.

O usuário *ASID* pode realizar a inserção de termos dentro de um domínio do sistema, mas estes termos se diferem dos demais pelo fato de só terem significância para o processo de integração semântica. Trabalhos na área de integração de dados utilizam vários mecanismos para garantir a equivalência entre dois termos. Isto é motivado pelo fato de que um dado em uma fonte de dados é definido por um usuário com uma semântica particular. Por exemplo, para se referenciar o nome de um autor de um livro, pode-se usar o termo *autor* (considerando que somente esta denominação é suficiente para representar a

informação), ou pode ser utilizado o termo *nomeAutor* para representar a mesma informação.

Como este último termo é importante para o mecanismo de integração, mas não tem sentido como conteúdo informativo em um domínio, é permitido que um termo ao ser inserido contenha um inf ormação de *finalidade*. Esta finalidade pode conter o valor “P”, indicando que este termo é sempre significativo em um domínio, ou “SID”, indicando que ele é útil somente no processo de integração de um SID. Essa diferenciação possibilita que o termo inserido pelo *ASID* não tenha que sofrer validação dos usuários responsáveis pelo domínio e que quando um *SID* fizer uma consulta a uma BDTerm, estes termos especiais sejam consultados. Esta terminologia especial é considerada “invisível” para os usuários públicos.

Uma característica do sistema é permitir que termos inseridos por um *ASID*, sejam consultados por qualquer *SID*. A intenção desta política de trabalho é cruzar o máximo de informações existentes dentro do sistema, independentemente da origem do usuário que a cadastrou. Isto foi motivado pelo fato de que grupos em pesquisa em integração procuram determinar padrões adicionais, na grafia de termos, na tentativa de aumentar a eficiência de seus trabalhos. Se esta informação for compartilhada de alguma forma, pode-se minimizar os custos em repetir trabalhos que foram realizados por outros. Outra justificativa mais simples sobre esta característica do sistema é que o conteúdo do sistema é de caráter público, podendo ser acessado por qualquer usuário. Se algum *ASID* tem o interesse em não disponibilizar o conhecimento adquirido com as suas pesquisas, basta não adicionar estas informações no BDTerm.

Neste sistema, considera-se que um *SID* realiza um mecanismo *abrangente* de busca a BDTerms, ou seja, como este usuário não tem a capacidade de interagir com o sistema de forma mais dinâmica, definiu-se que toda a consulta feita por ele deve retornar a maior quantidade de informações possível. Esta diferença pode melhor ser compreendida com o auxílio da Figura 5.2. Esta figura apresenta um domínio fictício, onde os termos equivalentes estão interligados por um traço e os termos que possuem relações de hierarquia estão conectados por uma flecha, onde a base representa o termo mais genérico e a ponta da flecha o termo mais específico.

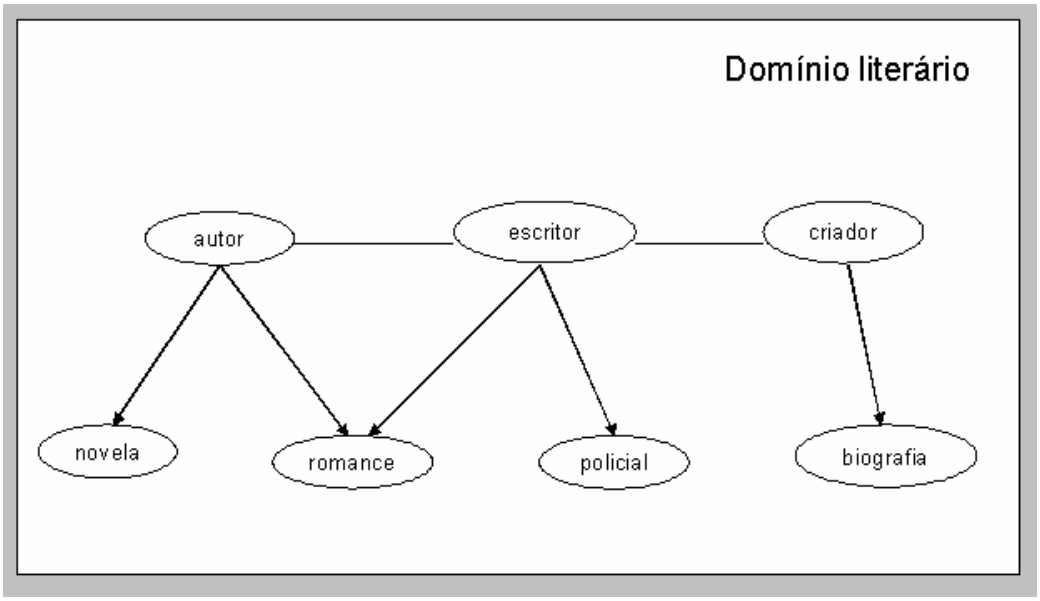

Figura 5.2 Exemplo de um domínio

O usuário público que realiza alguma consulta recebe como resposta informações diretas, ou seja, o que está imediatamente interligado semanticamente (“busca simples”). Por exemplo, ao ser realizada a busca pelos sinônimos de “autor”, a função do sistema associada a esta consulta retorna “escritor”. Isto pode parecer uma limitação da ferramenta, mas o principal objetivo é não sobrecarregar o usuário com muitas informações e desta forma deixar por sua vontade a busca do conhecimento associado a um domínio.

Quando a consulta tiver caráter de suporte a validação semântica, ou seja, for executada por um usuário SID, a ferramenta utiliza este princípio de busca abrangente, ou “busca completa”. Desta forma, toda informação que estiver relacionada de forma direta e indireta é retornada. Uma busca completa abrange todas as possibilidades de relações dentro de um domínio.

A partir do exemplo da Figura 5.2, ao se realizar a consulta dos sinônimos de “autor”, através de uma busca simples, é retornado escritor. Mas, pela “busca completa”, os sinônimos de “autor” são os seus equivalentes diretos e os seus equivalentes indiretos (sinônimos dos sinônimos). Desta forma, são retornados : “escritor” e “criador” como sinônimos de “autor”.

O mesmo se aplica quando se deseja saber os hipônimos e hiperônimos de um termo, sendo buscado primeiramente os relacionamentos diretos e posteriormente os seus relacionamentos indiretos. Desta forma, ao se buscar os

hiperônimos de “autor”, são retornados primeiramente “novela” e “romance”, e posteriormente são buscado os hiperônimos dos termos sinonímios a “autor”, retornando assim “policial” e “biografia”. O mesmo se aplica aos hipônimos de um termo.

Deve ser observado que o termo romance é retornado duas vezes. Por isto, controles para evitar redundância de informação são aplicados em buscas completas, bem como controles para evitar ciclos de execução sem parada garantida, ou “*looping infinito*”.

Os requisitos descritos para o sistema tentam garantir que os *SIDs* possam abranger o máximo da informação presente no domínio, vindo a melhorar a etapa de integração semântica. Seja porque é possível definir sobre qual domínio é feita a consulta, ou por abranger uma resposta ampla, ou por permitir que mais informações possam ser agregadas ao sistema, justamente para aumentar o poder de expressão deste junto aos mecanismos de integração.

Com as conceituações sobre o que é esperado deste sistema, é possível apresentar o projeto do seu banco de dados e descrever em detalhe as suas funcionalidades.

5.2.2 Modelagem Conceitual

Um modelo conceitual registra os dados que podem aparecer no banco de dados, sem a preocupação de como estes dados estão estruturados em um SGBD (HEUSER, 2004). Desta forma, o que importa neste modelo é o que pode ser abstraído do problema proposto em termos de dados, permitindo que esta informação sirva como um ponto de referência entre a realidade e o que será mantido dentro de um banco de dados.

A técnica de modelagem empregada neste trabalho é a entidade-relacionamento – modelagem ER, por ser amplamente utilizada para o projeto de banco de dados. Esta técnica provê recursos gráficos e textuais para realizar a modelagem. O formato gráfico é chamado diagrama entidade relacionamento – DER. Os conceitos de modelagem ER necessários ao entendimento este trabalho são os seguintes (HEUSER, 2004):

- entidade – conjunto de objetos da realidade modelada que se deseja manter alguma informação. No DER, uma entidade é representada por um retângulo contendo em seu interior o nome de sua representação;
- relacionamento – serve para manter informações de associações entre os objetos do modelo. Um relacionamento é representado por um losango no DER, descrito geralmente por um verbo, para dar a idéia de ação e ligado por linhas aos objetos que estão sendo relacionados. Uma das linhas pode ser mais forte do que a outra, para dar a idéia de dependência de uma entidade em relação a outra;
- cardinalidade – uma característica importante em um relacionamento, servindo para quantificar quantas ocorrências de uma determinada entidade podem fazer parte do relacionamento com a outra entidade. Uma cardinalidade é representada no DER por um par de valores entre parênteses, indicando respectivamente, a quantidade mínima e máxima de associações que um objeto de uma entidade pode ter com objetos de outra entidade;
- atributo – dado associado a uma entidade ou relacionamento, com a finalidade de descrever alguma característica da entidade ou do relacionamento. Um atributo pode conter informações que são utilizadas como identificador de uma entidade ou relacionamento. No DER, um atributo é representado por um pequeno círculo conectado à entidade ou ao relacionamento que está recebendo a informação. Ao lado do círculo fica o nome do atributo. Quando este é utilizado para identificar uma entidade ou relacionamento, o círculo encontra-se preenchido.

A modelagem conceitual proposta neste trabalho para implementar o sistema pode ser observada no DER da Figura 5.3. Este foi construído de acordo com a descrição dos requisitos apresentado na seção anterior. A partir do modelo é feita a leitura do que é representando, juntamente com a lista de restrições que está em anexo à figura, para que seja possível verificar a coerência entre o que foi proposto e o que foi modelado.

A entidade **Domínio** corresponde a um conjunto de BDTerms, mantendo informações sobre as BDTerms presentes no sistema. Esta possui um auto-

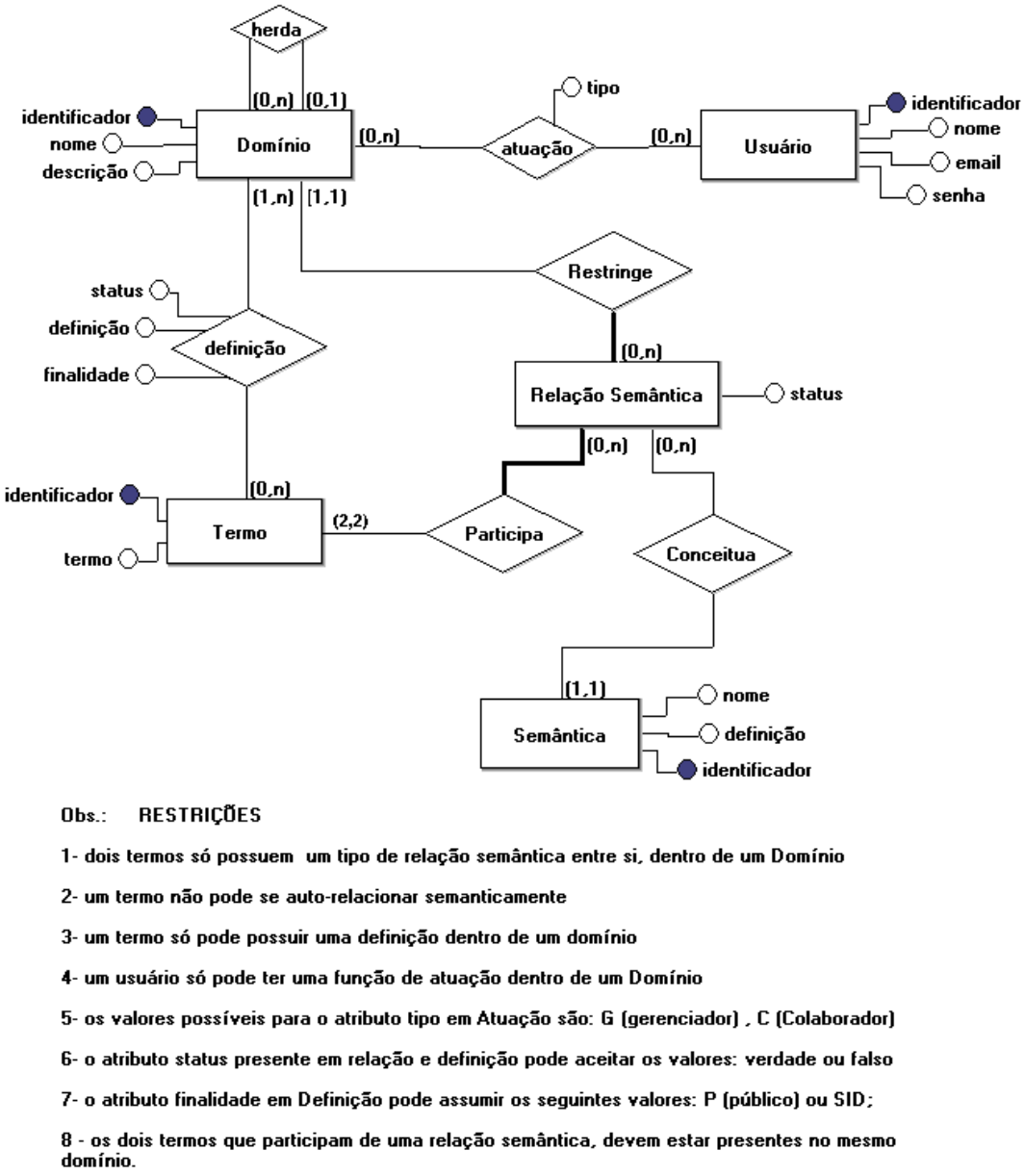

Figura 5.3 Modelagem conceitual do sistema e restrições de integridade associadas

relacionamento, **herda**, desta forma é feita a seguinte leitura: "*um domínio pode ser herdado por zero ou muitos domínios*" e "*um domínio pode herdar zero ou somente um domínio*". Esta entidade possui uma relação com **Usuário**, desta forma: "*zero ou mais usuários atuam em um domínio, com um determinado tipo (G ou C) de atuação*" e "*um domínio sofre a interação de zero ou mais tipos de*

usuários".

A entidade **Termo** contém as dados sobre os termos que estão inseridos no sistema, não carregando nenhuma informação além do nome e de um identificador. Através do relacionamento **definição** desta entidade com **Domínio** é possível fazer a seguinte inferência: "*um termo possui somente uma definição dentro de um ou muitos domínios e esta definição possui um status (verdadeiro ou falso) e uma finalidade (P ou SID)*" e "*um domínio possui a definição de zero ou muitos termos, cada qual com um status e uma finalidade particular*".

A entidade **Relação Semântica** armazena toda a informação que constrói a rede de relacionamentos entre os termos em um domínio e interage com outras três entidades: **Termo**, **Domínio** e **Semântica** (relaciona as classes semânticas presentes no BDTerm). Esta parte da modelagem pode ser entendida da seguinte maneira: "*uma relação semântica é composta pela participação de dois e somente dois termos, que estão restritos a um e somente um domínio e tem um e somente um conceito semântico e tem um status (verdadeiro ou falso), para garantir a sua corretude"* bem como: *"um termo participa de zero ou muitas relações semânticas, um domínio restringe zero ou muitas relações semânticas e a semântica conceitua zero ou muitas relações semânticas, todas estas informações associadas a um status"*.

5.2.3 Modelagem Lógica

Definida a modelagem conceitual do BDTerm, é possível passar para uma próxima etapa do projeto de um banco de dados, que a transforma, com o emprego de regras, em uma modelagem lógica. Esta modelagem se caracteriza por ser a descrição do banco de dados no nível do SGBD, ou seja, a modelagem lógica já é dependente do tipo de SGBD utilizado, como por exemplo: SGBD relacional (SGBDR), objeto-relacional (SGBDOR), orientado a objeto (SGBDOO), entre outros (HEUSER, 2004).

A seguir é descrito o mapeamento da modelagem ER para um SGBDR.

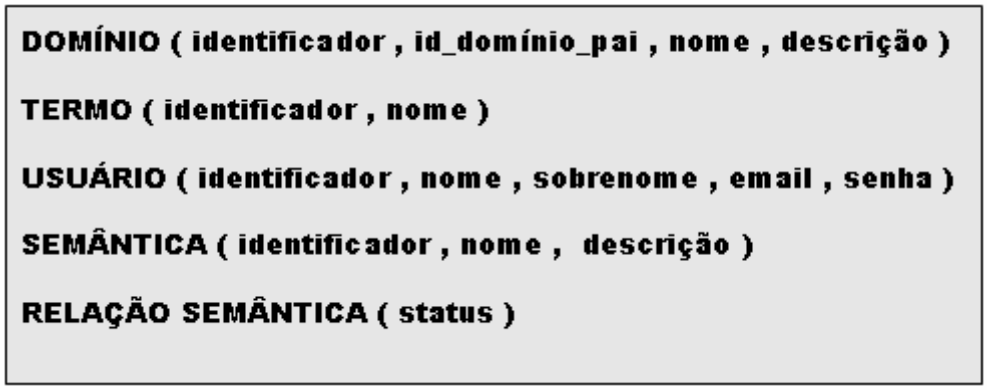

Figura 5.4 Modelo lógico parcial do BDTerm

5.2.3.1 Mapeamento de Entidades

De acordo com as regras de conversão para um BD relacional, basicamente transforma-se uma entidade em uma relação (tabela), utilizando-se o mesmo nome. Os atributos da modelagem conceitual serão os atributos (colunas) do modelo lógico, conforme mostra a Figura 5.4. Atributos são especificados entre parênteses e separados por vírgula. É observado neste modelo que as informações presentes nos relacionamentos não se fazem presentes.

5.2.3.2 Mapeamento de Relacionamentos

O mapeamento dos relacionamentos da modelagem conceitual não foi apresentado na Figura 5.4, uma vez que cada relacionamento existente entre duas entidades deve ser analisado em separado, pois é possível ter mais de um tratamento para acomodar as informações presentes em um relacionamento de um diagrama ER. As regras de conversão baseiam-se no valor mínimo e máximo da cardinalidade no relacionamento entre duas entidades. Por exemplo, na Figura 5.3, a cardinalidade do relacionamento entre **Termo** e **BDTerm** é do tipo **muitos-para-muitos** (valores máximos da cardinalidade da relação) ou **M:N**.

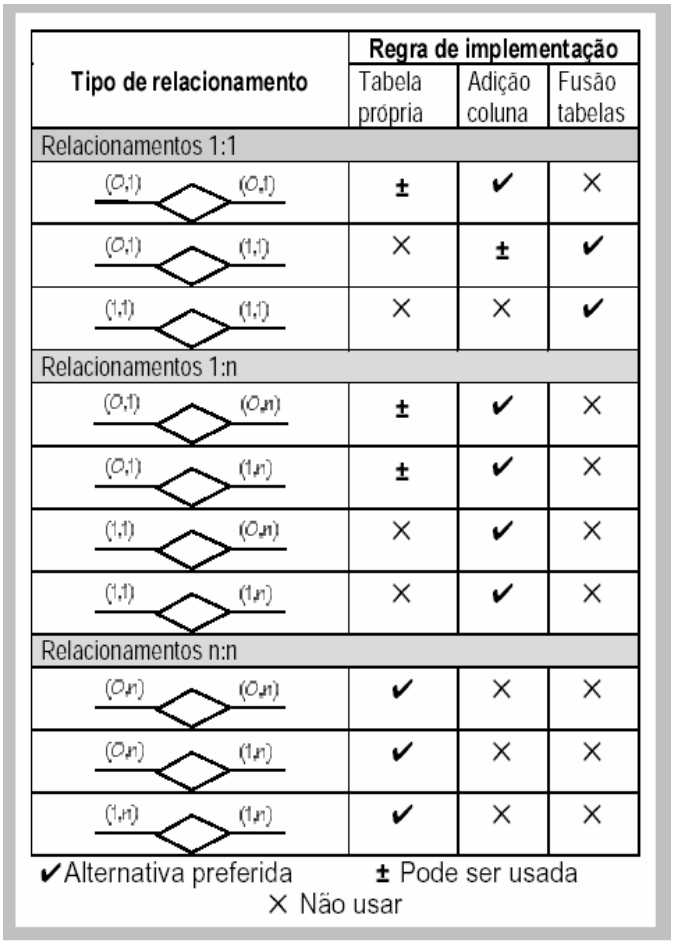

Figura 5. 5 Regras para relacionamentos (HEUSER, 2004)

A Figura 5. 5, extraída de (HEUSER, 2004), mostra as três possibilidades para o mapeamento de relacionamentos:

- *tabela própria*: o relacionamento é mapeado para uma tabela particular no modelo lógico, tendo duas colunas que representam os identificadores das tabelas envolvidas no relacionamento. Caso o relacionamento possua atributos próprios, estes devem ser incluídos nesta tabela;
- *adição de coluna*: quando esta alternativa é adotada, uma das tabelas das entidades envolvidas no relacionamento recebe uma coluna extra que mantém o identificador da tabela correspondente a outra entidade participante do relacionamento. Nem sempre existe critério para decidir qual das tabelas deve receber a coluna extra;
- *fusão de tabelas*: as duas entidades participantes do relacionamento são

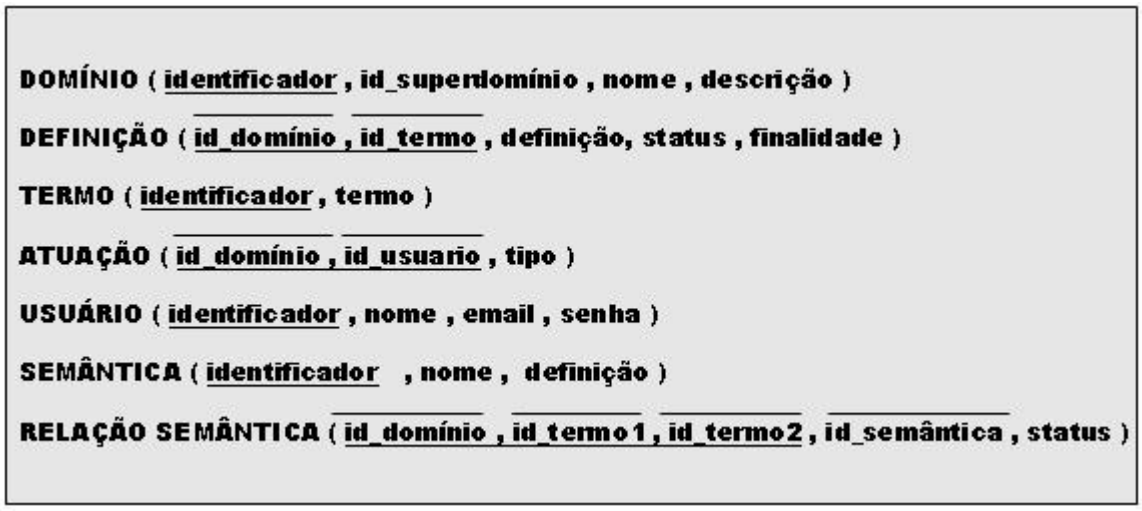

Figura 5. 6 Modelo lógico completo

unificadas em uma única tabela, e um dos identificadores das entidades pode ser excluído. Os atributos do relacionamento, se existirem, ficam também nesta tabela.

Uma vez descrito como os relacionamentos devem ser tratados, é apresentada na Figura 5. 6 a complementação da modelagem lógica do esquema conceitual da Figura 5.3. As escolhas de alternativas de mapeamento são feitas sobre a tabela da Figura 5. 5.

O relacionamento **herda** foi mapeado para uma coluna dentro da tabela **Domínio**, mantendo-se assim toda a informação desejada sobre os domínios e suas heranças. A associação entre **Termo** e **Domínio**, é mapeada para uma tabela própria, que contem os atributos que já faziam parte da relação, sendo adicionada a esta, campos para referenciar os elementos da entidade **Domínio** e **Termo** que participam de **definição**.

O mesmo tratamento foi adotado para interação entre o **Domínio** e **Usuário**, desta forma a tabela gerada contém as referências às entidades envolvidas e mais o atributo tipo.

Para os relacionamentos que envolvem a entidade **Relação Semântica**, adotou-se os seguintes procedimentos. Inseriu-se uma coluna pra fazer referência a **Domínio** e uma para **Semântica.** Para relacionar **Termo**, foram inseridas duas colunas, devido ao fato de que são dois os termos que participam

de um relacionamento. O atributo status manteve-se presente. Nesta tabela é necessária uma restrição de integridade que garanta que os dois termos inseridos realmente façam parte do mesmo domínio em que estão sendo relacionados.

Na Figura 5. 6, os valores sublinhados indicam as chaves primárias e os que possuem um traço sobre o nome são as chaves estrangeiras. As chaves primárias têm por objetivo garantir que o seu valor seja único e diferente de nulo dentro de uma tabela, aumentando assim a performance da consulta. No caso da Figura 5. 6, estas chaves podem ser simples ou compostas. As chaves primárias compostas tem o mesmo objetivo, garantindo que um conjunto de valores seja único dentro da tabela. Isto pode ser visto na tabela **definição**, os campos *id_domínio* e *id_termo* formam a chave primária.

As chaves estrangeiras têm por objetivo relacionar duas tabelas, mantendo assim uma conectividade conceitual entre elas. Por exemplo, na tabela **definição,** o atributo *id_termo* é uma chave estrangeira e faz referência a um valor a um elemento presente na tabela **termo**.

5.3 Funcionalidades

Esta seção apresenta as funcionalidades que estão presentes no sistema de gerenciamento do BDTerm. Elas foram definidas para a interação dos quatro tipos de usuários com o sistema. Estas funcionalidades compreendem a biblioteca de funções do sistema, onde são descritos para cada uma delas: funcionamento, parâmetros, tipos de resposta e usuários.

A biblioteca de funções, é apresentada na

Tabela 5.1 e está classificada conforme a finalidade, que pode ser atualização ou consulta, e sub-dividida por conceitos do sistema. No primeiro caso, são listadas as funções que tem caráter de atualizar o conteúdo informativo do sistema. No segundo, são apresentadas as funções que realizam consultas ao sistema, seja com caráter de busca do conhecimento ou para dar suporte ao sistema.

Quando forem feitas chamadas as funções implementadas no sistema dois tipos de erros podem ocorrer. Erros devidos a falhas internas do sistema e erros gerados pela execução de ações que violem as regras de construção do sistema.

No segundo caso mensagens explicativas dos erros são retornadas, estas mensagens estão explicadas e exemplificadas ao longo das subseções.

As próximas seções detalham todas as funções do sistema.

Tabela 5.1 Biblioteca de funções do sistema de gerenciamento de BDTerms

finalidade	Conceito	Função
atualização	Domínio	cadastrar domínio
		remover domínio
	Usuário	cadastrar usuário
		alterar dados usuário
		cadastrar usuário domínio
		alterar atuação usuário
	Termo	cadastrar termo e definição
		remover termo e definição
		alterar definição de um termo
		alterar validação termo
	Relação Semântica	cadastrar relação semântica
		remover relação semântica
		alterar status relação semântica
consulta	Domínio	retornar domínios
	Usuário	retornar usuários domínio
	Termo	retornar termos públicos
		retornar termos sid
		consultar definição termo
	Relação Semântica	retornar semântica
		retornar relações semânticas público
		retornar relações semânticas sid
		consultar termo relação
		consultar termo relação – busca avançada
		consultar se existe relação entre dois termos
		consultar se a relação entre dois termos está correta
		consultar todas as relações de um termo

5.3.1 Atualização

As funções de atualização têm a intenção de alterar o conteúdo informativo do sistema: inserindo, alterando e/ou removendo informações. Ao serem executadas, estas funções retornam, respostas afirmativas ou negativas.

A primeira alternativa significa sucesso da operação e caracteriza-se por retornar uma mensagem que começa com “OK:”, seguida de um expressão explicativa sobre o que acabou de ser executado. Na presença de uma falha, a resposta de erro é apresentada de forma semelhante a anterior, com a diferença de que o que precede a mensagem é a expressão “ERRO:”.

5.3.1.1 Cadastrar Domínio

Permite que um novo domínio, ou seja, que uma nova base de dados terminológica seja inserida no sistema. Na chamada desta função devem ser informados o nome e a descrição (opcional) e se este novo domínio é especialização de outro existente. Quando o domínio não é uma especialização de outro, deve ser informado nulo para o parâmetro “identificadorSuperDomínio”, desta forma o domínio é considerado base.

Esta função retorna uma negativa quando o nome do domínio a ser inserido já existe dentro do mesmo superdomínio, ou quando o identificador informado para o superdomínio for incorreto.

Assinatura:

cadastrarDomínio(nome, descrição, identificadorSuperDomínio).

Mensagens de Retorno para a Interface:

OK: - o domínio foi cadastrado.

ERRO: - nome do domínio deve ser informado.

ERRO: - o identificador do superdomínio está incorreto.

ERRO: - o domínio já está cadastrado.

Usuários que executam a função:

administrador.

5.3.1.2 Remover Domínio

Remove um domínio presente no sistema. Todos os domínios que herdam o que será removido perdem a referência que fazem a este e passam a ser tratados como domínios base.

Esta função retorna uma negativa quando for informado um identificador de domínio que não existe, ou quando um dos subdomínios violar a regra de unicidade de nome, pelo fato de já haver um domínio base com o mesmo nome.

Assinatura:

removerDomínio(identificadorDomínio)

Mensagens de Retorno para a Interface:

OK: o domínio foi removido.

ERRO: o domínio informado não existe.

ERRO: um dos subdomínios já existe como base.

Usuários que executam a função:

administrador.

5.3.1.3 Cadastrar Usuário

Adiciona um novo usuário ao sistema. Devendo ser informado o nome (opcional), e-mail e senha. Somente os dois últimos parâmetros se fazem obrigatórios, pois são as informações básicas para o usuário acessar o sistema.

Esta função retorna uma negativa quando o e-mail informado, já fizer parte do sistema e quando um dos parâmetros obrigatórios não se faz presente.

Assinatura:

cadastrarUsuario(nome, email, senha).

Mensagens de Retorno para a Interface:

OK: o usuário foi inserido.

Erro: email e senha são informações obrigatórias.

Erro: o usuário já faz parte do sistema.

Usuários que executam a função:

administrador e *UPG.*

5.3.1.4 Alterar Dados Usuário

Permite que um usuário altere os seus dados cadastrais, podendo ser esta alteração qualquer uma de suas próprias informações.

Esta função retorna uma negativa quando o e-mail informado, já fizer parte do sistema (está cadastrado para outro usuário) e quando um dos parâmetros obrigatórios não se faz presente. O mesmo ocorre quando o identificador do usuário informado não exista.

Assinatura:

alterarDadosUsuário (identificadorUsuário,nome,email,senha).

Mensagens de Retorno para a Interface:

OK: os dados foram alterados;

ERRO: o usuário não existe no sistema.

ERRO: o e-mail informado já pertence a outro usuário.

Usuários que executam a função:

UPG e *UPC* - somente o proprietário das informações.

5.3.1.5 Cadastrar Usuário Domínio

Relaciona um usuário cadastrado ao sistema a um determinado domínio, informando qual o tipo de atuação que este deve exercer dentro do sistema: *UPG* ou *UPC*.

Esta função retorna uma negativa quando o identificador do usuário e/ou o identificador do domínio não existem ou quando o usuário já estiver relacionado ao domínio. Também é gerado erro, quando o valor da atuação informado for diferente de “C” ou “G”.

Assinatura:

cadastrarUsuárioDomínio(identificadorUsuário,identificadoDomínio,atuação).

Mensagens de Retorno para a Interface:

Verdadeiro – o usuário foi relacionado ao domínio.

ERRO: o usuário já está definido dentro deste domínio.

ERRO: o usuário e/ou domínio não existem.

ERRO: a atuação escolhida dever ser C – colaborador ou G - gerente.

ERRO: todos os parâmetros são obrigatórios.

Usuários que executam a função:

administrador, *UPG* e *UPC.*

5.3.1.6 Alterar Atuação do Usuário

Realiza a troca de função dentro de um domínio do usuário. Nesta primeira versão, esta função permite somente a troca de “UPC” para “UPG”, por não se ter uma regra sobre o tratamento contrário.

Esta função retorna uma negativa quando o usuário não estiver relacionado ao domínio ou quando um dos parâmetros for omitidos durante a execução da função.

Assinatura:

alterarAtuacaoUsuário(identificadorUsuário,identificadoDomínio).

Mensagens de Retorno para a Interface:

OK: o tipo de atuação do usuário foi alterada.

ERRO: o usuário não se encontra relacionado ao domínio.

ERRO: identificador do usuário e do domínio são obrigatórios.

Usuários que executam a função:

administrador, *UPG* .

5.3.1.7 Cadastrar Termo e Definição

Esta função realiza a inserção de um termo e sua definição em um determinado domínio, sendo que o processo ocorre em duas partes.

Primeiro é verificado se o termo já se encontra na tabela **Termo**. Caso afirmativo o identificador do termo é retornado, do contrário o termo é inserido nesta tabela e o seu identificador é recuperado.

A segunda etapa desta função insere a definição (opcional) do termo dentro da tabela **Definição** relacionando esta informação com o identificador do termo e do domínio. Também é adicionada a finalidade do termo, ou seja, se este está sendo inserido para o conteúdo público (P) ou para definições associadas aos sistemas de integração (SID). Essa verificação é feita da seguinte forma, o último parâmetro da função deve ser fixo e conter o seguinte valor “current_user”. Desta forma é possível saber qual o usuário que chamou a função, porque depois que esta começar a ser executada, o usuário muda, passando a ser o administrador, devido ao fato de que somente este usuário tem permissão para escrever nas tabelas.

Esta função retorna uma negativa quando o termo ou o identificador do domínio forem omitidos. Assim como se o termo já estiver definido dentro do sistema e ou for informado um domínio que não exista. Erros também podem ser gerados quando a finalidade violar a restrição de valores (diferente de público – “P” ou de SID – “S”), caso esta seja omitida o valor padrão será “P”.

Assinatura:

cadastrarDefiniçãoTermo
(termo,definição,identificadodomínio,finalidade,current_user).

Mensagens de Retorno para a Interface:

OK: a definição do termo foi inserida.

ERRO: termo e domínio são valores obrigatórios.

ERRO: o domínio especificado não existe.

ERRO: o termo já está definido dentro deste domínio.

ERRO: a finalidade dever ser P – público ou S – SID.

Usuários que executam a função:

ASID, UPG, UPC e *UPP* .

5.3.1.8 Remover Termo e Definição

Esta função remove a relação existente entre um termo e um domínio, não removendo o termo propriamente dito do sistema, devido ao fato que este pode estar associado a outros domínios. As relações semânticas que existiam entre este termo e os demais presentes no domínio são removidas em cascata.

Esta função retorna uma negativa quando a relação entre o termo e o domínio não exista, ou quando pelo menos um dos parâmetros for omitido na chamada da função.

Assinatura:

removerDefiniçãoTermo(identificadorTermo,identificadodomínio).

Mensagens de Retorno para a Interface:

OK: a definição do termo foi removida.

ERRO: o identificador do termo e/ou domínio não foi informado.

ERRO: a relação entre o termo e o domínio não existe.

Usuários que executam a função:

ASID, *UPG* e *UPC*.

5.3.1.9 Alterar Definição de um Termo

Esta função permite que a definição de um termo possa ser alterada. Mas não é possível alterar o domínio ao qual o termo foi inserido e nem alterar o status do termo. Como somente os usuários especialistas podem executar esta função, o status do termo passa a ser verdadeiro, independentemente do seu valor anterior.

Esta função retorna uma negativa quando a relação entre o termo e o domínio não existe, ou caso um dos identificadores seja omitido, durante a sua chamada.

Assinatura:

alterarDefiniçãoTermo(identificadorTermo,identificadodomínio,novaDefinição).

Mensagens de Retorno para a Interface:

OK: a definição do termo foi alterada;

ERRO: o identificador do termo e/ou domínio não foi informado.

ERRO: a relação entre o termo e o domínio não existe.

Usuários que executam a função:

ASID, *UPG* e *UPC*.

5.3.1.10 Alterar Validação do Termo

Permite que a qualquer momento o *status* da definição de um termo dentro de um domínio possa ser alterado para “verdadeiro”, ou seja, define-se que este termo está definido de forma correta dentro do domínio. Esta função não tem a necessidade de ser executada para termos que tem *finalidade* valorada com “SID”, uma vez que estes termos são considerados como verdadeiros para o *SID*.

Esta função retorna uma negativa quando a relação entre o termo e o domínio não existe, ou caso um dos identificadores seja omitido, durante a sua chamada.

Assinatura:

alterarValidaçãoTermo(identificadorTermo,identificadodomínio).

Mensagens de Retorno para a Interface:

OK: o status foi alterado para verdadeiro.

ERRO: o identificador do termo e/ou domínio não foi informado.

ERRO: a relação entre o termo e o domínio não existe.

Usuários que executam a função:

UPG e *UPC*.

5.3.1.11 Cadastrar Relação Semântica

Esta função permite que seja inserida uma relação semântica entre dois termos dentro de um domínio. Os dois termos a relacionar devem estar presentes sob o mesmo domínio. Quando esta função é executada, duas entradas na

tabela de relacionamentos são inseridas, uma para cadastrar o termo1 em relação ao termo2 e outra para cadastrar o termo2 em relação ao termo1. Desta forma os relacionamentos que envolvem hierarquia, devem ter um tratamento diferenciado, para que a informação do relacionamento possa ser lida de ambos os lados.

É permitido que um *UPP,* ao inserir um novo termo no sistema relacione este semanticamente a um termo já existente. Por isso a relação semântica terá que ser validada, a exemplo do que ocorre com a definição do termo.

Uma relação também pode ser inserida por um *ASID,* não necessitando de validação neste caso. Da mesma forma que ocorre em cadastrar termo, esta função possui um parâmetro fixo “current_user”.

Esta função retorna uma negativa quando pelo menos um dos termos não estiver associado ao domínio desejado ou quando um dos parâmetros for omitidos durante a chamada da função. Assim como quando for passado uma semântica que não exista no sistema.

Assinatura:

cadastrarRelaçãoSemântica(identificadorTermo1,identificadorTermo2,identificadoDominio,identificadorSemantica).

Mensagens de Retorno para a Interface:

OK: a relação entre os termos foi inserida.

ERRO: um dos parâmetros foi omitido na chamada da função.

ERRO: os dois termos devem pertencer ao domínio.

ERRO: um dos parâmetros informados não existe.

ERRO: esta relação já foi cadastrada.

Usuários que executam a função:

ASID,UPG ,*UPC* e *UPP*.

5.3.1.12 Remover Relação Semântica

Permite que uma relação semântica entre dois termos dentro de um domínio possa ser removida. Pela forma como foi definido o sistema, a relação entre termo1 e termo2 deve ser removida, bem como a relação entre termo2 e

termo1.

Esta função retorna uma negativa quando pelo menos um dos parâmetros for omitido ou quando não existir o relacionamento especificado entre o termo1 e o termo2 dentro do domínio desejado.

Assinatura:

removerRelaçãoSemântica(identificadorTermo1,identificadorTermo2,identificadodomínio).

Mensagens de Retorno para a Interface:

OK: a relação semântica foi removida;

ERRO: um dos parâmetros foi omitido na chamada da função.

ERRO: o relacionamento informado não existe

Usuários que executam a função:

ASID,UPG ,*UPC*.

5.3.1.13 Alterar Status de uma Relação Semântica

Permite que a qualquer momento uma relação semântica tenha o seu *status* de visualização alterado para “verdadeiro”, garantindo se que o relacionamento entre os dois termos está correto. O inverso não pode ser feito, ou seja, marcar o status como “falso”.

Não é permitido que o *ASID* execute esta função, já que todo relacionamento cadastrado por este usuário é aceito como verdade.

Esta função retorna uma negativa quando pelo menos um dos parâmetros for omitido ou quando não houver a relação entre termo1 e o termo2 dentro do domínio desejado.

Assinatura:

alterarStatusRelaçãoSemântica(identificadorTermo1,identificadorTermo2,identificadoDomínio).

Mensagens de Retorno para a Interface:

OK: o status da relação foi alterado.

ERRO: um dos parâmetros foi omitido.

ERRO: o relacionamento informado não existe.

Usuários que executam a função:

UPG ,*UPC*.

5.3.2 Consulta

As funções de consulta foram definidas com o intuito de buscar informações no BDTerm, ou seja, em nenhum momento alteram o conteúdo de um domínio. Este tipo de função retorna informações para auxiliar os usuários especialistas na administração dos domínios, bem como informações ligadas a buscas conceituais sobre os termos e seus relacionamentos semânticos.

Sempre que a chamada a função gerar uma exceção, uma mensagem de erro é retornada, informando ao usuário o que ocorreu, esta mensagem é precedida pela expressão “ERRO:”. Quando uma busca não tiver valores para retornar e nenhuma exceção foi gerada durante a sua execução uma mensagem informando a falta de resposta é apresentada. Esta mensagem é precedida pela expressão “ATENÇÂO:”.

As funcionalidades de busca estão definidas ao longo desta seção.

5.3.2.1 Retornar Domínios

Esta função retorna uma lista com os domínios presentes nos sistema, ordenados alfabeticamente.

Esta função retorna uma negativa quando não for encontrado nenhum domínio for encontrado no sistema.

Assinatura:

retornarDomínios().

Mensagens de Retorno para a Interface:

retorna uma lista com os domínios (em caso de sucesso).

ATENÇÃO: nenhum domínio foi encontrado no sistema.

Usuários que executam a função:

administrador, ASID,UPG ,*UPC* e *UPP*.

5.3.2.2 Retornar Usuários de um Domínio

Esta função retorna uma lista com os usuários que fazem parte de um determinado domínio, incluindo a sua atuação dentro do domínio especificado. Esta lista é ordenada alfabeticamente pelo nome do usuário.

Esta função retorna uma negativa quando o domínio informado não exista no BDTerm ou quando não existir nenhum usuário relacionado ao domínio. Se o parâmetro for omitido também é gerada uma mensagem de erro.

Assinatura:

retornarUsuáriosDomínios(identificadorDomínio).

Mensagens de Retorno para a Interface:

retorna uma lista com os usuários do domínio (em caso de sucesso).

ATENÇÃO: não foram encontrados valores que satisfação a consulta.

ERRO: o parâmetro foi omitido.

Usuários que executam a função:

administrador, UPG, UPC.

5.3.2.3 Retornar Termos Públicos

Esta função retorna todos os termos que tem a sua finalidade definida como “P” dentro de um domínio. A lista é retornada por ordem alfabética do nome do termo, incluindo, entre os valores retornados, o status da definição do termo.

Esta função retorna uma negativa quando o domínio informado não existir ou não possuir termos associados. Se o parâmetro for omitido também é gerada uma mensagem de erro.

Assinatura:

retornarTermosPúblicos(identificadorDomínio).

Mensagens de Retorno para a Interface:

retorna uma lista com todos os termos do domínio que tem contexto público (em caso de sucesso).

ATENÇÃO: o domínio informado não existe e/ou nenhum termo está associado a este.

ERRO: o parâmetro foi omitido.

Usuários que executam a função:

UPG,UPC.

5.3.2.4 Retornar Termos SID

Esta função retorna todos os termos que tem a sua finalidade definida como “SID” dentro de um domínio. A lista retornada está pela ordem alfabética. Nesta situação, como todos os termos têm *status* identificado como verdadeiro, este atributo é desconsiderado.

Esta função retorna uma negativa quando o domínio informado não existir ou não possuir termos associados a este. Se o parâmetro for omitido também é gerada uma mensagem de erro.

Assinatura:

retornarTermosSID(identificadorDomínio).

Mensagens de Retorno para a Interface:

retorna uma lista com todos os termos do domínio, que tem contexto para integração de dados (em caso de sucesso)

ATENÇÃO: o domínio informado não existe e/ou nenhum termo está associado a este.

ERRO: o parâmetro foi omitido.

Usuários que executam a função:

ASID.

5.3.2.5 Consultar Definição do Termo

Esta função permite realizar a pesquisa da definição de um termo dentro de um domínio. Os parâmetros necessários para esta chamada são o termo e o domínio. Retornando para o usuário o termo que foi pesquisado e a sua definição.

Esta função retorna uma negativa quando o domínio informado não existir ou não possuir termos associados a este. Se pelo menos um dos parâmetros for omitido também é gerada uma mensagem de erro.

Assinatura:

consultarDefiniçãoTermo(termo,identificadorDomínio).

Mensagens de Retorno para a Interface:

retorna o termo e sua definição (em caso de sucesso).

ATENÇÃO: o domínio informado não existe e/ou o termo não está associado a este.

ERRO: todos os parâmetros são obrigatórios.

Usuários que executam a função:

UPP

5.3.2.6 Retornar Semântica

Esta função retorna uma lista com todas as classificações semânticas presentes no BDTerm.

Esta função retorna uma negativa quando não houver nenhuma classificação semântica registrada no BDTerm.

Assinatura:

retornarClassificaçõesSemânticas().

Mensagens de Retorno para a Interface:

retorna uma lista com todos classes semânticas do sistema (em caso de sucesso).

ATENÇÃO: nenhuma definição semântica está cadastrada.

Usuários que executam a função:

Administrador, ASID, UPG, UPC e *UPP.*

5.3.2.7 Retornar Relações Semânticas Público

Esta função retorna uma lista com os termos e suas relações semânticas diretas dentro de um determinado domínio. Esta lista retorna somente as relações que envolvem termos definidos como finalidade “P”. A listagem está organizada de forma alfabética, pelo nome do termo2, uma vez que o termo1 é sempre o mesmo.

Esta função retorna uma negativa quando o domínio não existir e/ou não possuir nenhuma relação semântica associada ao termo1. O mesmo ocorre quando pelo menos um dos parâmetros for omitido.

Assinatura:

retornarRelaçãoSemânticaPúblico(identificadorTermo1,identificadorDomínio).

Mensagens de Retorno para a Interface:

retorna uma lista com todos os termos e suas relações semânticas diretas (em caso de sucesso).

ATENÇÃO: o domínio informado não existe e/ou o termo não está associado a este.

ERRO: todos os parâmetros são obrigatórios.

Usuários que executam a função:

UPG, UPC.

5.3.2.8 Retornar Relações Semânticas SID

Esta função retorna uma lista com os termos e suas relações semânticas diretas dentro de um determinado domínio. Esta lista retorna somente as relações que envolvem termos definidos como finalidade “SID”. A lista está

organizada de forma alfabética pelo nome do termo2, uma vez que o termo1 é sempre o mesmo.

Esta função retorna uma negativa quando o domínio não existir e/ou não possuir nenhuma relação semântica associada ao termo1. O mesmo ocorre quando pelo menos um dos parâmetros for omitido.

retornarRelaçãoSemânticaSID(identificadorTermo1,identificadorDomínio).

Mensagens de Retorno para a Interface:

retorna uma lista com todos os termos e suas relações semânticas diretas (em caso de sucesso).

ATENÇÃO: o domínio informado não existe e/ou o termo não está associado a este.

ERRO: todos os parâmetros são obrigatórios.

Usuários que executam a função:

ASID

5.3.2.9 Consultar Relação Semântica de um Termo

Possibilita consultar uma determinada relação semântica de um termo dentro de um domínio, retornando uma lista com os termos e suas definições.

Esta função retorna uma negativa quando não for possível encontrar informações com os parâmetros informados. O mesmo ocorre quando pelo menos um dos parâmetros for omitido.

Assinatura:

consultarRelaçãoTermo(identificadorTermo1,identificadorDomínio,identificadorRelação).

Mensagens de Retorno para a Interface:

retorna uma lista com os termos e suas definições (em caso de sucesso).

ATENÇÃO: não existe informação relacionada aos parâmetros informados.

ERRO: todos os parâmetros são obrigatórios.

Usuários que executam a função:

UPP

5.3.2.10 Consultar Relação Semântica de um Termo – Busca Avançada

Possibilita consultar uma determinada relação semântica de um termo, com o método de busca avançada, dentro de um domínio. Retornando uma lista com os valores dos termos e suas definições. Esta função tem somente caráter de uso para sistemas de integração de dados.

Esta função retorna uma negativa quando não for possível encontrar informações com os parâmetros informados. O mesmo ocorre quando pelo menos um dos parâmetros for omitido.

Assinatura:

consultarRelaçãoTermoAvançado(identificadorTermo1,identificadorDomínio ,identificadorRelação).

Mensagens de Retorno para a Interface:

retorna uma lista com os termos e suas definições (em caso de sucesso).

ATENÇÃO: não existe informação relacionada aos parâmetros informados.

ERRO: todos os parâmetros são obrigatórios.

Usuários que executam a função:

SID

5.3.2.11 Consultar se Existe uma Relação Semântica Entre Termos

Esta função busca em um domínio uma relação semântica qualquer entre dois termos. A consulta é feita na forma de busca completa e retorna o tipo de relação entre os termos.

Esta função retorna uma negativa quando não for possível encontrar informações com os parâmetros informados. O mesmo ocorre quando pelo menos um dos parâmetros for omitido.

Assinatura:

consultarExisteRelaçãoTermos(identificadorTermo1,identificadorTermo2,identificadorDomínio).

Mensagens de Retorno para a Interface:

retorna a relação existente entre os termos (em caso de sucesso).

ATENÇÃO: não existe informação relacionada aos parâmetros informados.

ERRO: todos os parâmetros são obrigatórios.

Usuários que executam a função:

SID

5.3.2.12 Consultar se uma Determinada Relação Entre os Termos Está Correta

Esta função busca em um domínio por uma determinada relação semântica entre dois termos. A consulta é feita na forma de busca completa e retorna verdadeiro, caso a relação exista e falso do contrário.

Esta função retorna uma negativa quando não for possível encontrar informações com os parâmetros informados. O mesmo ocorre quando pelo menos um dos parâmetros for omitido.

Assinatura:

consultarRelaçãoTermosCorreta(identificadorTermo1,identificadorTermo2,identificadorDomínio,identificadorSemântica).

Mensagens de Retorno para a Interface:

Verdadeiro (quando existir a relação).

Falso (quando não existir a relação).

ERRO: todos os parâmetros são obrigatórios .

Usuários que executam a função:

SID

5.3.2.13 Consultar Todas as Relações de um Termo

Esta função executa a busca por todos os tipos de relacionamentos que um termo possui em um domínio, a verificação é feita na forma de busca completa. Esta função tem o objetivo de retornar o máximo de informação de uma única vez. Os dados retornados são ordenados pelo tipo de relação e posteriormente pela ordem alfabética do nome do termo2.

Esta função retorna uma negativa quando não for possível encontrar informações com os parâmetros informados. O mesmo ocorre quando pelo menos um dos parâmetros for omitido.

Assinatura:

consultarTodasRelaçõesTermo(identificadorTermo,identificadorDomínio).

Mensagens de Retorno para a Interface:

retorna os termos,sua definições e o tipo de relacionamento (em caso de sucesso).

ATENÇÃO: não existe informação relacionada aos parâmetros informados.

ERRO: todos os parâmetros são obrigatórios.

Usuários que executam a função:

SID

5.4 Interface

Esta seção apresenta a interface do sistema e durante a sua descrição e não são feitos maiores detalhes sobre o seu funcionamento (segurança, mensagens de erro, entre outras alternativas). A interface permite um mecanismo de interação simples entre os usuários e o sistema, para que desta forma fique melhor compreendido sobre as atribuições de cada usuário.

A interface gráfica é um aplicativo Web que transforma as atividades de administração dos domínios e busca da informação em atividades mais simples. Ela abrange os usuários *ASIDs, UPGs, IPCs* e *UPPs.* O único que não dispõe de acesso a esta é o usuário *SID*, por este se tratar de um mecanismo de interação

automático (usuário não-humano). Cabe a este estabelecer a conexão com o sistema e consultar as funções que foram definidas para a sua interação e processar as respostas provenientes.

Dos quatro tipos de usuários que interagem com a interface, somente o *UPP* não necessita validar *login* e senha para ter acesso as funcionalidades do sistema. Como este usuário acessa o sistema somente para consultar e/ou sugerir um novo termo para o domínio, não há a necessidade de aplicar maiores políticas de restrições além das que já foram definidas (limite de acesso à funções e validação do conteúdo inserido).

O usuário *ASID* é um usuário único, ou seja, existe somente um nome de usuário e senha para todos os acessos. Este interage com todos os domínios, uma vez que as suas ações são sempre de apoio aos mecanismos de integração de dados.

Os usuários *UPG* e *UPC* são os usuários que administram um determinado domínio e o seu papel é determinado conforme o domínio que ele venha a gerenciar. Isto se deve ao fato de que um mesmo usuário pode estar associado a mais de um domínio e ter um papel diferente em cada um destes (mais a frente é apresentando um exemplo sobre esta possibilidade). Desta forma, toda vez que um usuário se autenticar no sistema e escolher um domínio para administrar, cabe a este domínio dizer qual o papel deste usuário e desta forma a interface se

Figura 5.7 Tela de seleção do domínio

Figura 5.8 Tela de login

adaptará, para oferecer somente as funcionalidades disponíveis para este tipo de usuário.

Para apresentar as possíveis adaptações que a interface pode sofrer durante a interação com um usuário é apresentada seguinte situação de exemplo: dois usuários diferentes acessam a tela de inicialização do sistema, Figura 5.8. Considera-se que o primeiro é um *ASID* e o outro um usuário público qualquer, que possui um cadastro no sistema. Após a validação das informações (nome e senha) , estes são direcionados para uma outra tela, Figura 5.7, onde o domínio a ser administrado deve ser selecionado. No caso do *ASID*, todos os domínios presentes no sistema são listados. Para o outro usuário, somente os domínios que este possui um papel definido, como mostra a. Figura 5.7.

Após esta escolha, a interface redireciona o *ASID* para a tela apresentada na Figura 5.9, onde está sendo demonstrada a funcionalidade “listar termos”. Pode ser observado nesta figura que para o *ASID*, os termos estão divididos em duas partes, onde a primeira apresenta os termos com finalidade aos mecanismos de integração de dados e na outra os termos correntes do domínio. Para os usuários *UPG* e *UPC* esta mesma funcionalidade apresentaria somente a

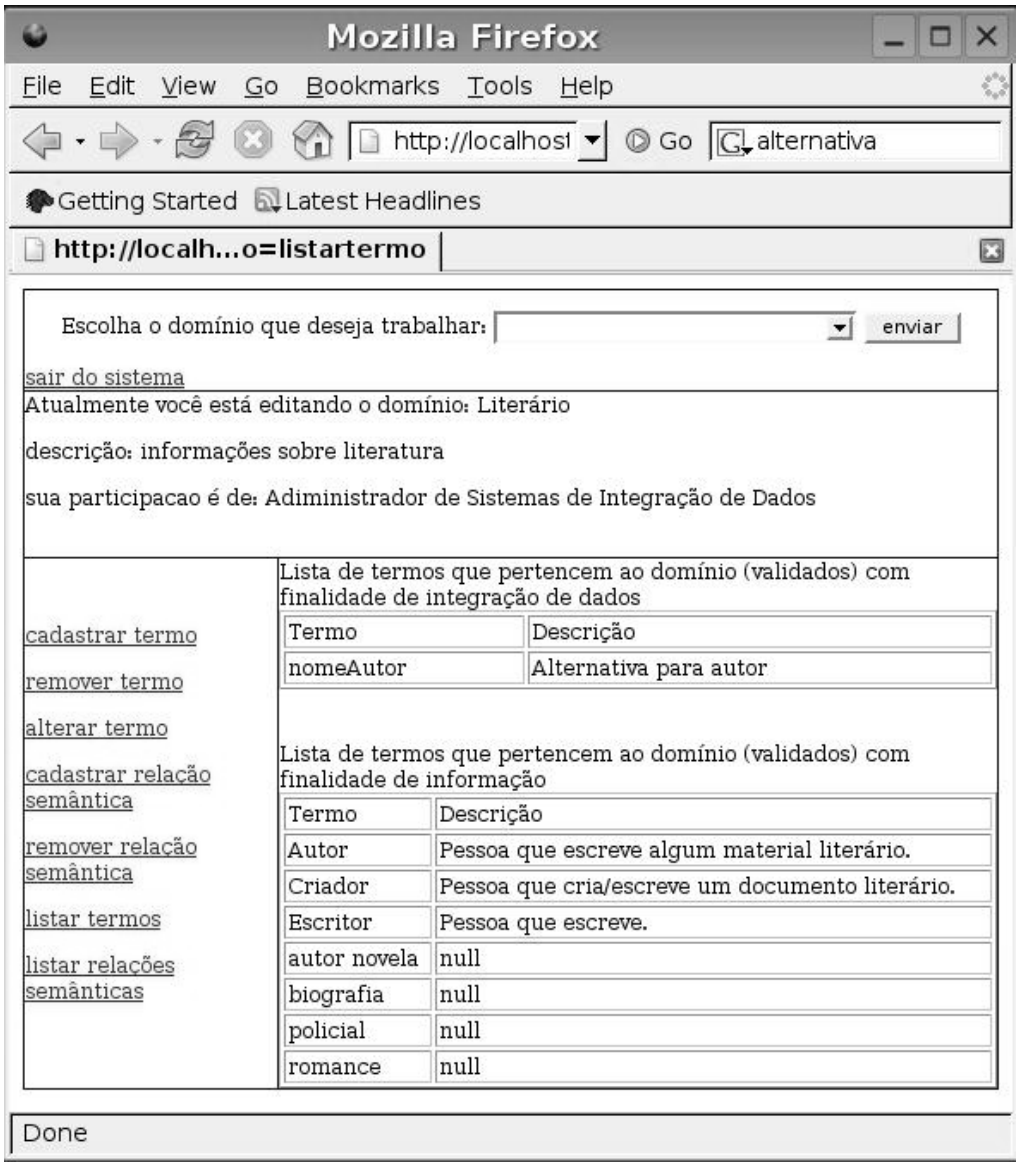

Figura 5.9 Tela com as funcionalidades para o ASID

segunda parte.

Quando o usuário público desejar administrar um domínio que este tem papel de gerenciador este passa a ser referenciado como *UPG* e a interface o redireciona para a tela mostrada na Figura 5.10. Nesta é apresentada a possibilidade de cadastrar uma relação semântica entre dois termos do domínio corrente. Pode ser observado na Figura 5.10 que a caixa de seleção estendida apresenta somente termos que compõe o conteúdo do domínio, nesta figura as

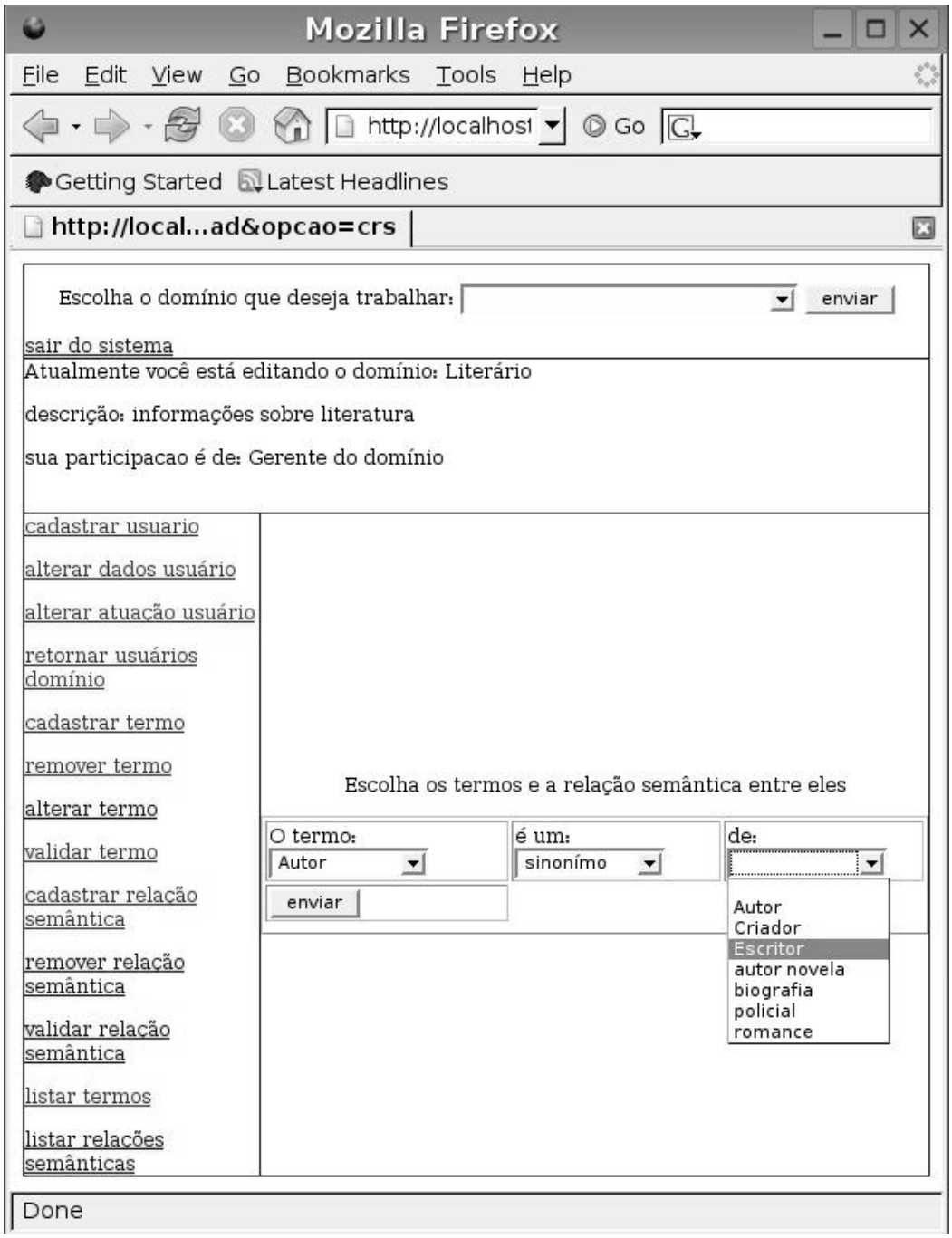

Figura 5.10 Tela com as funcionalidades para o UPG

duas caixas de seleção apresentam o mesmo conteúdo. Se o fosse o *ASID* que estivesse executando esta chamada, a primeira caixa de seleção conteria somente os termos destinados a integração de dados, desta forma procura-se fazer com que termos caracterizados para integração de dados sempre façam

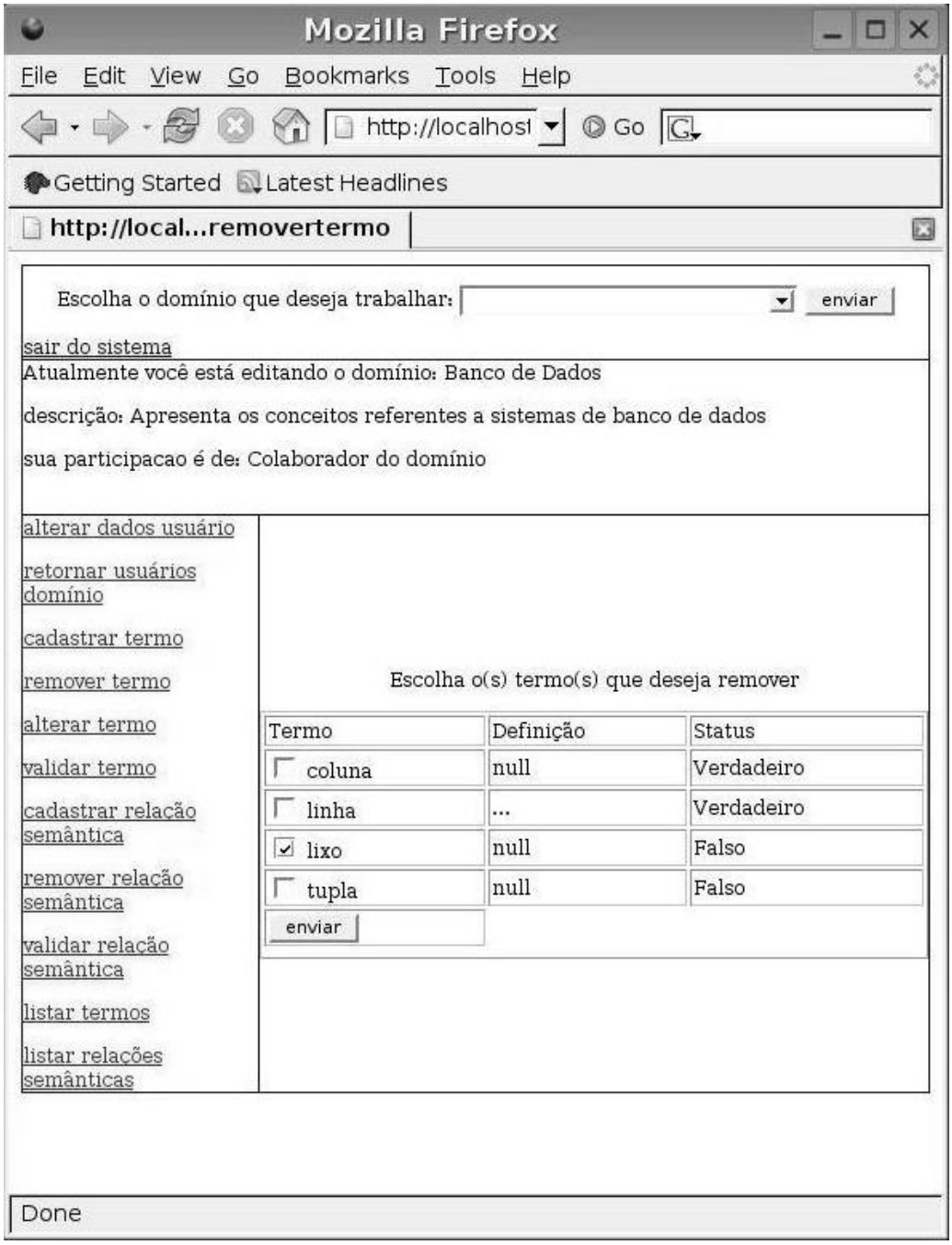

Figura 5.11 Tela com as funcionalidades para o UPG

relação a termos correntes do domínio.

Agora este mesmo usuário que estava com a atribuição de *UPG*, escolhe um outro domínio para administrar, onde a sua atuação é de colaborador, a tela demonstrada pela Figura 5.11, é retornada e este pode executar as funcionalidades previstas para o *UPC*. Esta figura apresenta a funcionalidade

Figura 5.12 Tela com as funcionalidades para o UPP

referente a manutenção do conteúdo do sistema, ou seja, o usuário está removendo um termo que foi inserido de forma errônea. Pode-se observar nesta figura que uma das informações presentes nesta tela é o status do termo no domínio, assim pode-se remover qualquer termo mesmo que este já tenha sido validado por outro usuário.

A interface para um *UPP* é vista na Figura 5.12. Esta é bem simples, contendo campos para inserir o termo a ser procurado e o domínio onde a busca deve ser feita. Apresenta ainda a possibilidade de realizar uma chamada para a inserção de dados. Quando a busca for a funcionalidade acionada pelo usuário a interface irá processar a ação e retornar as informações provenientes do banco de dados, como mostra a Figura 5.13. Nesta é apresentada o termo e sua definição dentro do domínio, no canto esquerdo existe um alista com possíveis relacionamentos semânticos que este termo pode ter . No caso ao se escolher ver uma relação semântica, a interface retornar uma lista com os termos que atendem ao relacionamento desejado com o termo que está sendo pesquisado, o usuário poderá selecionar um dos termos e através de um *link*, será redirecionado

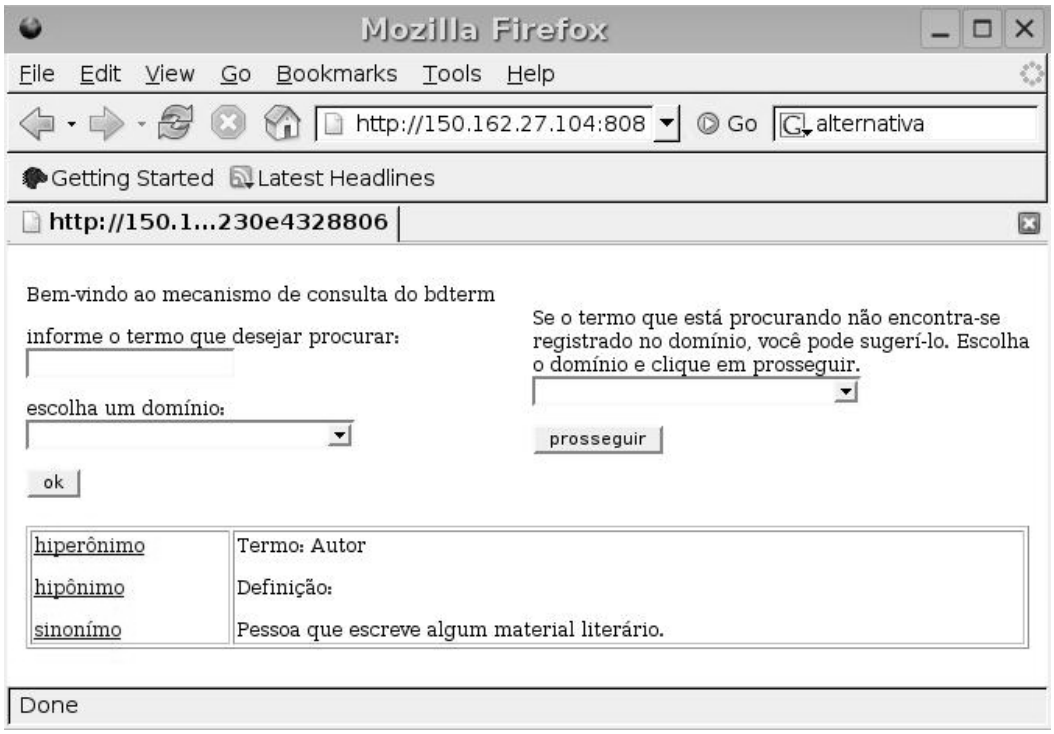

Figura 5.13 Tela com o resultado para uma pesquisa feita pelo UPP

para a tela de resposta da pesquisa, Figura 5.13, agora com a definição do novo termo.

Se o usuário desejar cadastrar um novo termo, este deve escolher o domínio ao qual este termo deve ser inserido e desta forma a interface o redireciona para outra tela, como mostra a Figura 5.14. Onde o termo deve ser inserido juntamente com uma definição, caso o usuário desejar. Pode ainda ser inserida juntamente com esta opção uma relação semântica entre o novo termo e os que já estão contidos no domínio. Esta tela para cadastrar o termo é a mesma para todos os usuários.

Quando for o usuário SID, que desejar realizar consultas ao sistema, este deverá estabelecer a conexão e realizar a chamada a funcionalidade desejada e que tenha sido pré-definida para a sua execução. A Figura 5.15 demonstra um script em PHP, para realizar uma consulta ao BDTerm como se fosse o usuário SID. Neste fragmento de código é estabelecida a conexão, feita a consulta e processada a resposta. Neste exemplo o SID consulta os sinônimos do termo “autor”, e a resposta é apresentada na Figura *5.16*, onde pode ser observada os

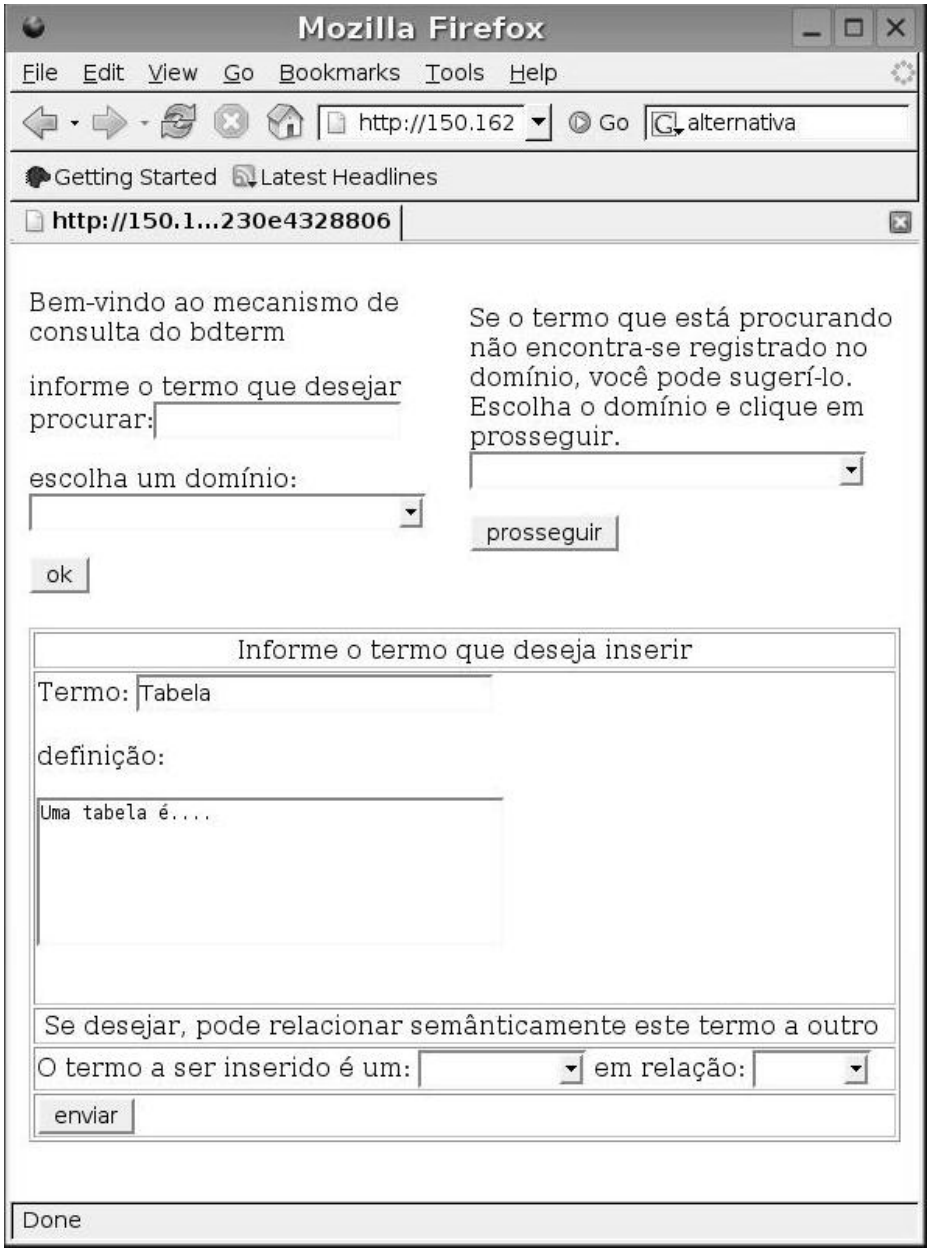

Figura 5.14 Tela com a funcionalidade de “cadastrar termo”

efeitos da busca completa (esta consulta é feita sobre o exemplo apresentado na Figura 5.2), onde pode ser observada na resposta a presença de um termo cadastrado para finalidade de integração de dados “nomeAutor”.

```php
<?
    $parametrosconexao = "host='ip_do_servidor' port=5432 dbname=bdterm user=sid password=****";
    $con = pg_connect($parametrosconexao);
    $sql = "select * from consultarelacaotermoavancado(60,19,1);";
    $aux = pg_query($con,$sql);

    while($aux2[] = pg_fetch_array($aux)){
    }
    array_pop($aux2);
    echo "<pre>";
    print_r($aux2);
?>
```

Figura 5.15 Script em PHP, simulando uma conexão do SID

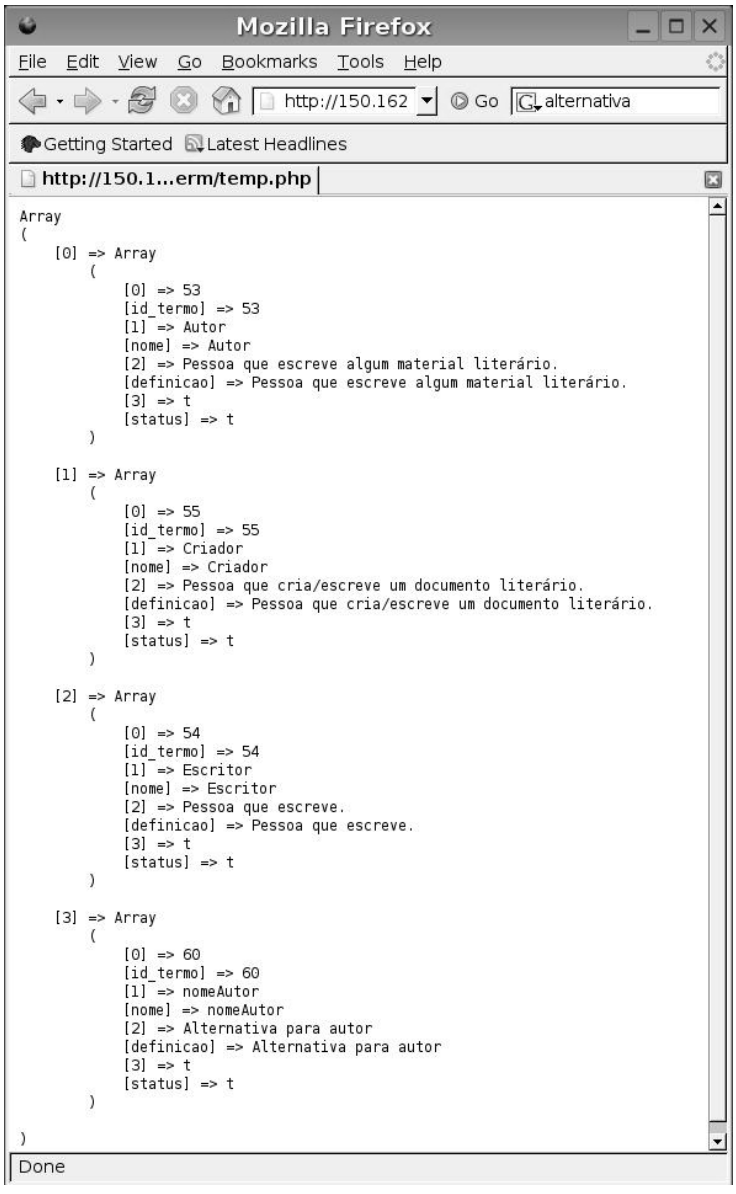

Figura 5.16 Busca feita pelo SID pelos sinônimos de “autor”

6 Conclusão

O corrente trabalho apresenta uma ferramenta cujo propósito principal é o auxílio aos mecanismos de integração de dados, durante a etapa de determinação de correspondências semânticas, para fins de unificação de dados. Todas as decisões tomadas ao longo da etapa de modelagem tiveram este objetivo como foco. Entretanto, o processo de popular a base de dados é o que demonstrava ser mais dispendioso, ou seja, cada usuário responsável por um mecanismo de integração de dados teria que construir a sua própria base de dados para consultas futuras. Desta forma, a solução proposta foi, se a integração de dados ocorre em domínios específicos e estes possuem participantes que estão acostumados aos seus termos, conceitos e relacionamentos, então estes devem ser os usuários responsáveis por popular o BDTerm, sejam eles responsáveis ou não por mecanismos de integração de dados.

Assim o objetivo do trabalho foi estendido. Além de suprir o suporte aos mecanismos de integração de dados, a ferramenta oferece suporte à troca de conhecimento entre membros de um campo do conhecimento e a comunidade em geral. A comunidade pode ajudar, cadastrando as informações que sentir a necessidade de existir dentro de um domínio e seus possíveis relacionamentos semânticos. O que garante a validade da informação é a atuação de especialistas de cada área da informação. Assim sendo, novas categorias de usuários foram incorporados ao sistema e sua aplicabilidade tornou-se mais ampla.

Para estender ainda mais os recursos disponíveis aos mecanismos de integração de dados, é permitido aos grupos de pesquisa sobre este assunto, cadastrarem “termos especiais” aos domínios. Esta informação tem o objetivo de suprir a base com grafias particulares encontradas nos dados a serem integrados. Desta forma, o conhecimento compartilhado belo BDTerm não é somente o que se encontra em uso corrente no domínio, agregando-se a este semânticas particulares encontradas em algumas fontes de dados. Através da rede de relacionamentos semânticos é possível manter a conceituação entre os “termos especiais” com os temos do domínio.

Diante desta justificativa, este trabalho vem a contribuir com a disponibilização de um sistema que permite cadastro e consulta sobre termos e relacionamentos semânticos para o público em geral. Ele permite que mecanismos

de integração de dados utilizem as bases terminológicas como suporte a suas tomadas de decisões, assim como permite que os centros de pesquisa sobre integração de dados troquem conhecimento, sem que estes dados venham a sobrecarregar os demais usuários com informações irrelevantes, ou seja, convenções semânticas específicas de certas fontes de dados.

Como trabalhos futuros, sugere-se o seguinte:

- Melhorar a interface com os usuários, para facilitar ainda mais a interação destes com o sistema , como nível de interatividade do usuário;
- Criar uma interface especial para os mecanismos de integração de dados como por exemplo chamadas a procedimentos remotos;
- Aumentar o número de relações semânticas que o sistema pode abrigar, como por exemplo siglas, abreviaturas, para que desta forma seja possível aumentar a rede de relacionamentos entre os termos;
- Propor novas funcionalidades para aumentar a flexibilidade do sistema, como permitir a re-organização dos domínios ao longo de sua vida útil e não somente no momento de sua criação.

O primeiro protótipo do sistema de BDTerm está disponível no *site* do Grupo de Banco de Dados da UFSC (http://www.grupobd.inf.ufsc.br/) e sua utilização inicial, como salientado anteriormente, será como apoio ao processo de integração de esquemas XML realizado pelo BInXS. BInXS é o principal projeto de pesquisa atualmente em desenvolvimento pelo Grupo de BD.

Referências

ABITEBOUL, Serge**.** **Querying Semi-Structured Data**. Disponível em: <http://heuser.inf.ufrgs.br/cmp170-02.2/1-1/Abiteboul.pdf>. Acesso em: 27 novembro 2004

ANDRADE, Maria Margarida, **Lexicologia, Terminologia**: definições, finalidades, conceitos operacionais. **in** (Oliveira, A. M. P. P; Isquerdo, A. N. **As ciências do Léxico**: lexicologia, lexicografia, terminologia. 2 ed. Campo Grande, Editora UFMS, 2001)

ARTEMIS. Analysis and Reconciliation Tool Environment for Multiple Information System. Disponível em: <http://islab.dico.unimi.it/*ARTEMIS*/> Acesso em: 27 novembro 2004

Bellahsene, Z. **Data integration over the Web**. Disponível em: <http://www.sciencedirect.com/>. Acesso em: 27 novembro 2004-12-03

BIDERMAN, M. T. C., **AS CIÊNCIAS DO LÉXICO**. **in** (Oliveira, A. M. P. P; Isquerdo, A. N. **As ciências do Léxico**: lexicologia, lexicografia, terminologia. 2 ed. Campo Grande, Editora UFMS, 2001)

BOAS, R. M. F. V. **XMLS+Matcher**: Um Método para Identificação de Correspondências entre Esquemas XML Schemas Semânticos. 106 f. Dissertação (Mestrado em Ciencias da Computação) – Departamento de Computação, Universidade Federal do Ceará

BOTELHO, Fabiano Cupertino. **IMPLEMENTAÇÃO DE UM EDITOR ESTRUTURAL PARA ONTOLOGIAS**. 2002/2. 45f. Trabalho de conclusão de curso (Bacharelado em Informática) – Universidade Federal de Viçosa, Viçosa.

BRADLEY, Neil. **TheXML Companion**. 2 ed. Harlow, USA: Addison-Wesley Longman 2000. 435p.

CABRÉ, M. Teresa. **La Terminologia**: Teoria, metodologia, aplicaciones. 1. ed. Barcelona, Editorial Empúres, S. A., 1993.

CORAZZON, Raul. Ontology. A resource guide for philosophers. Disponível em <http://www.formalontology.it/>. Acesso em: 27novembro 2004

GRUBER, Tom. **Virtual Documents**. Disponível em: <http://ksl-web.stanford.edu/people/gruber/virtual-documents/index.html>. Acesso em: 27 novembro 2004

KOWALSKI, Gerald; MAYBURY, M. T. **Information Storage and Retrieval Systems**: Theory and Implematation. 2 ed. USA: Kluwer Academic Publishers, 2000.

KRIEGER, Maria da Graça; FINATTO, Maria José Bocorny. **Introdução à Terminologia**: teoria e prática. 1 ed. São Paulo, Editora Contexto, 2004.

LANCASTER, F. W., **Indexação e Resumos**: Teoria e Prática. 2 ed. Brasília, DF: Brinquet de Lemos, 2004.

MELLO, R. S., *et al*. **Dados Semi-Estruturados**. Disponível em: <http://www.inf.ufrgs.br/~vanessa/artigos/tutorial.pdf >. Acesso em: 27 novembro 2004

MELLO, Ronaldo dos Santos. **Uma Abordagem *Bottom-Up* para a Integração Semântica de Esquemas XML**. 2002. 145f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

NEELY, Steven. **Mobile Computations over Distributed Semistructured Data**. 2003. 205f. Tesis (Doctor of Philosophy) – Department of Computer and Information Science, University of Strathclyde, Glasgow.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Database management systems**. 3rd ed. Boston: McGraw-Hill, 2000.

SAINT-DIZIER, Patrick; VIEGAS, Evelyne. **Computational Lexical Semantics**. 1 ed. Cambridge: Cambridge University Press, 1995.

SIEDLER, Marcelo da Silveira. **Integração de Dados na Web Utilizando o Tamino XML Server**. Disponívlel em: <http://www.cin.ufpe.br/~mss3/artigo1.zip>. Acesso em: 27 novembro 2004
SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados**.. São Paulo: MaKron Books do Brasil, 1999

SILVA, Tércio de Morais Sampaio. **Extração de Informações para Busca**

Semântica na web Baseada em Ontologias. 2003. 79f. Dissertação (Mestrado em Engenharia Elétrica) – Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis.

SOWA, John F. **Ontology**. Disponível em: <http://www.jfsowa.com/ontology/index.htm>. Acessado em: 27 novembro 2004

TANENBAUM, Andrew S. **Redes de Computadores**. 4. ed Rio de Janeiro: Campus; 1999.

WORDNET, A lexical database for the English language. Disponível em: <http://www.cogsci.princeton.edu/~wn/ > Acesso em: 27 novembro 2004

W3C, **World Wide Web Consortium**. Disponível em: <http://w3c.org/ >. Acesso em 03 dezembro 2004.

ANEXO A – ARTIGO

Projeto de uma Base de Dados Terminológica

Fabrício Santos da Silva, Ronaldo dos Santos Mello

INE – Departamento de Informática e Estatística
UFSC – Universidade Federal de Santa Catarina

{fss,ronaldo}@inf.ufsc.br

Resumo. O intercâmbio de dados em documentos XML está crescendo. Devido as suas características novos mecanismos de busca estão sendo propostos, focados na integração de esquemas XML. Esta abordagem necessita de uma ferramenta auxiliar, que resolva conflitos semânticos entre os termos a serem integrados. Este trabalho propõe a criação e construção de uma base de dados terminológica, para dar este suporte aos mecanismos de integração, durante o processo de validação semântica entre os termos. Este sistema permite que o conhecimento possa ser acessado por usuários em geral, sendo que toda a informação é gerenciada por usuários com conhecimento sobre o domínio, seja para a integração de dados ou não.

Palavras-chave: XML, terminologia, banco de dados terminológicos, semântica

Abstract. The interchange of data in documents XML is growing. Had its new characteristics search mechanisms are being considered, target in the integration of XML schema. This boarding needs a auxiliary tool, that resolve semantic conflicts between the terms to be integrated. This work considers the creation and construction of a terminological database, to give this support to the integration mechanisms, during the validation process semantics between the terms. This system allows that the knowledge can be accessed by general users, being that all the information is managed by users with knowledge on the domain, either for the integration of data or not.

KEYWORD: XML, terminological, terminological databases, semantic

1 Introdução

As principais conquistas tecnológicas do século XX se deram no campo da informação (TANENBAUM, 1999). Este crescimento é conseqüência do grande número de computadores espalhados pelo mundo, ao número de usuários conectados na WWW (*World Wide Web* ou simplesmente *Web*) e também a facilidade que as pessoas têm para publicar, compartilhar, ler documentos nos mais diversos formatos e idiomas, com conteúdos dos mais variados domínios.

O crescimento do conteúdo informativo disponível para consulta acabou por gerar novos problemas, no campo de armazenamento, disponibilização e o principal, a recuperação da informação. Os recursos hoje disponíveis para buscar informações, as denominadas “máquinas ou sítios de busca” não são eficientes nem suficientes para indexar todo o conteúdo disponível na Internet relativo a uma consulta desejada (BOTELHO, 2002).

Assim, a recuperação da informação pode se tornar uma tarefa árdua ou, às vezes, até uma grande perda de tempo para o usuário, até que este consiga encontrar um documento que satisfaça as suas necessidades de consulta.

Motivado pelas diversas formas de representar e divulgar a informação através da *Web*, faz-se necessário definir um padrão para a representação dos dados. XML é um formato bastante utilizado para armazenar textos estruturados e semi-estruturados, pretendidos para disseminação e finalmente para publicação, tais como catálogos, formulários (BRADLEY, 2000).

Atualmente existem várias propostas para realizar a integração de esquemas XML. *BInXS* (MELLO,2002) é a proposta na qual este trabalho está inserido e que possui as seguintes características:

- utilização de uma representação conceitual para esquemas XML, com relacionamentos de associação e herança que modelam a intenção semântica dos dados XML.
- um processo de integração semântica destes esquemas conceituais que considera a determinação de equivalências e a resolução de conflitos entre representações semi-estruturadas.
- em particular, são tratados casos específicos na unificação de dois tipos de elementos (texto e estruturados) e elementos com representações alternativas. Um esquema conceitual global é o resultado deste processo, funcionando como mediador entre o usuário e fontes de dados XML na *Web*.

O presente trabalho está relacionado ao suporte necessário à fase de integração semântica do *BInXS*, tendo por objetivo prover um maior suporte automático a esta fase no que se refere à determinação de equivalências semânticas entre conceitos de um mesmo domínio que estão em fontes de dados XML diferentes.

Especificamente, este trabalho propõe a definição de uma base de dados terminológica, bem como formas de interação com esta base, para auxiliar o *BInXS* no processo de integração semântica, onde os termos a serem validados sobre o aspecto de sua intenção semântica (equivalência e ou de hierarquia), estão sob um mesmo domínio, diminuindo a tarefa de um usuário especialista para resolver pequenos conflitos. Estes deixaram de ocorrer pelo fato de que a relação semântica entre os dois termos passa a estar vinculada ao domínio, eliminando-se assim o fato de que o usuário especialista do BInXS, teria para determinar entre as respostas, a que abrangia a sua necessidade.

Para diminuir a tarefa de inserção do conteúdo informativo da base, foi estendida a sua possibilidade de uso. Assim esta base pode ser consultada e administrada por usuários que não tem envolvimento direto com as atividades de integração de dados. Desta forma esta passa a ser um repositório para consulta de termos. Existe ainda a possibilidade de que os usuários responsáveis pelos mecanismos de integração de dados possam inserir termos específicos para este propósito, aumentando assim o poder expressivo de consulta da ferramenta.

Como a integração semântica, se aplica a domínios específicos, um banco de dados terminológico, parece ser a melhor alternativa pra solucionar o problema.

Este trabalho está dividido em outros quatro capítulos. O capítulo dois trata sobre conceitos básicos de dados semi-estruturados. O capítulo três apresenta uma definição de banco de dados terminológicos e os conceitos que estão envolvidos na sua definição. No capítulo quatro será apresentado um resumo sobre o *BInXS*, o processo de integração semântica e os motivos pelo qual um banco de dados terminológico será desenvolvido. O capítulo cinco apresenta o sistema proposto, com a conceituação sobre o problema, modelagens geradas, funcionalidades e interface. O último capítulo apresenta a conclusão sobre este trabalho, considerações e propostas para atividades futuras.

2 Dados Semi-Estruturados e XML

Dados semi-estruturados são dados não convencionais cuja representação pode ser altamente heterogênea mesmo para instâncias em uma mesma fonte de dados (MELLO, 2002)

Dados semi-estruturados podem ser caracterizados por (Abiteboul *apud* MELLO et. al., 2000):

- Estrutura irregular;
- Estrutura implícita;
- Estrutura parcial;
- Estrutura extensa;
- Estrutura evolucionária;
- Estrutura descritiva e não prescritiva;
- Distinção entre estrutura e dados não é clara.

Os documentos XML estão divididos em duas partes. Uma lógica, que permite que o documento seja representado por um elemento, definido pelos metadados (palavra que representa a informação) e que podem conter uma informação associada ou não (um elemento pode ser vazio).

A segunda parte, chamada de física, permite que os componentes do documento, as *entidades*, sejam nomeadas e armazenadas separadamente (BRADLEY, 2000).

XML atende todos os pontos anteriormente citados, sobre as características de dados semi-estruturados. Podendo existir uma DTD (Document Type Definition), ou uma DTD pode ser associada a um dado XML quando esta não pré-definida. Uma DTD, pode ser vista como as especificações das regras de construção de elementos, por exemplo pode se associar uma DTD ao elemento "nome", não permitindo que seja inserido caracteres numéricos. As DTDs possuem regras próprias para a sua construção o que não será apresentado aqui, pois foge ao escopo deste trabalho.

3 Banco de Dados Terminológicos e Conceitos Relacionados

Banco de dados terminológicos ou simplesmente BDTerm, são uma forma de armazenar o conhecimento terminológico em um de banco de dados com uma modelagem voltada às necessidades de um domínio específico (este domínio pode ser o conhecimento de Ciências da Computação ou o conhecimento sobre banco de dados, não havendo uma definição sobre especificidade). Ele utiliza os conceitos de classificação do léxico, porém através de um subconjunto - a Terminologia (BIDERMAN, 2001).

Na seqüência é definido com mais detalhes o conceito de banco de dados terminológicos e outros conceitos relacionados (ontologias e *thesauri*). Para que fique de forma clara a diferença entre os três conceitos e que mais adiante possa ser entendido os motivos da abordagem adotada.

3.1 Ontologias

Ontologia provê critérios para distinguir os tipos de objetos (concretos e abstratos, existência e não existência, real e ideal, independente ou dependente) e suas equivalências (relações, dependências e predicados) (CORAZZON, 2004).

No campo do compartilhamento do conhecimento, ontologia pode significar uma especificação explícita de uma conceituação (GRUBER, 2004). Já segundo (SOWA, 2000) ontologia é um catálogo de tipos de coisas que existem em um domínio de interesse ***D*** sobre a perspectiva de uma pessoa que utiliza uma linguagem ***L*** com o objetivo de falar sobre ***D***. Os tipos em ontologia representam os predicados, significado das palavras ou conceitos e tipos de relações.

3.2 Thesaurus

Thesaurus vem do latim "treasure" e significa uma estrutura similar a um dicionário, que ao invés de armazenar definições, fornece sinônimos e antônimos para as palavras (KOWALSKI E MAYBUNY, 2000). Atualmente um *thesaurus* pode armazenar mais do que simples relações de equivalência ou oposição entre os termos, mantendo estruturas de armazenamento de informações úteis nos dias de hoje.

Thesaurus são listas de termos interligados semanticamente entre si (CABRÉ, 1993), que pode ser considerado um vocabulário dinâmico de termos relacionados semanticamente e genericamente, de um domínio do conhecimento em comum (ISO, 1982).

Pelas duas definições acima citadas, pode-se perceber que, assim como ontologias, *thesauri* estão intimamente ligadas a um domínio específico.

A construção básica de um *thesaurus* envolve três características: (CABRÉ, 1993)

- conteúdo - relação semântica entre as palavras
- estrutura - documento formalizado composto por um vocabulário controlado e dinâmico e de relações conceituais expressas por regras formalizadas;
- função – são instrumentos de controle, que regulam o uso da linguagem natural.

Os *thesauri* podem ter dois tipos de controle, um onde somente uma palavra pode ter a função de referência ao conceito denotado e o outro onde várias palavras podem denotar o mesmo conceito. Em ambos os casos estes termos de referência são chamados *descritores* (ISO, 1982).

A inter-relação entre os descritores é realizada através da formação de uma rede de relações, seja hierárquica e/ou de equivalência e/ou associativa (ISO, 1982):

- Equivalência;
- Hierárquica;
- Associativa.

Desta forma *thesauri* possuem características de uma ontologia (sobre um domínio), mas com restrições e definições próprias (regras de escrita e relacionamento entre as palavras) que em termos de aplicação podem constituir ferramentas muito mais poderás.

3.3 Banco de Dados Terminológicos (BDTerm)

A constituição de uma terminologia própria marca, em toda ciência, o advento ou o desenvolvimento de uma conceituação nova, assinalando assim um momento decisivo de sua história (Benveniste *apud* KRIEGER e FINATTO, 2004).

Banco de dados terminológicos apresentam-se como um sistema de informações interconectados, composto por uma base principal, onde está a lista de termos, e outras bases auxiliares de caráter suplementar, conectadas por meio de informações (KRIEGER e FINATTO, 2004).

BDTerm foram inicialmente concebidos a serviço da tradução, formados por bancos de dados bilíngües ou plurilíngües, mas a possibilidade de armazenar uma grande quantidade de termos e suas respectivas informações, bem como manter este conhecimento atualizado e de fácil divulgação aos usuários é que estão transformando estas bases em importantes ferramentas para diversos profissionais (CABRÉ, 1993).

Na década de 1970, surgiram os primeiros bancos de dados terminológicos, permitindo assim o tratamento automatizado de termos técnico-cientificos e armazenamento de quantidades de informações, antes não imaginadas (KRIEGER e FINATTO, 2004).

4 BInXS – Um Processo de Integração de Esquemas XML

A motivação para realizar a integração de esquemas XML, ou seja, a partir de esquemas locais (que estão distribuídos por várias fontes de dados) gerar um esquema global, se deve ao fato de resolver as diversidades estruturais e semânticas entre eles, permitindo desta forma o acesso integrado a uma grande quantidade de fonte de dados (MELLO, 2002).

A integração de esquemas XML, ou dados semi-estruturados, não pode seguir as mesmas regras definidas para a integração de dados obtidos a partir de banco de dados. Isto porque a natureza dos dados semi-estruturados, é diferente dos dados tradicionais de banco de dados, uma vez que a instância de uma mesma informação em uma mesma fonte de dados podem ser diferentes (MELLO, 2002).

Dentro das possibilidades de realizar o processo de integração de esquemas XML, será considerada a proposta apresentada como *BInXS*, uma vez que este trabalho tem por objetivo apresentar uma ferramenta que auxilie o *BInXS* no processo de integração.

4.1 BInXS

BInXS é parte integrante de uma camada de mediação para acesso a múltiplas fontes de dados XML, com o objetivo de realizar a integração semântica dos esquemas XML das fontes de dados na *Web* e possibilitar a consulta às mesmas através de um esquema global. As informações nesta seção foram retiradas de (MELLO, 2002).

O processo de integração de *BInXS* apresenta duas etapas, conforme mostra a Figura 4.2.

A etapa de Conversão da DTD é responsável pela conversão de cada DTD associada a uma ou mais fontes XML em um esquema conceitual canônico. A utilização deste modelo permite uma abstração da DTD que leva em conta a semântica dos seus dados, facilitando a etapa posterior de integração. Como nem sempre é possível determinar a intenção semântica do dado, a intervenção do usuário pode ser requerida. Outra característica desta etapa é que não somente informação sobre os esquemas é analisada, mas também o conteúdo dos arquivos XML para auxiliar na definição do esquema conceitual.

A etapa de integração semântica é responsável pela integração dos esquemas canônicos, gerando o esquema global e mapeamentos dos seus conceitos para elementos e atributos semanticamente equivalentes em cada DTD. O esquema global é também representado no modelo canônico. A tarefa de integração segue basicamente os passos do processo de integração de esquemas tradicionais, sendo o usuário solicitado para confirmar ou efetivar certas ações que ocorrem nas fases de comparação dos esquemas e unificação. Nesta etapa uma ferramenta chamada *ARTEMIS* é utilizada na determinação de afinidades semânticas entre conceitos de esquemas locais diferentes.

O *ARTEMIS* utiliza de *thesaurus* para

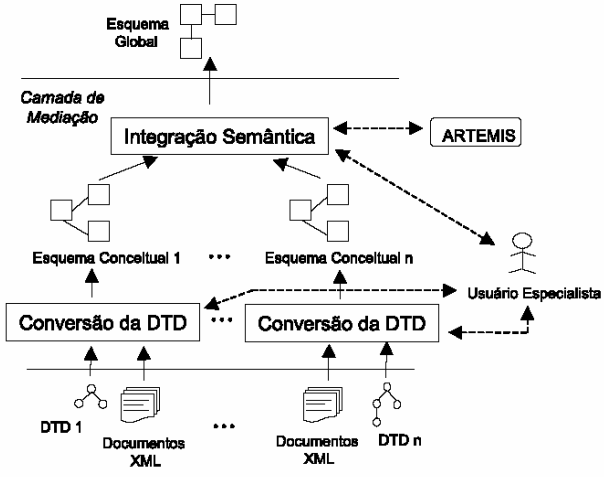

Figura 4. 10 Integração de esquemas XML

geração de esquemas da intenção semântica, através de três alternativas: o usuário cria um *thesaurus* com o seu próprio domínio, usa um domínio extraído do *WordNet* e um terceiro que é um híbrido dos dois primeiros (*ARTEMIS*, 2004). O *ARTEMIS* provê métodos e ferramentas para a integração semântica para base de dados heterogênea.

A utilização de um BDTerm, traz uma série de vantagens para o processo de integração de *BInXS* e contribui com funcionalidades que melhoram o processo de integração do *BInXS*. Primeiramente por evitar a conversão para esquemas conceituais ODL do *ARTEMIS* e elimina o acesso ao WordNet. Vale observar que o *WordNet* possui relação de equivalência, através de sinonímias, mas mesmo assim o *BInXS* consulta o *ARTEMIS* para esta etapa, isto se ao fato de que o *ARTEMIS* permite atribuir valores empíricos, definido pelo usuário do sistema, de quanto um termo é mais semelhante a outro.

5 SISTEMA DE GERENCIAMENTO DE BASE DE DADOS TERMINOLÓGICAS

Conforme foi justificado anteriormente, para dar suporte a etapa de integração semântica utilizada por sistemas de integração de dados, como o *BInXS*, a solução encontrada foi a criação de um Banco de Dados Terminológico (BDTerm) e um conjunto de funcionalidades que permita o gerenciamento destas bases, definindo assim um sistema de gerenciamento de BDTerms.

O principal motivo que leva à adoção desta solução é que termos têm significados diferenciados, conforme um domínio do conhecimento. O processo de integração de documentos XML apresentado pelo *BInXS* interage sempre dentro de uma mesma área afim, ou seja, dentro de um mesmo domínio. Com a disponibilização de BDTerms para domínios específicos aumenta-se a confiabilidade da etapa de determinação de equivalências semânticas em um processo de integração de dados, reduzindo o esforço de validação por parte de um usuário especialista.

5.1 Arquitetura

Foi definida uma arquitetura para permitir que os usuários interajam com o BDTerm de maneira adequada. Esta arquitetura é mostrada na Figura 5.1. Nesta são previstos três tipos de usuários (definidos a seguir) para interagir com o sistema através de uma camada (biblioteca) de funções.

O usuário *ASID* (Administrador de Sistemas de Integração de Dados) é um especialista em sistemas de integração de dados, tendo permissão para inserir informações dentro de uma BDTerm. Estas têm a intenção de aumentar o conteúdo do domínio, melhorando assim o poder de determinação de equivalências semânticas por parte de um *SID.*

O usuário público é responsável pelo gerenciamento do conteúdo de um domínio de uma BDTerm, estando subdividido em três sub-categorias: *gerenciador* (UPG), *colaborador* (UPC) e *participante* (UPP). O *gerenciador* é o responsável por um determinado domínio, podendo inserir informações, validar conceitos, relações e inserir novos usuários, sejam estes, gerenciadores ou colaboradores. O *colaborador* pode somente inserir ou validar as informações presentes no domínio de qual faz parte. Já o *participante* pode realizar consultas a um determinado domínio e realizar a inserção de informações.

5.2 Projeto do Sistema e do Banco de Dados

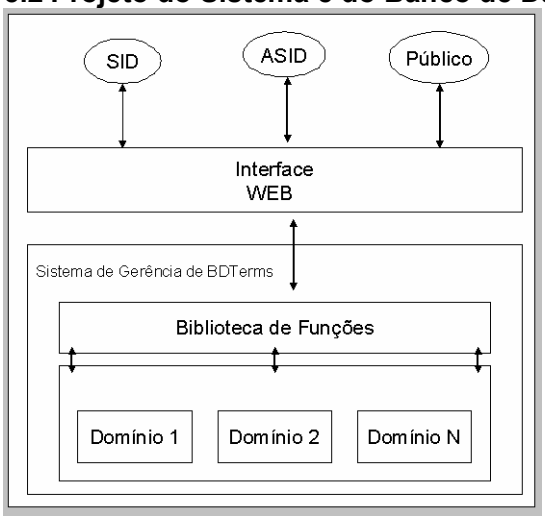

Figura 5.17 Arquitetura

Esta seção apresenta os requisitos e os fatos do mundo real relevantes para o domínio do problema

5.3 Descrição de Requisitos do Sistema

O conceito sobre termo apresentado pela terminologia nos permite deduzir que um termo possui somente uma definição dentro de um determinado domínio. Com base nesta afirmativa é que o sistema apresentado neste trabalho foi proposto.

No entanto, esta afirmativa nem sempre pode ser empregada, pois um termo dentro de uma área específica do conhecimento pode estar associado a mais de uma definição. Isto é comum quando novos termos vão surgindo e a sua conceituação ainda esteja se moldando conforme as especificações definidas pelos pesquisadores da área.

Como exemplo, podemos citar o conceito de SGBD (Sistemas de Gerenciamento de Banco de Dados), apresentados por dois autores diferentes. Segundo (Silberschatz, 1999) um SGBD consiste em uma coleção de dados inter-relacionados e em um conjunto de programas para acessá-los. Para (Ramakrishnan, 2000) SGBD é um programa para ajudar na manutenção e utilização de grandes coleções de dados. Existe uma diferença sutil nas duas definições, mas, no geral, carregam a mesma informação.

Desta forma, pode-se descrever um termo que apresente duas definições aparentemente iguais, mas que com pontos de vista diferenciados para dois autores consagrados sobre o assunto. Para tratar o problema, pode ser utilizado o seguinte tratamento:

Inserir dois ou mais significados para um mesmo termo, deixando a responsabilidade para o usuário de distinguir a conceituação correta dentro do texto da definição do termo (recomendado);

Cadastrar o termo o mais abrangente e para cada conceituação diferenciada inserir novos termos com o mesmo nome, mas que apresente alguma informação adicional ao

nome, mantendo assim a unicidade da informação. Depois com a utilização de relações semânticas (apresentadas mais à frente) é possível manter a inter-relação entre os termos.

5.3.1 Modelagem Conceitual

Um modelo conceitual registra os dados que podem aparecer no banco de dados, sem a preocupação de como estes dados estão estruturados em um SGBD (HEUSER, 2004). Desta forma, o que importa neste modelo é o que pode ser abstraído do problema proposto em termos de dados, permitindo que esta informação sirva como um ponto de referência entre a realidade e o que será mantido dentro de um banco de dados.

A modelagem conceitual proposta neste trabalho para implementar o sistema pode ser observada no DER da Figura 5.3. Este foi construído de acordo com a descrição dos requisitos apresentado na seção anterior.

A entidade **Domínio** corresponde a um conjunto de BDTerms, mantendo informações sobre as BDTerms presentes no sistema. Esta possui um auto-relacionamento, **herda**, desta forma é feita a seguinte leitura: "*um domínio pode ser herdado por zero ou muitos domínios*" e "*um domínio pode herdar zero ou somente um domínio*". Esta entidade possui uma relação com **Usuário**, desta forma: "*zero ou mais usuários atuam em um domínio, com um determinado tipo (G ou C) de atuação*" e "*um domínio sofre a interação de zero ou mais tipos de usuários"*.

A entidade **Termo** contém as dados sobre os termos que estão inseridos no sistema, não carregando nenhuma informação além do nome e de um identificador. Através do relacionamento **definição** desta entidade com **Domínio** é possível fazer a seguinte inferência: "*um termo possui somente uma definição dentro de um ou muitos domínios e esta definição possui um status (verdadeiro ou falso) e uma finalidade (P ou SID)*" e "*um domínio possui a definição de zero ou muitos termos, cada qual com um status e uma finalidade particular*".

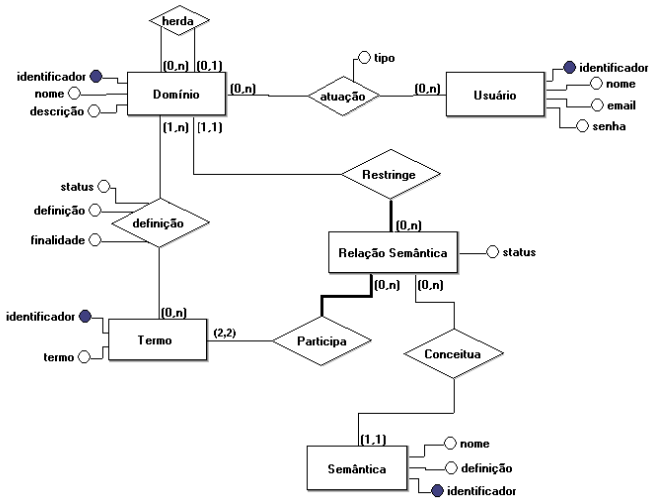

Figura 5. 18 Modelagem conceitual do sistema e restrições de integridade associadas

A entidade **Relação Semântica** armazena toda a informação que constrói a rede de relacionamentos entre os termos em um domínio e interage com outras três entidades: **Termo**, **Domínio** e **Semântica** (relaciona as classes semânticas presentes no BDTerm). Esta parte da modelagem pode ser entendida da seguinte maneira: "*uma relação semântica é composta pela participação de dois e somente dois termos, que estão restritos a um e somente um domínio e tem um e somente um conceito semântico e tem um status (verdadeiro ou falso), para garantir a sua corretude"* bem como: *"um termo participa de zero ou muitas relações semânticas, um domínio restringe zero ou muitas relações semânticas e a semântica conceitua zero ou muitas relações semânticas, todas estas informações associadas a um status"*.

5.3.2 Modelagem Lógica

Esta modelagem se caracteriza por ser a descrição do banco de dados no nível do SGBD, ou seja, a modelagem lógica já é dependente do tipo de SGBD utilizado. Basicamente transforma-se uma entidade em uma relação (tabela), utilizando-se o mesmo nome. Os atributos da modelagem conceitual serão os atributos (colunas) do modelo lógico.

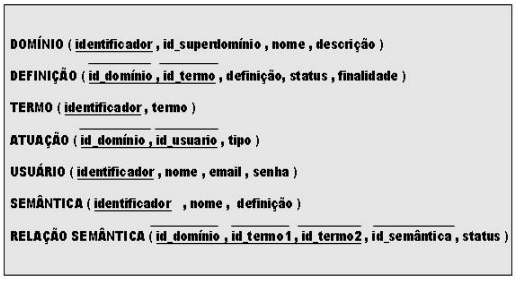

Figura 5. 19 Modelo lógico

O relacionamento existente entre duas entidades deve ser analisado em separado, pois é possível ter mais de um tratamento para acomodar as informações presentes.

Ficando o resultado do modelo lógico representado na Figura 5. 19, onde os atributos sublinhados são as chaves primárias e os atributos com um traço sobre o nome, representam as chaves estrangeiras.

5.4 Funcionalidades

As funcionalidades que estão presentes no sistema de gerenciamento do BDTerm foram definidas para a interação dos quatro tipos de usuários com o sistema. Estas funcionalidades compreendem a biblioteca de funções do sistema, onde são descritos para cada uma delas: funcionamento, parâmetros, tipos de resposta e usuários.

A biblioteca de funções, é apresentada na

Tabela 5.1 e está classificada conforme a finalidade, que pode ser atualização ou consulta, e sub-dividida por conceitos do sistema. No primeiro caso, são listadas as funções que tem caráter de atualizar o conteúdo informativo do sistema. No segundo, são apresentadas as funções que realizam consultas ao sistema, seja com caráter de busca do conhecimento ou para dar suporte ao sistema.

Tabela 5. 2 Biblioteca de funções do sistema de gerenciamento de BDTerms

finalidade	conceito	Função
atualização	Domínio	cadastrar domínio
		remover domínio
	Usuário	cadastrar usuário
		alterar dados usuário
		cadastrar usuário domínio
		alterar atuação usuário
	Termo	cadastrar termo e definição
		remover termo e definição
		alterar definição de um termo
		alterar validação termo
	Relação Semântica	cadastrar relação semântica
		remover relação semântica
		alterar status relação semântica
consulta	Domínio	retornar domínios
	Usuário	retornar usuários domínio
	Termo	retornar termos públicos
		retornar termos sid
		consultar definição termo
	Relação Semântica	retornar semântica
		retornar relações semânticas público
		retornar relações semânticas sid
		consultar termo relação
		consultar termo relação – busca avançada
		consultar se existe relação entre dois termos
		consultar se a relação entre dois termos está correta
		consultar todas as relações de um termo

5.5 Interface

A interface permite um mecanismo de interação simples entre os usuários e o sistema, para que desta forma fique melhor compreendido sobre as atribuições de cada usuário.

A interface gráfica é um aplicativo Web que transforma as atividades de administração dos domínios e busca da informação em atividades mais simples. Ela abrange os usuários *ASIDs, UPGs, IPCs* e *UPPs.* O único que não dispõe de acesso a esta é o usuário *SID*, por este se tratar de um mecanismo de interação automático (usuário não-humano). Cabe a este estabelecer a conexão com o

sistema e consultar as funções que foram definidas para a sua interação e processar as respostas provenientes.

Dos quatro tipos de usuários que interagem com a interface, somente o *UPP* não necessita validar *login* e senha para ter acesso as funcionalidades do sistema. Como este usuário acessa o sistema somente para consultar e/ou sugerir um novo termo para o domínio, não há a necessidade de aplicar maiores políticas de restrições além das que já foram definidas (limite de acesso à funções e validação do conteúdo inserido).

O usuário *ASID* é um usuário único, ou seja, existe somente um nome de usuário e senha para todos os acessos. Este interage com todos os domínios, uma vez que as suas ações são sempre de apoio aos mecanismos de integração de dados.

Os usuários UPG e UPC são os usuários que administram um determinado domínio e o seu papel é determinado conforme o domínio que ele venha a gerenciar. Isto se deve ao fato de que um mesmo usuário pode estar associado a mais de um domínio e ter um papel diferente em cada um destes (mais a frente é apresentando um exemplo sobre esta possibilidade). Desta forma, toda vez que um usuário se autenticar no sistema e escolher um domínio para administrar, cabe a este domínio dizer qual o papel deste usuário e desta forma a interface se adaptará, para oferecer somente as funcionalidades disponíveis para este tipo de usuário.

6 CONCLUSÃO

O corrente trabalho apresenta uma ferramenta cujo propósito principal é o auxílio aos mecanismos de integração de dados. Entretanto, o processo de popular a base de dados é o que demonstrava ser mais dispendioso. Desta forma os usuários responsáveis por popular o BDTerm, são os participantes de um determinado domínio.

Assim além de suprir o suporte aos mecanismos de integração de dados, a ferramenta oferece suporte à troca de conhecimento entre membros de um campo do conhecimento e a comunidade em geral. A comunidade pode ajudar, cadastrando as informações que sentir a necessidade de existir dentro de um domínio e seus possíveis relacionamentos semânticos. O que garante a validade da informação é a atuação de especialistas de cada área da informação. Assim sendo, novas categorias de usuários foram incorporados ao sistema e sua aplicabilidade tornou-se mais ampla.

Para estender ainda mais os recursos disponíveis aos mecanismos de integração de dados, é permitido aos grupos de pesquisa sobre este assunto, cadastrarem “termos especiais” aos domínios. Esta informação tem o objetivo de suprir a base com grafias particulares encontradas nos dados a serem integrados. Desta forma, o conhecimento compartilhado belo BDTerm não é somente o que se encontra em uso corrente no domínio, agregando-se a este semânticas particulares encontradas em algumas fontes de dados.

Diante desta justificativa, este trabalho vem a contribuir com a disponibilização de um sistema que permite cadastro e consulta sobre termos e relacionamentos semânticos para o público em geral. Ele permite que mecanismos de integração de dados utilizem as bases terminológicas como suporte a suas tomadas de decisões, assim como permite que os centros de pesquisa sobre integração de dados troquem conhecimento, sem que estes dados venham a sobrecarregar os demais usuários com informações irrelevantes, ou seja, convenções semânticas específicas de certas fontes de dados.

Como trabalhos futuros, sugere-se o seguinte:

- Melhorar a interface com os usuários;
- Criar uma interface especial para os mecanismos de integração de dados como por exemplo chamadas a procedimentos remotos;
- Aumentar o número de relações semânticas que o sistema pode abrigar;
- Propor novas funcionalidades.

O primeiro protótipo do sistema de BDTerm está disponível no *site* do Grupo de Banco de Dados da UFSC (http://www.grupobd.inf.ufsc.br/) e sua utilização inicial, como salientado anteriormente, será como apoio ao processo de integração de esquemas XML realizado pelo BInXS. BInXS é o principal projeto de

pesquisa atualmente em desenvolvimento pelo Grupo de BD.

Referência

ABITEBOUL, Serge. Querying Semi-Structured Data. Disponível em: <http://heuser.inf.ufrgs.br/cmp170-02.2/1-1/Abiteboul.pdf>. Acesso em: 27 novembro 2004

ARTEMIS. Analysis and Reconciliation Tool Environment for Multiple Information System. Disponível em: <http://islab.dico.unimi.it/*ARTEMIS*/> Acesso em: 27 novembro 2004

BIDERMAN, M. T. C., AS CIÊNCIAS DO LÉXICO. in (Oliveira, A. M. P. P; Isquerdo, A. N. As ciências do Léxico: lexicologia, lexicografia, terminologia. 2 ed. Campo Grande, Editora UFMS, 2001)

BOTELHO, Fabiano Cupertino. IMPLEMENTAÇÃO DE UM EDITOR ESTRUTURAL PARA ONTOLOGIAS. 2002/2. 45f. Trabalho de conclusão de curso (Bacharelado em Informática) – Universidade Federal de Viçosa, Viçosa.

BRADLEY, Neil. TheXML Companion. 2 ed. Harlow, USA: Addison-Wesley Longman 2000. 435p.

CABRÉ, M. Teresa. La Terminologia: Teoria, metodologia, aplicaciones. 1. ed. Barcelona, Editorial Empúres, S. A., 1993.

CORAZZON, Raul. Ontology. A resource guide for philosophers. Disponível em <http://www.formalontology.it/>. Acesso em: 27novembro 2004

GRUBER, Tom. Virtual Documents. Disponível em: <http://ksl-web.stanford.edu/people/gruber/virtual-documents/index.html>. Acesso em: 27 novembro 2004

KOWALSKI, Gerald; MAYBURY, M. T. Information Storage and Retrieval Systems: Theory and Implematation. 2 ed. USA: Kluwer Academic Publishers, 2000.

KRIEGER, Maria da Graça; FINATTO, Maria José Bocorny. Introdução à Terminologia: teoria e prática. 1 ed. São Paulo, Editora Contexto, 2004.

MELLO, Ronaldo dos Santos. Uma Abordagem *Bottom-Up* para a Integração Semântica de Esquemas XML. 2002. 145f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. Database management systems. 3rd ed. Boston: McGraw-Hill, 2000.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de banco de dados.. São Paulo: MaKron Books do Brasil, 1999

SOWA, John F. Ontology. Disponível em: <http://www.jfsowa.com/ontology/index.htm>. Acessado em: 27 novembro 2004

TANENBAUM, Andrew S. Redes de Computadores. 4. ed Rio de Janeiro: Campus; 1999.

ANEXO B – ESTRUTURA E FUNÇÕES DO BANCO DE DADOS

```
--
-- Name: termoesemantica; Type: TYPE; Schema: public; Owner: adm
--

CREATE TYPE termoesemantica AS (
	id_termo integer,
	id_dominio integer,
	id_semantica integer,
	termo text,
	definicao text,
	relacao text,
	rstatus boolean,
	tstatus boolean
);


ALTER TYPE public.termoesemantica OWNER TO adm;

--
-- Name: termos; Type: TYPE; Schema: public; Owner: adm
--

CREATE TYPE termos AS (
	id_termo integer,
	nome text,
	definicao text,
	status boolean
);


ALTER TYPE public.termos OWNER TO adm;

--
-- Name: termossemantica; Type: TYPE; Schema: public; Owner: adm
--

CREATE TYPE termossemantica AS (
	id_termo integer,
	nome text,
	definicao text,
	id_semantica integer,
	relacao text,
	status boolean
);


ALTER TYPE public.termossemantica OWNER TO adm;

--
-- Name: termossemanticacompleto; Type: TYPE; Schema: public; Owner: adm
--
```

```
CREATE TYPE termossemanticacompleto AS (
	id_termo2 integer,
	termo text,
	nome text,
	id_semantica integer
);


ALTER TYPE public.termossemanticacompleto OWNER TO adm;

--
-- Name: usuarios; Type: TYPE; Schema: public; Owner: adm
--

CREATE TYPE usuarios AS (
	id_usuario integer,
	id_dominio integer,
	nome text,
	email text,
	tipo character(1)
);


ALTER TYPE public.usuarios OWNER TO adm;

--
-- Name: alteraratuacaousuario(integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION alteraratuacaousuario(integer, integer) RETURNS text
    AS $_$
BEGIN
    -- $1 = identificador do usuario
    -- $2 = identiticador do dominio

    if $1 isnull or $2 isnull then
        raise exception '';
    end if;

    update atuacao set tipo = 'G' where id_usuario = $1 and id_dominio = $2;
    return 'OK: O tipo de participação do usuário foi alterado';

    exception
        when FOREIGN_KEY_VIOLATION then
            return 'ERRO: O domínio e/ou usuário não existe(m)';

        when RAISE_EXCEPTION then
            return 'ERRO: Identificador do usuário e do domínio são obrigatórios';
END; $_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.alteraratuacaousuario(integer, integer) OWNER TO adm;

--
-- Name: alterardadosusuario(integer, text, text, text); Type: FUNCTION; Schema: public; Owner: adm
--
```

```
CREATE FUNCTION alterardadosusuario(integer, text, text, text) RETURNS text
    AS $_$
declare
    resp  text;
BEGIN
    -- $1 = identificador do usuario
    -- $2 = nome
    -- $3 = e-mail
    -- $4 = senha

    if $4 isnull then
        update usuario set nome = $2, email = $3 where identificador = $1;
        resp := 'OK: Os dados foram alterados e a sua senha não foi alterada';
    else
        update usuario set nome = $2, email = $3, senha = $4 where identificador = $1;
        resp := 'OK: Os dados foram alterados e a sua senha foi alterada';
    end if;

    if FOUND = false then
        resp := "ERRO: O usuário não existe no sistema";
    end if;
    return resp;

    exception
        when UNIQUE_VIOLATION then
            return 'ERRO: O e-mail informado já pertence a outro usuário';

END; $_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.alterardadosusuario(integer, text, text, text) OWNER TO adm;

--
-- Name: alterardefinicaotermo(integer, integer, text); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION alterardefinicaotermo(integer, integer, text) RETURNS text
    AS $_$
BEGIN
    -- $1 = identificador termo
    -- $2 = identificador dominio
    -- $3 = nova definicao

    if $1 isnull or $2 isnull then
        raise exception '';
    end if;

    update definicao set definicao = $3 where id_termo = $1 and id_dominio = $2;
    return 'OK: A definição do termo foi alterada';

    exception
        when FOREIGN_KEY_VIOLATION then
            return 'ERRO: A relação entre o termo e o domínio não existe';

        when RAISE_EXCEPTION then
```

```
        return 'ERRO: O identificador do termo e/ou domínio não foi informado';
END; $_$
   LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.alterardefinicaotermo(integer, integer, text) OWNER TO adm;

--
-- Name: cadastrardominio(text, text, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION cadastrardominio(text, text, integer) RETURNS text
   AS $_$
BEGIN
   -- $1 = nome
   -- $2 = descricao
   -- $3 = identificador superdominio

   if $i isnull then
      raise exception;
   end if;

   insert into dominio (id_superdominio,nome,descricao) values ($3,$1,$2);
   return 'OK: O Domínio foi inserido';

   exception
      when UNIQUE_VIOLATION then
         return 'ERRO: O domínio já está cadastrado';

      when FOREIGN_KEY_VIOLATION then
         return 'ERRO: O identificado do superdomínio está incorreto';

      when RAISE_EXCEPTION then
         return 'ERRO: O nome do domínio deve ser informado';

END; $_$
   LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.cadastrardominio(text, text, integer) OWNER TO adm;

--
-- Name: cadastrarrelacaosemantica(integer, integer, integer, integer, text); Type: FUNCTION; Schema: public;
Owner: adm
--

CREATE FUNCTION cadastrarrelacaosemantica(integer, integer, integer, integer, text) RETURNS text
   AS $_$
declare
   valor boolean;
   aux integer;
BEGIN
   -- $1 = identificador termo1
   -- $2 = identificador termo2
   -- $3 = identificador dominio
   -- $4 = identificador semantica
   -- $5 = usuario que chamou a funcao
```

```
    if $1 isnull or $2 isnull or $3 isnull or $4 isnull or $5 isnull then
        raise exception 'ERRO: todos os parâmetros são obrigatórios';
    end if;

    select into aux count(id_termo) from definicao where id_dominio = $3 and id_termo = $1 or id_termo = $2;
    if aux != 2 then
        raise exception 'ERRO: os dois termos devem pertencer ao mesmo domínio';
    end if;

    valor := false;
    if $5 = 'asid' then
        valor := true;
    end if;
    if($4 = 1) then
        insert into relacaosemantica (id_dominio,id_termo1,id_termo2,id_semantica,status) values
($3,$1,$2,$4,valor);
        insert into relacaosemantica (id_dominio,id_termo1,id_termo2,id_semantica,status) values
($3,$2,$1,$4,valor);
    else
        if($4 = 2) then
            insert into relacaosemantica (id_dominio,id_termo1,id_termo2,id_semantica,status) values
($3,$1,$2,2,valor);
            insert into relacaosemantica (id_dominio,id_termo1,id_termo2,id_semantica,status) values
($3,$2,$1,3,valor);
        else
            insert into relacaosemantica (id_dominio,id_termo1,id_termo2,id_semantica,status) values
($3,$1,$2,3,valor);
            insert into relacaosemantica (id_dominio,id_termo1,id_termo2,id_semantica,status) values
($3,$2,$1,2,valor);
        end if;
    end if;

    return 'OK: A relação semântica entre os dois termos foi cadastrada';
    exception
        when FOREIGN_KEY_VIOLATION then
            return 'ERRO: um dos parâmetros não existe';

        when UNIQUE_VIOLATION then
            return 'ERRO: esta relação semântica já foi cadastrada';

END; $_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.cadastrarrelacaosemantica(integer, integer, integer, integer, text) OWNER TO adm;

--
-- Name: cadastrarsemantica(text, text); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION cadastrarsemantica(text, text) RETURNS text
    AS $_$
begin
    -- $1 = conceito semantico
    -- $2 = definicao do conceito
```

```
   insert into semantica (nome,definicao) values ($1,$2);
   return 'OK: Conceito semântico inserido';
   exception
      when UNIQUE_VIOLATION then
         return 'ERRO: O conceito já está cadastrado';
end;$_$
   LANGUAGE plpgsql STRICT SECURITY DEFINER;


ALTER FUNCTION public.cadastrarsemantica(text, text) OWNER TO adm;

--
-- Name: cadastrartermo(text, text, integer, text); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION cadastrartermo(text, text, integer, text) RETURNS text
   AS $_$
Declare
   idt integer;
   aux text;
BEGIN
   -- $1 = termo
   -- $2 = definicao
   -- $3 = identificador dominio
   -- $4 = usuario que estah chamando a funcao

   if $1 isnull or $3 isnull then
      raise exception '';
   end if;

   select into idt identificador from termo where termo = $1;
   if idt isnull then
      insert into termo (termo) values ($1);
      select into idt identificador from termo where termo = $1;
   end if;
   if $4 = 'asid' then
      insert into definicao (id_dominio,id_termo,definicao,status,finalidade) values ($3,idt,$2,'true','SID');
   else
      insert into definicao (id_dominio,id_termo,definicao,finalidade) values ($3,idt,$2,'P');
   end if;
   return 'OK: A definição do termo foi cadastrada';

   exception
      when UNIQUE_VIOLATION then
         return 'ERRO: O termo já está definido no domínio';

      when FOREIGN_KEY_VIOLATION then
         return 'ERRO: O domínio especificado não existe';

      when RAISE_EXCEPTION then
         return 'ERRO: O termo e o identificador do domínio são obrigatórios';

      when CHECK_VIOLATION then
         return 'ERRO: O valor de atuação deve ser P ou SID';
END;$_$
   LANGUAGE plpgsql SECURITY DEFINER;
```

```
ALTER FUNCTION public.cadastrartermo(text, text, integer, text) OWNER TO adm;

--
-- Name: cadastrarusuario(text, text, text); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION cadastrarusuario(text, text, text) RETURNS text
    AS $_$
BEGIN
    -- $1 = nome
    -- $2 = e-mail
    -- $3 = senha

    if $2 isnull or $3 isnull then
        raise exception '';
    end if;
    insert into usuario (nome,email,senha) values ($1,$2,$3);
    return 'OK: O usuário foi inserido';

    exception
        when UNIQUE_VIOLATION then
            return 'ERRO: O usuário já faz parte do sistema';

        when RAISE_EXCEPTION then
            return 'ERRO: email e senha são informações obrigatórias';
END; $_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.cadastrarusuario(text, text, text) OWNER TO adm;

--
-- Name: cadastrarusuariodominio(integer, integer, character); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION cadastrarusuariodominio(integer, integer, character) RETURNS text
    AS $_$
BEGIN
    -- $1 = identificador do usuario
    -- $2 = identiticador do dominio
    -- $3 = atuacao

    if $1 isnull or $2 isnull or $3 isnull then
        raise exception '';
    end if;

    insert into atuacao (id_dominio,id_usuario,tipo) values ($2,$1,$3);
    return 'OK: O usuário foi cadastrado no domínio';

    exception
        when UNIQUE_VIOLATION then
            return 'ERRO: O usuário já definido dentro do domínio';

        when FOREIGN_KEY_VIOLATION then
            return 'ERRO: O domínio e/ou usuário não existe(m)';
```

```
        when CHECK_VIOLATION then
           return 'ERRO: O valor de atuação deve ser G ou C';

        when RAISE_EXCEPTION then
           return 'ERRO: Todos os parâmetros são obrigatórios';
END; $_$
   LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.cadastrarusuariodominio(integer, integer, character) OWNER TO adm;

--
-- Name: consultaexisterelacoasemantica(integer, integer, integer); Type: FUNCTION; Schema: public; Owner:
adm
--

CREATE FUNCTION consultaexisterelacoasemantica(integer, integer, integer) RETURNS text
   AS $_X$
   # $_[1] = identificador termo1
   # $_[2] = identificador termo2
   # $_[3] = identificador dominio

   if ($_[0] == null || $_[1] == null || $_[2] == null){
      elog(ERROR,'ERRO: Todos os parâmetros são obrigatórios');
   }
   @resp = ();
   sub busca{ # busca todos os sinonimos do termo1
      $sql = 'select id_termo2 from relacaosemantica where id_termo1 = '.$_[0].' and id_dominio = '.$_[1].' and
status = \'true\' and id_termo2 in (select id_termo from definicao d where d.id_dominio = '.$_[1].' and d.status =
\'true\')';
      my $rv = spi_exec_query($sql);
      my $nrows = $rv->{processed};
      foreach my $rn (0 .. $nrows - 1) {
         my $row = $rv->{rows}[$rn];
         my $a = $row->{id_termo2};
         my $tmp = grep {/^$a$/} @resp;
         my $tm = @resp;
          if($tmp == 0){
             push @resp,$a;
             busca($a,$_[1]);
          }
      }
   }
   push (@resp, $_[0]);
   busca($_[0],$_[2]);

   $encontra = grep {/^$_[1]$/} @resp;
   if($encontra == 0){

      $text = join ",",@resp;
      $sql = 'select s.nome from semantica s, relacaosemantica r where s.identificador = r.id_semantica and
r.id_termo1 in ('.$text.') and r.id_termo2 = '.$_[1].' and r.id_termo2 in (select d.id_termo from definicao d where
d.id_dominio = '.$_[2].' and d.status = \'true\')';
      $rv = spi_exec_query($sql);
      $nrows = $rv->{processed};
      if($nrows == 0){
         elog(ERROR,'ATENÇÃO: não existe informação relacionada aos parâmetros informados');
```

```
        }
        $row = $rv->{rows}[0];
        $text = $row->{nome};
    }
    else{
        $text = 'sinonímia';
    }
    return $text;
$_X$
    LANGUAGE plperl SECURITY DEFINER;


ALTER FUNCTION public.consultaexisterelacoasemantica(integer, integer, integer) OWNER TO adm;

--
-- Name: consultardefinicaotermo(text, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION consultardefinicaotermo(text, integer) RETURNS termos
    AS $_$

    -- $1 = termo
    -- $2 = identificador dominio

declare
    r termos%rowtype;
begin

    if $1 isnull or $2 isnull then
        raise exception 'ERRO: todos os parâmetros são obrigatórios';
    end if;

    for r in select t.identificador,t.termo,d.definicao,d.status from
    (termo t inner join definicao d on t.identificador = d.id_termo)
    where
    d.id_dominio = $2 and t.termo = $1 and d.status = 'true' loop
    return next r;
    end loop;

    if FOUND = false then
        raise exception 'ATENÇÂO: o domínio informado não existe e/ou nenhum termo está associado a este';
    end if;
return;
end;$_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.consultardefinicaotermo(text, integer) OWNER TO adm;

--
-- Name: consultarelacaosemanticacorreta(integer, integer, integer, integer); Type: FUNCTION; Schema: public;
Owner: adm
--

CREATE FUNCTION consultarelacaosemanticacorreta(integer, integer, integer, integer) RETURNS text
    AS $_X$
    # $_[0] = identificador termo1
```

```
    # $_[1] = identificador termo2
    # $_[2] = identificador dominio
    # $_[3] = identificador semantica

    if ($_[0] == null || $_[1] == null || $_[2] == null || $_[3] == null){
        elog(ERROR,'ERRO: Todos os parâmetros são obrigatórios');
    }
    @resp = ();
    sub busca{ # busca todos os sinonimos do termo1
        $sql = 'select id_termo2 from relacaosemantica where id_termo1 = '.$_[0].' and id_dominio = '.$_[1].' and
status = \'true\' and id_termo2 in (select id_termo from definicao d where d.id_dominio = '.$_[1].' and d.status =
\'true\') and id_semantica = 1';
        my $rv = spi_exec_query($sql);
        my $nrows = $rv->{processed};
        foreach my $rn (0 .. $nrows - 1) {
            my $row = $rv->{rows}[$rn];
            my $a = $row->{id_termo2};
            my $tmp = grep {/^$a$/} @resp;
            my $tm = @resp;
             if($tmp == 0){
                push @resp,$a;
                busca($a,$_[1]);
             }
        }
    }
    push (@resp, $_[0]);
    busca($_[0],$_[2]);

    $encontra = grep {/^$_[1]$/} @resp;
    $text = false;
    if($encontra == 0){

        $text2 = join ",",@resp;
        $sql2 = 'select s.nome from semantica s, relacaosemantica r where s.identificador = r.id_semantica and
r.id_termo1 in ('.$text2.') and r.id_termo2 = '.$_[1].' and id_semantica = '.$_[3].' and r.id_termo2 in (select
d.id_termo from definicao d where d.id_dominio = '.$_[2].' and d.status = \'true\')';
        $rv = spi_exec_query($sql2);
        $row = $rv->{processed};
        if($row > 0){
            $text = true;
        }

    }
    else{
        if($_[3] == 1){
            $text = true;
        }
    }

    return $text;
$_X$
    LANGUAGE plperl STRICT SECURITY DEFINER;


ALTER FUNCTION public.consultarelacaosemanticacorreta(integer, integer, integer, integer) OWNER TO adm;

--
```

```
-- Name: consultarelacaotermoavancado(integer, integer, integer); Type: FUNCTION; Schema: public; Owner:
adm
--

CREATE FUNCTION consultarelacaotermoavancado(integer, integer, integer) RETURNS SETOF termos
    AS $_X$
    # $_[0] = identificador termo1
    # $_[1] = identificador dominio
    # $_[2] = identificador semantica

    if ($_[0] == null || $_[1] == null || $_[2] == null){
        elog(ERROR,'ERRO: Todos os parâmetros são obrigatórios');
    }
    @resp = ();
    sub busca{ # busca todos os sinonimos do termo1
        $sql = 'select id_termo2 from relacaosemantica where id_termo1 = '.$_[0].' and id_dominio = '.$_[1].' and
status = \'true\' and id_termo2 in (select id_termo from definicao d where d.id_dominio = '.$_[1].' and d.status =
\'true\') and id_semantica = 1';
        my $rv = spi_exec_query($sql);
        my $nrows = $rv->{processed};
        foreach my $rn (0 .. $nrows - 1) {
          my $row = $rv->{rows}[$rn];
          my $a = $row->{id_termo2};
          my $tmp = grep {/^$a$/} @resp;
          my $tm = @resp;
           if($tmp == 0){
              push @resp,$a;
              busca($a,$_[1]);
           }
        }
    }
    push (@resp, $_[0]);
    busca($_[0],$_[1]);

    @resp2 = ();
    $text2 = join ",",@resp;

    $sql2 = 'select distinct t.identificador as id_termo, t.termo as nome,d.definicao, d.status from (termo t inner join
definicao d on t.identificador = d.id_termo) where d.id_termo in (select id_termo2 from relacaosemantica where
id_dominio = '.$_[1].' and id_termo1 in ('.$text2.') and status = \'true\' and id_semantica = '.$_[2].') and d.status =
\'true\' order by t.termo';

    $rv = spi_exec_query($sql2);
    $nrows = $rv->{processed};
    if($nrows == 0){
        elog(ERROR,'ATENÇÃO: não existe informação relacionada aos parâmetros informados');
    }
    foreach my $rn (0 .. $nrows - 1) {
        my $row = $rv->{rows}[$rn];
        push (@$resp2, $row);
    }
    return $resp2;
$_X$
    LANGUAGE plperl SECURITY DEFINER;


ALTER FUNCTION public.consultarelacaotermoavancado(integer, integer, integer) OWNER TO adm;
```

```
--
-- Name: consultarrelacaotermo(integer, integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION consultarrelacaotermo(integer, integer, integer) RETURNS SETOF termos
    AS $_$

    --$1 = identificador do termo
    --$2 = identificador do dominio
    --$3 = identificador da semantica

declare
    r termos%rowtype;
begin

    for r in select t.identificador,t.termo,d.definicao,d.status from
    ((relacaosemantica r inner join termo t on r.id_termo2 = t.identificador)
    inner join definicao d on d.id_termo = r.id_termo2)
    where
    r.id_termo1 = $1 and r.id_dominio = $2 and r.id_semantica = $3 and r.status = 'true' and d.finalidade = 'P'
    order by t.termo loop
    return next r;
    end loop;

    if FOUND = false then
        raise exception 'ATENÇÂO: nenhuma definição semântica está cadastrada';
    end if;
return;
end;$_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.consultarrelacaotermo(integer, integer, integer) OWNER TO adm;

--
-- Name: consultartodasrelacoes(integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION consultartodasrelacoes(integer, integer) RETURNS SETOF termossemanticacompleto
    AS $_X$
    # $_[0] = identificador termo1
    # $_[1] = identificador dominio


    if ($_[0] == null || $_[1] == null || $_[2] == null || $_[3] == null){
        elog(ERROR,'ERRO: Todos os parâmetros são obrigatórios');
    }
    @resp = ();
    sub busca{ # busca todos os sinonimos do termo1
        $sql = 'select id_termo2 from relacaosemantica where id_termo1 = '.$_[0].' and id_dominio = '.$_[1].' and
status = \'true\' and id_termo2 in (select id_termo from definicao d where d.id_dominio = '.$_[1].' and d.status =
\'true\') and id_semantica = 1';
        my $rv = spi_exec_query($sql);
        my $nrows = $rv->{processed};
        foreach my $rn (0 .. $nrows - 1) {
            my $row = $rv->{rows}[$rn];
```

```
          my $a = $row->{id_termo2};
          my $tmp = grep {/^$a$/} @resp;
          my $tm = @resp;
           if($tmp == 0){
              push @resp,$a;
              busca($a,$_[1]);
           }
        }
    }
    push (@resp, $_[0]);
    busca($_[0],$_[1]);

    @resp2 = ();
    $text2 = join ",",@resp;
    $sql2 = 'select distinct r.id_termo2, t.termo,s.nome,r.id_semantica from ((relacaosemantica r inner join
semantica s on r.id_semantica = s.identificador) inner join termo t on t.identificador = r.id_termo2) where
id_dominio = '.$_[1].' and id_termo1 in ('.$text2.') and r.status = \'true\' and id_termo2 in (select id_termo from
definicao d where id_dominio = '.$_[1].' and d.status = \'true\') order by r.id_semantica,r.id_termo2';
    $rv = spi_exec_query($sql2);
    $nrows = $rv->{processed};
    if($nrows == 0){
       elog(ERROR,'ATENÇÃO: não existe informação relacionada aos parâmetros informados');
    }
    foreach my $rn (0 .. $nrows - 1) {
       my $row = $rv->{rows}[$rn];

       push (@$resp2, $row);
    }
    return $resp2;
$_X$
    LANGUAGE plperl STRICT SECURITY DEFINER;


ALTER FUNCTION public.consultartodasrelacoes(integer, integer) OWNER TO adm;

SET default_tablespace = '';

SET default_with_oids = true;

--
-- Name: dominio; Type: TABLE; Schema: public; Owner: adm; Tablespace:
--

CREATE TABLE dominio (
    identificador integer DEFAULT nextval('inc_id_dominio'::text) NOT NULL,
    id_superdominio integer DEFAULT 0,
    nome text,
    descricao text
);


ALTER TABLE public.dominio OWNER TO adm;

--
-- Name: infodominio(integer); Type: FUNCTION; Schema: public; Owner: adm
--
```

```
CREATE FUNCTION infodominio(integer) RETURNS dominio
    AS $_$

    -- $1 = identificador do dominio

    select * from dominio where identificador = $1;
$_$
    LANGUAGE sql STRICT SECURITY DEFINER;


ALTER FUNCTION public.infodominio(integer) OWNER TO adm;

--
-- Name: logginusuario(text, text); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION logginusuario(text, text) RETURNS integer
    AS $_$

    -- $1 = email do usuario
    -- $2 = senha

    select identificador from usuario where email = $1 and senha = $2;
$_$
    LANGUAGE sql STRICT SECURITY DEFINER;


ALTER FUNCTION public.logginusuario(text, text) OWNER TO adm;

--
-- Name: removerdominio(integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION removerdominio(integer) RETURNS text
    AS $_$
BEGIN
    -- $1 = identificador dominio

    delete from dominio where identificador = $1;
    if FOUND = true then
        return 'OK: O domínio foi removido';
    else
        return 'ERRO: O domínio informado não existe';
    end if;

    exception
        when UNIQUE_VIOLATION then
            return 'ERRO: Um dos subdomínios já existe na base';

END; $_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.removerdominio(integer) OWNER TO adm;

--
-- Name: removerrelacaosemantica(integer, integer, integer); Type: FUNCTION; Schema: public; Owner: adm
```

```
--

CREATE FUNCTION removerrelacaosemantica(integer, integer, integer) RETURNS text
    AS $_$
BEGIN
    -- $1 = identificador termo1
    -- $2 = identificador termo2
    -- $3 = identificador dominio

    if $1 isnull or $2 isnull or $3 isnull then
        raise exception 'ERRO: um dos parâmetros foi omitido na chamada da função';
    end if;

    delete from relacaosemantica where id_dominio = $3 and id_termo1 = $1 and id_termo2 = $2;
    delete from relacaosemantica where id_dominio = $3 and id_termo1 = $2 and id_termo1 = $2;

    if FOUND = false then
        raise exception 'ERRO: o relacionametno informado não existe';
    end if;
    return 'OK: a relação semântica entre os dois termos foi cadastrada';
END; $_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.removerrelacaosemantica(integer, integer, integer) OWNER TO adm;

--
-- Name: removertermo(integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION removertermo(integer, integer) RETURNS text
    AS $_$
BEGIN
    -- $1 = identificador termo
    -- $2 = identificador dominio

    if $1 isnull or $2 isnull then
        raise exception '';
    end if;

    delete from definicao where id_termo = $1 and id_dominio = $2;
    return 'OK: A definição do termo foi removida do domínio';

    exception
        when FOREIGN_KEY_VIOLATION then
            return 'ERRO: A relação entre o termo e o domínio não existe';

        when RAISE_EXCEPTION then
            return 'ERRO: O identificador do termo e/ou domínio não foi informado';
END; $_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.removertermo(integer, integer) OWNER TO adm;

--
-- Name: usuario; Type: TABLE; Schema: public; Owner: adm; Tablespace:
```

```
--

CREATE TABLE usuario (
    identificador integer DEFAULT nextval('inc_id_usuario'::text) NOT NULL,
    nome text,
    email text,
    senha text
);


ALTER TABLE public.usuario OWNER TO adm;

--
-- Name: retornardadosusuario(integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornardadosusuario(integer) RETURNS usuario
    AS $_$

    -- $1 = identificador do usuário

    select * from usuario where identificador = $1;
$_$
    LANGUAGE sql STRICT SECURITY DEFINER;


ALTER FUNCTION public.retornardadosusuario(integer) OWNER TO adm;

--
-- Name: retornardominio(integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornardominio(integer) RETURNS SETOF dominio
    AS $_$

    -- $1 = identificador usurio

    select * from dominio where
    identificador in (select id_dominio from atuacao where id_usuario = $1 ) and identificador > 0 order by nome;
$_$
    LANGUAGE sql SECURITY DEFINER;


ALTER FUNCTION public.retornardominio(integer) OWNER TO adm;

--
-- Name: retornardominio(); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornardominio() RETURNS SETOF dominio
    AS $$
declare
r dominio%rowtype;
begin
    for r in select * from dominio where identificador > 0 order by nome loop
    return next r;
    end loop;
```

```
    if FOUND = false then
       raise exception 'Nenhum domínio foi encontrado no sistema';
    end if;
return;
end;$$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.retornardominio() OWNER TO adm;

--
-- Name: retornardominio2(); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornardominio2() RETURNS SETOF dominio
    AS $_X$
    # $_[0] = identificador termo1

     $sql = 'select * from dominio';
     $rv = spi_exec_query($sql);
     $nrows = $rv->{processed};
     foreach my $rn (0 .. $nrows - 1) {
        my $row = $rv->{rows}[$rn];
        push (@$resp2, $row);
     }
     return $resp2;

$_X$
    LANGUAGE plperl STRICT SECURITY DEFINER;


ALTER FUNCTION public.retornardominio2() OWNER TO adm;

--
-- Name: retornaridtermo(text); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornaridtermo(text) RETURNS integer
    AS $_$
    -- $1 = termo

    select identificador from termo where termo = $1;
$_$
    LANGUAGE sql STRICT SECURITY DEFINER;


ALTER FUNCTION public.retornaridtermo(text) OWNER TO adm;

--
-- Name: retornaridusuario(text); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornaridusuario(text) RETURNS usuario
    AS $_$

    -- $1 = email do usuário
```

```
    select * from usuario where email = $1;
$_$
    LANGUAGE sql STRICT SECURITY DEFINER;


ALTER FUNCTION public.retornaridusuario(text) OWNER TO adm;

--
-- Name: retornarrelacoessemanticasasid(integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornarrelacoessemanticasasid(integer, integer) RETURNS SETOF termoesemantica
    AS $_$

    -- $1 = identificador do termo
    -- $2 = identificador do dominio

declare
    r termoesemantica%rowtype;
begin

    if $1 isnull or $2 isnull then
        raise exception 'ERRO: todos os parâmetros são obrigatórios';
    end if;

    for r in select r.id_termo2,r.id_dominio,r.id_semantica,t.termo,d.definicao,s.nome,r.status,d.status from
        (((definicao d inner join termo t on d.id_termo = t.identificador)
        inner join relacaosemantica r on t.identificador = r.id_termo2)
        inner join semantica s on r.id_semantica = s.identificador)
        where
        r.id_dominio = $2 and r.id_termo1 = $1
        order by s.nome,t.termo loop
    return next r;
    end loop;

    if FOUND = false then
        raise exception 'ATENÇÂO: o domínio informado não existe e/ou nenhum termo está associado a este';
    end if;
return;
end;$_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.retornarrelacoessemanticasasid(integer, integer) OWNER TO adm;

--
-- Name: retornarrelacoessemanticaspublico(integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornarrelacoessemanticaspublico(integer, integer) RETURNS SETOF termoesemantica
    AS $_$

    -- $1 = identificador do termo
    -- $2 = identificador do dominio

declare
    r termoesemantica%rowtype;
```

```
begin

   if $1 isnull or $2 isnull then
      raise exception 'ERRO: todos os parâmetros são obrigatórios';
   end if;

   for r in select r.id_termo2,r.id_dominio,r.id_semantica,t.termo,d.definicao,s.nome,r.status,d.status from
      (((definicao d inner join termo t on d.id_termo = t.identificador)
      inner join relacaosemantica r on t.identificador = r.id_termo2)
      inner join semantica s on r.id_semantica = s.identificador)
      where
      r.id_dominio = $2 and r.id_termo1 = $1 and d.finalidade = 'P'
      order by s.nome,t.termo loop
   return next r;
   end loop;

   if FOUND = false then
      raise exception 'ATENÇÂO: o domínio informado não existe e/ou nenhum termo está associado a este';
   end if;
return;
end;$_$
   LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.retornarrelacoessemanticaspublico(integer, integer) OWNER TO adm;

--
-- Name: semantica; Type: TABLE; Schema: public; Owner: adm; Tablespace:
--

CREATE TABLE semantica (
   identificador integer DEFAULT nextval('inc_id_semantica'::text) NOT NULL,
   nome text,
   definicao text
);


ALTER TABLE public.semantica OWNER TO adm;

--
-- Name: retornarsemantica(); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornarsemantica() RETURNS SETOF semantica
   AS $$

declare
   r semantica%rowtype;
begin

   for r in select * from semantica order by nome loop
   return next r;
   end loop;

   if FOUND = false then
      raise exception 'ATENÇÂO: nenhuma definição semântica está cadastrada';
   end if;
```

```
return;
end;$$
   LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.retornarsemantica() OWNER TO adm;

--
-- Name: retornartermospublico(integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornartermospublico(integer) RETURNS SETOF termos
   AS $_$

   -- $1 = identificador dominio

declare
   r termos%rowtype;
begin

   if $1 isnull then
      raise exception 'ERRO: o parâmetro foi omitido';
   end if;

   for r in select t.identificador, t.termo, d.definicao, d.status from
   (termo t inner join definicao d on t.identificador = d.id_termo)
   where
   d.id_dominio = $1 and d.finalidade = 'P' order by t.termo loop
   return next r;
   end loop;

   if FOUND = false then
      raise exception 'ATENÇÂO: o domínio informado não existe e/ou nenhum termo está associado a este';
   end if;
return;
end;$_$
   LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.retornartermospublico(integer) OWNER TO adm;

--
-- Name: termo; Type: TABLE; Schema: public; Owner: adm; Tablespace:
--

CREATE TABLE termo (
   identificador integer DEFAULT nextval('inc_id_termo'::text) NOT NULL,
   termo text
);


ALTER TABLE public.termo OWNER TO adm;

--
-- Name: retornartermospublicocadastro(integer); Type: FUNCTION; Schema: public; Owner: adm
--
```

```
CREATE FUNCTION retornartermospublicocadastro(integer) RETURNS SETOF termo
    AS $_$

    -- $1 = identificador do dominio

    select * from termo t
    where
    t.identificador in (select id_termo from definicao where id_dominio = $1 and status = 'true' and finalidade = 'P')
order by termo;
$_$
    LANGUAGE sql STRICT SECURITY DEFINER;


ALTER FUNCTION public.retornartermospublicocadastro(integer) OWNER TO adm;

--
-- Name: retornartermossid(integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornartermossid(integer) RETURNS SETOF termos
    AS $_$

    -- $1 = identificador dominio
declare
    r termos%rowtype;
begin

    if $1 isnull then
        raise exception 'ERRO: o parâmetro foi omitido';
    end if;

    for r in select t.identificador, t.termo, d.definicao, d.status from
    (termo t inner join definicao d on t.identificador = d.id_termo)
    where
    d.id_dominio = $1 and d.finalidade = 'SID' order by t.termo loop
    return next r;
    end loop;

    if FOUND = false then
        raise exception 'ATENÇÂO: o domínio informado não existe e/ou nenhum termo está associado a este';
    end if;
return;
end;$_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.retornartermossid(integer) OWNER TO adm;

--
-- Name: retornarusuariodominio(integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornarusuariodominio(integer, integer) RETURNS usuarios
    AS $_$

    -- $1 = identificador do usuario
    -- $2 = identificador do dominio
```

```
    select u.identificador,a.id_dominio, u.nome, u.email, a.tipo from
    (usuario u inner join atuacao a on u.identificador = a.id_usuario)
    where
    a.id_dominio = $2 and a.id_usuario = $1;
$_$
    LANGUAGE sql STRICT SECURITY DEFINER;


ALTER FUNCTION public.retornarusuariodominio(integer, integer) OWNER TO adm;

--
-- Name: retornarusuariosdominio(integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION retornarusuariosdominio(integer) RETURNS SETOF usuarios
    AS $_$

    -- $1 = identificador dominio
declare
r usuarios%rowtype;
begin
    if $1 isnull then
        raise exception 'ERRO: o parâmetro foi omitido';
    end if;

    for r in select u.identificador,a.id_dominio, u.nome, u.email, a.tipo from
    (usuario u inner join atuacao a on u.identificador = a.id_usuario)
    where
    a.id_dominio = $1 order by u.nome loop
    return next r;
    end loop;
    if FOUND = false then
        raise exception 'ATENÇÂO: Não foram encontrados valores que satisfação a consulta';
    end if;
return;
end;$_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.retornarusuariosdominio(integer) OWNER TO adm;

--
-- Name: validarrelacaosemantica(integer, integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION validarrelacaosemantica(integer, integer, integer) RETURNS text
    AS $_$
declare
    resp text;
BEGIN
    -- $1 = identificador termo1
    -- $2 = identificador termo2
    -- $3 = identificador dominio
    -- $4 = novo status

    if $1 isnull or $2 isnull or $3 isnull then
```

```
        raise exception 'ERRO: um dos parâmetros foi omitido na chamada da função';
    end if;

    resp := 'OK: O status da relação semântica entre os dois termos foi alterada';
    update relacaosemantica set status = 'true' where id_dominio = $3 and id_termo1 = $1 and id_termo2 = $2;
    update relacaosemantica set status = 'true' where id_dominio = $3 and id_termo1 = $2 and id_termo2 = $1;
    if FOUND = false then
        raise exception 'ERRO: o relacionametno informado não existe';
    end if;
    return resp;

END; $_$
    LANGUAGE plpgsql SECURITY DEFINER;


ALTER FUNCTION public.validarrelacaosemantica(integer, integer, integer) OWNER TO adm;

--
-- Name: validartermo(integer, integer); Type: FUNCTION; Schema: public; Owner: adm
--

CREATE FUNCTION validartermo(integer, integer) RETURNS text
    AS $_$
declare
    resp text;
begin
    -- $1 = termo
    resp := 'OK: termo foi validado no domínio';
    update definicao set status = 'true' where id_termo = $1 and id_dominio = $2;
    if FOUND = false then
        resp := 'OK: termo foi validado no domínio';
    end if;
    return resp;
end$_$
    LANGUAGE plpgsql STRICT SECURITY DEFINER;


ALTER FUNCTION public.validartermo(integer, integer) OWNER TO adm;

--
-- Name: atuacao; Type: TABLE; Schema: public; Owner: adm; Tablespace:
--

CREATE TABLE atuacao (
    id_dominio integer NOT NULL,
    id_usuario integer NOT NULL,
    tipo character(1),
    CONSTRAINT atuacao_tipo_check CHECK (((tipo = 'G'::bpchar) OR (tipo = 'C'::bpchar)))
);


ALTER TABLE public.atuacao OWNER TO adm;

--
-- Name: definicao; Type: TABLE; Schema: public; Owner: adm; Tablespace:
--
```

```
CREATE TABLE definicao (
    id_dominio integer NOT NULL,
    id_termo integer NOT NULL,
    definicao text,
    status boolean DEFAULT false,
    finalidade character(3),
    CONSTRAINT definicao_finalidade_check CHECK (((finalidade = 'SID'::bpchar) OR (finalidade = 'P'::bpchar)))
);


ALTER TABLE public.definicao OWNER TO adm;

--
-- Name: inc_id_dominio; Type: SEQUENCE; Schema: public; Owner: adm
--

CREATE SEQUENCE inc_id_dominio
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.inc_id_dominio OWNER TO adm;

--
-- Name: inc_id_semantica; Type: SEQUENCE; Schema: public; Owner: adm
--

CREATE SEQUENCE inc_id_semantica
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.inc_id_semantica OWNER TO adm;

--
-- Name: inc_id_termo; Type: SEQUENCE; Schema: public; Owner: adm
--

CREATE SEQUENCE inc_id_termo
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.inc_id_termo OWNER TO adm;

--
-- Name: inc_id_usuario; Type: SEQUENCE; Schema: public; Owner: adm
--

CREATE SEQUENCE inc_id_usuario
    INCREMENT BY 1
```

```
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;


ALTER TABLE public.inc_id_usuario OWNER TO adm;

--
-- Name: relacaosemantica; Type: TABLE; Schema: public; Owner: adm; Tablespace:
--

CREATE TABLE relacaosemantica (
    id_dominio integer NOT NULL,
    id_termo1 integer NOT NULL,
    id_termo2 integer NOT NULL,
    id_semantica integer,
    status boolean DEFAULT false
);


ALTER TABLE public.relacaosemantica OWNER TO adm;

--
-- Name: atuacao_pkey; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY atuacao
    ADD CONSTRAINT atuacao_pkey PRIMARY KEY (id_dominio, id_usuario);


ALTER INDEX public.atuacao_pkey OWNER TO adm;

--
-- Name: definicao_pkey; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY definicao
    ADD CONSTRAINT definicao_pkey PRIMARY KEY (id_dominio, id_termo);


ALTER INDEX public.definicao_pkey OWNER TO adm;

--
-- Name: dominio_id_superdominio_key; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY dominio
    ADD CONSTRAINT dominio_id_superdominio_key UNIQUE (id_superdominio, nome);


ALTER INDEX public.dominio_id_superdominio_key OWNER TO adm;

--
-- Name: dominio_pkey; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY dominio
```

```
    ADD CONSTRAINT dominio_pkey PRIMARY KEY (identificador);


ALTER INDEX public.dominio_pkey OWNER TO adm;

--
-- Name: relacaosemantica_pkey; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY relacaosemantica
    ADD CONSTRAINT relacaosemantica_pkey PRIMARY KEY (id_dominio, id_termo1, id_termo2);


ALTER INDEX public.relacaosemantica_pkey OWNER TO adm;

--
-- Name: semantica_nome_key; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY semantica
    ADD CONSTRAINT semantica_nome_key UNIQUE (nome);


ALTER INDEX public.semantica_nome_key OWNER TO adm;

--
-- Name: semantica_pkey; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY semantica
    ADD CONSTRAINT semantica_pkey PRIMARY KEY (identificador);


ALTER INDEX public.semantica_pkey OWNER TO adm;

--
-- Name: termo_pkey; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY termo
    ADD CONSTRAINT termo_pkey PRIMARY KEY (identificador);


ALTER INDEX public.termo_pkey OWNER TO adm;

--
-- Name: usuario_email_key; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
--

ALTER TABLE ONLY usuario
    ADD CONSTRAINT usuario_email_key UNIQUE (email);


ALTER INDEX public.usuario_email_key OWNER TO adm;

--
-- Name: usuario_pkey; Type: CONSTRAINT; Schema: public; Owner: adm; Tablespace:
```

```
--

ALTER TABLE ONLY usuario
    ADD CONSTRAINT usuario_pkey PRIMARY KEY (identificador);


ALTER INDEX public.usuario_pkey OWNER TO adm;

--
-- Name: atuacao_id_dominio_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--

ALTER TABLE ONLY atuacao
    ADD CONSTRAINT atuacao_id_dominio_fkey FOREIGN KEY (id_dominio) REFERENCES
dominio(identificador) ON DELETE CASCADE;


--
-- Name: atuacao_id_usuario_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--

ALTER TABLE ONLY atuacao
    ADD CONSTRAINT atuacao_id_usuario_fkey FOREIGN KEY (id_usuario) REFERENCES
usuario(identificador) ON DELETE CASCADE;


--
-- Name: definicao_id_dominio_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--

ALTER TABLE ONLY definicao
    ADD CONSTRAINT definicao_id_dominio_fkey FOREIGN KEY (id_dominio) REFERENCES
dominio(identificador) ON DELETE CASCADE;


--
-- Name: definicao_id_termo_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--

ALTER TABLE ONLY definicao
    ADD CONSTRAINT definicao_id_termo_fkey FOREIGN KEY (id_termo) REFERENCES termo(identificador)
ON DELETE CASCADE;


--
-- Name: dominio_id_superdominio_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--

ALTER TABLE ONLY dominio
    ADD CONSTRAINT dominio_id_superdominio_fkey FOREIGN KEY (id_superdominio) REFERENCES
dominio(identificador) ON DELETE SET DEFAULT;


--
-- Name: relacaosemantica_id_dominio_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--
```

```
ALTER TABLE ONLY relacaosemantica
    ADD CONSTRAINT relacaosemantica_id_dominio_fkey FOREIGN KEY (id_dominio) REFERENCES
dominio(identificador) ON DELETE CASCADE;


--
-- Name: relacaosemantica_id_semantica_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--

ALTER TABLE ONLY relacaosemantica
    ADD CONSTRAINT relacaosemantica_id_semantica_fkey FOREIGN KEY (id_semantica) REFERENCES
semantica(identificador) ON DELETE CASCADE;


--
-- Name: relacaosemantica_id_termo1_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--

ALTER TABLE ONLY relacaosemantica
    ADD CONSTRAINT relacaosemantica_id_termo1_fkey FOREIGN KEY (id_termo1) REFERENCES
termo(identificador) ON DELETE CASCADE;


--
-- Name: relacaosemantica_id_termo2_fkey; Type: FK CONSTRAINT; Schema: public; Owner: adm
--

ALTER TABLE ONLY relacaosemantica
    ADD CONSTRAINT relacaosemantica_id_termo2_fkey FOREIGN KEY (id_termo2) REFERENCES
termo(identificador) ON DELETE CASCADE;


--
-- Name: public; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;
GRANT ALL ON SCHEMA public TO adm;
GRANT USAGE ON SCHEMA public TO publico;


--
-- Name: plperl_call_handler(); Type: ACL; Schema: public; Owner: postgres
--

REVOKE ALL ON FUNCTION plperl_call_handler() FROM PUBLIC;
REVOKE ALL ON FUNCTION plperl_call_handler() FROM postgres;
GRANT ALL ON FUNCTION plperl_call_handler() TO postgres;
GRANT ALL ON FUNCTION plperl_call_handler() TO PUBLIC;
GRANT ALL ON FUNCTION plperl_call_handler() TO adm;
GRANT ALL ON FUNCTION plperl_call_handler() TO asid;
GRANT ALL ON FUNCTION plperl_call_handler() TO mol;
GRANT ALL ON FUNCTION plperl_call_handler() TO participativo;
GRANT ALL ON FUNCTION plperl_call_handler() TO publico;
```

```
GRANT ALL ON FUNCTION plperl_call_handler() TO sid;


--
-- Name: plpgsql_call_handler(); Type: ACL; Schema: public; Owner: postgres
--

REVOKE ALL ON FUNCTION plpgsql_call_handler() FROM PUBLIC;
REVOKE ALL ON FUNCTION plpgsql_call_handler() FROM postgres;
GRANT ALL ON FUNCTION plpgsql_call_handler() TO postgres;
GRANT ALL ON FUNCTION plpgsql_call_handler() TO PUBLIC;
GRANT ALL ON FUNCTION plpgsql_call_handler() TO adm;
GRANT ALL ON FUNCTION plpgsql_call_handler() TO asid;
GRANT ALL ON FUNCTION plpgsql_call_handler() TO mol;
GRANT ALL ON FUNCTION plpgsql_call_handler() TO participativo;
GRANT ALL ON FUNCTION plpgsql_call_handler() TO publico;
GRANT ALL ON FUNCTION plpgsql_call_handler() TO sid;


--
-- Name: alteraratuacaousuario(integer, integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION alteraratuacaousuario(integer, integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION alteraratuacaousuario(integer, integer) FROM adm;
GRANT ALL ON FUNCTION alteraratuacaousuario(integer, integer) TO adm;
GRANT ALL ON FUNCTION alteraratuacaousuario(integer, integer) TO PUBLIC;
GRANT ALL ON FUNCTION alteraratuacaousuario(integer, integer) TO publico;


--
-- Name: alterardadosusuario(integer, text, text, text); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION alterardadosusuario(integer, text, text, text) FROM PUBLIC;
REVOKE ALL ON FUNCTION alterardadosusuario(integer, text, text, text) FROM adm;
GRANT ALL ON FUNCTION alterardadosusuario(integer, text, text, text) TO adm;
GRANT ALL ON FUNCTION alterardadosusuario(integer, text, text, text) TO PUBLIC;
GRANT ALL ON FUNCTION alterardadosusuario(integer, text, text, text) TO publico;


--
-- Name: cadastrartermo(text, text, integer, text); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION cadastrartermo(text, text, integer, text) FROM PUBLIC;
REVOKE ALL ON FUNCTION cadastrartermo(text, text, integer, text) FROM adm;
GRANT ALL ON FUNCTION cadastrartermo(text, text, integer, text) TO adm;
GRANT ALL ON FUNCTION cadastrartermo(text, text, integer, text) TO PUBLIC;
GRANT ALL ON FUNCTION cadastrartermo(text, text, integer, text) TO asid;
GRANT ALL ON FUNCTION cadastrartermo(text, text, integer, text) TO participativo;
GRANT ALL ON FUNCTION cadastrartermo(text, text, integer, text) TO publico;


--
-- Name: cadastrarusuario(text, text, text); Type: ACL; Schema: public; Owner: adm
--
```

```
REVOKE ALL ON FUNCTION cadastrarusuario(text, text, text) FROM PUBLIC;
REVOKE ALL ON FUNCTION cadastrarusuario(text, text, text) FROM adm;
GRANT ALL ON FUNCTION cadastrarusuario(text, text, text) TO adm;
GRANT ALL ON FUNCTION cadastrarusuario(text, text, text) TO PUBLIC;
GRANT ALL ON FUNCTION cadastrarusuario(text, text, text) TO publico;


--
-- Name: cadastrarusuariodominio(integer, integer, character); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION cadastrarusuariodominio(integer, integer, character) FROM PUBLIC;
REVOKE ALL ON FUNCTION cadastrarusuariodominio(integer, integer, character) FROM adm;
GRANT ALL ON FUNCTION cadastrarusuariodominio(integer, integer, character) TO adm;
GRANT ALL ON FUNCTION cadastrarusuariodominio(integer, integer, character) TO PUBLIC;
GRANT ALL ON FUNCTION cadastrarusuariodominio(integer, integer, character) TO publico;


--
-- Name: consultarelacaotermoavancado(integer, integer, integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION consultarelacaotermoavancado(integer, integer, integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION consultarelacaotermoavancado(integer, integer, integer) FROM adm;
GRANT ALL ON FUNCTION consultarelacaotermoavancado(integer, integer, integer) TO adm;
GRANT ALL ON FUNCTION consultarelacaotermoavancado(integer, integer, integer) TO PUBLIC;
GRANT ALL ON FUNCTION consultarelacaotermoavancado(integer, integer, integer) TO sid;


--
-- Name: infodominio(integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION infodominio(integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION infodominio(integer) FROM adm;
GRANT ALL ON FUNCTION infodominio(integer) TO adm;
GRANT ALL ON FUNCTION infodominio(integer) TO PUBLIC;
GRANT ALL ON FUNCTION infodominio(integer) TO asid;
GRANT ALL ON FUNCTION infodominio(integer) TO publico;
GRANT ALL ON FUNCTION infodominio(integer) TO sid;


--
-- Name: logginusuario(text, text); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION logginusuario(text, text) FROM PUBLIC;
REVOKE ALL ON FUNCTION logginusuario(text, text) FROM adm;
GRANT ALL ON FUNCTION logginusuario(text, text) TO adm;
GRANT ALL ON FUNCTION logginusuario(text, text) TO PUBLIC;
GRANT ALL ON FUNCTION logginusuario(text, text) TO publico;


--
-- Name: removerrelacaosemantica(integer, integer, integer); Type: ACL; Schema: public; Owner: adm
--
```

```
REVOKE ALL ON FUNCTION removerrelacaosemantica(integer, integer, integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION removerrelacaosemantica(integer, integer, integer) FROM adm;
GRANT ALL ON FUNCTION removerrelacaosemantica(integer, integer, integer) TO adm;
GRANT ALL ON FUNCTION removerrelacaosemantica(integer, integer, integer) TO PUBLIC;
GRANT ALL ON FUNCTION removerrelacaosemantica(integer, integer, integer) TO asid;
GRANT ALL ON FUNCTION removerrelacaosemantica(integer, integer, integer) TO publico;


--
-- Name: retornardadosusuario(integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornardadosusuario(integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornardadosusuario(integer) FROM adm;
GRANT ALL ON FUNCTION retornardadosusuario(integer) TO adm;
GRANT ALL ON FUNCTION retornardadosusuario(integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornardadosusuario(integer) TO publico;


--
-- Name: retornardominio(integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornardominio(integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornardominio(integer) FROM adm;
GRANT ALL ON FUNCTION retornardominio(integer) TO adm;
GRANT ALL ON FUNCTION retornardominio(integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornardominio(integer) TO publico;


--
-- Name: retornarrelacoessemanticasasid(integer, integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornarrelacoessemanticasasid(integer, integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornarrelacoessemanticasasid(integer, integer) FROM adm;
GRANT ALL ON FUNCTION retornarrelacoessemanticasasid(integer, integer) TO adm;
GRANT ALL ON FUNCTION retornarrelacoessemanticasasid(integer, integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornarrelacoessemanticasasid(integer, integer) TO asid;


--
-- Name: retornarrelacoessemanticaspublico(integer, integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornarrelacoessemanticaspublico(integer, integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornarrelacoessemanticaspublico(integer, integer) FROM adm;
GRANT ALL ON FUNCTION retornarrelacoessemanticaspublico(integer, integer) TO adm;
GRANT ALL ON FUNCTION retornarrelacoessemanticaspublico(integer, integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornarrelacoessemanticaspublico(integer, integer) TO publico;


--
-- Name: retornarsemantica(); Type: ACL; Schema: public; Owner: adm
--
```

```
REVOKE ALL ON FUNCTION retornarsemantica() FROM PUBLIC;
REVOKE ALL ON FUNCTION retornarsemantica() FROM adm;
GRANT ALL ON FUNCTION retornarsemantica() TO adm;
GRANT ALL ON FUNCTION retornarsemantica() TO PUBLIC;
GRANT ALL ON FUNCTION retornarsemantica() TO asid;
GRANT ALL ON FUNCTION retornarsemantica() TO participativo;
GRANT ALL ON FUNCTION retornarsemantica() TO publico;


--
-- Name: retornartermospublico(integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornartermospublico(integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornartermospublico(integer) FROM adm;
GRANT ALL ON FUNCTION retornartermospublico(integer) TO adm;
GRANT ALL ON FUNCTION retornartermospublico(integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornartermospublico(integer) TO publico;
GRANT ALL ON FUNCTION retornartermospublico(integer) TO asid;


--
-- Name: retornartermospublicocadastro(integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornartermospublicocadastro(integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornartermospublicocadastro(integer) FROM adm;
GRANT ALL ON FUNCTION retornartermospublicocadastro(integer) TO adm;
GRANT ALL ON FUNCTION retornartermospublicocadastro(integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornartermospublicocadastro(integer) TO asid;
GRANT ALL ON FUNCTION retornartermospublicocadastro(integer) TO participativo;
GRANT ALL ON FUNCTION retornartermospublicocadastro(integer) TO publico;


--
-- Name: retornartermossid(integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornartermossid(integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornartermossid(integer) FROM adm;
GRANT ALL ON FUNCTION retornartermossid(integer) TO adm;
GRANT ALL ON FUNCTION retornartermossid(integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornartermossid(integer) TO asid;


--
-- Name: retornarusuariodominio(integer, integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornarusuariodominio(integer, integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornarusuariodominio(integer, integer) FROM adm;
GRANT ALL ON FUNCTION retornarusuariodominio(integer, integer) TO adm;
GRANT ALL ON FUNCTION retornarusuariodominio(integer, integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornarusuariodominio(integer, integer) TO publico;


--
```

```
-- Name: retornarusuariosdominio(integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION retornarusuariosdominio(integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION retornarusuariosdominio(integer) FROM adm;
GRANT ALL ON FUNCTION retornarusuariosdominio(integer) TO adm;
GRANT ALL ON FUNCTION retornarusuariosdominio(integer) TO PUBLIC;
GRANT ALL ON FUNCTION retornarusuariosdominio(integer) TO publico;


--
-- Name: validarrelacaosemantica(integer, integer, integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION validarrelacaosemantica(integer, integer, integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION validarrelacaosemantica(integer, integer, integer) FROM adm;
GRANT ALL ON FUNCTION validarrelacaosemantica(integer, integer, integer) TO adm;
GRANT ALL ON FUNCTION validarrelacaosemantica(integer, integer, integer) TO PUBLIC;
GRANT ALL ON FUNCTION validarrelacaosemantica(integer, integer, integer) TO publico;


--
-- Name: validartermo(integer, integer); Type: ACL; Schema: public; Owner: adm
--

REVOKE ALL ON FUNCTION validartermo(integer, integer) FROM PUBLIC;
REVOKE ALL ON FUNCTION validartermo(integer, integer) FROM adm;
GRANT ALL ON FUNCTION validartermo(integer, integer) TO adm;
GRANT ALL ON FUNCTION validartermo(integer, integer) TO PUBLIC;
GRANT ALL ON FUNCTION validartermo(integer, integer) TO publico;
```

ANEXO C – INTERFACE

```
<?//gerencia a conexão  conexão.php
class conexao{

   var $con = false;
   var $usuario = null;
   var $senha = null;

   function conexao($usr = 0,$snh = 0){
      $this->usuario = $usr;
      $this->senha = $snh;
   }

   function getConexao(){
      if($this->usuario === "asid"){
         $par = "host='' port=5432 dbname=bdterm user=asid password=".$this->senha;
         $this->con = pg_connect($par);
         $_SESSION["usuario"]="asid";
      }
      else{
         if($this->usuario === "publico"){
            $par = "host='' port=5432 dbname=bdterm user=publico password=publico";
            $this->con = pg_connect($par);
            $_SESSION["usuario"]="publico";
         }
         else{
            $par = "host='' port=5432 dbname=bdterm user=participativo password=participativo";
            $this->con = pg_connect($par);
            $_SESSION["usuario"]="participativo";
         }
      }
      return $this->con;
   }
}
?>



<? // arquivo principal o index.php
include_once 'conexao.php';
include_once 'menuesquerdo.php';

session_start();

if(!(isset($_SESSION['conexao']) && isset($_SESSION['uid']))){
         $_SESSION['msg']="ERRO: Favor logar no sistema";
   header("Location: /bdterm/loggin.php?PHPSESSID=".session_id());
}

$conexao =  $_SESSION['conexao'];

if(isset($_POST['dominio'])){
   $_SESSION['iddom'] = $_POST['dominio'];
   $sql = "select * from infodominio(".$_SESSION['iddom'].")";
   $aux = pg_query($conexao->getConexao(),$sql);
```

```
    $aux = pg_fetch_object($aux);
    $_SESSION['nomedom'] = $aux->nome;
    $_SESSION['descdom'] = $aux->descricao;


    if($_SESSION['uid'] == -1){
        $_SESSION['tusu'] = "asid";
    }
    else{
        $sql = "select * from retornarusuariodominio(".$_SESSION['uid'].",".$_SESSION['iddom'].")";
        $aux = pg_query($conexao->getConexao(),$sql);
        $aux = pg_fetch_object($aux);
        $_SESSION['tusu'] = $aux->tipo;
    }
}

if(isset($_GET['opcao'])){
$_SESSION['acao'] = $_GET['opcao'].".php";
}
?>

<table border="1" cellpadding="0" cellspacing="0" style="border-collapse: collapse" bordercolor="#111111"
width="100%" id="AutoNumber1">
  <tr>
    <td width="100%" colspan="2">
    <form method="POST" action="index.php?PHPSESSID=<?=session_id()?>">
      <p align="center">
      <?
        if($_SESSION['uid'] === -1){
            $sql = "select * from retornardominio()";
        }
        else{
            $sql = "select * from retornardominio(".$_SESSION['uid'].")";
        }
        $aux = pg_query($conexao->getConexao(),$sql);
      ?>
      Escolha o domínio que deseja trabalhar: <select size="1" name="dominio">
      <option value = null selected> </option>
      <?
          while($aux2 = pg_fetch_object($aux)){
              ?><option value='<?echo $aux2->identificador?>'><?
              echo $aux2->nome;
              ?></option><?
          }

      ?>
      </select>
      <input type="submit" value="enviar" name="B1"></p>
      <a href='loggin.php?PHPSID=<?=session_id()?>'>sair do sistema</a>
    </form>
    </td>
  </tr>
  <?if(isset($_SESSION['iddom'])){?>
  <tr>
    <td width="100%" colspan="2"><p>Atualmente você está editando o domínio:
<?=$_SESSION['nomedom']?></p>
    <p>descrição: <?=$_SESSION['descdom']?></p>
```

```
   <p>sua participacao é de:
      <?
         switch($_SESSION['tusu']){
            case "asid":
               echo "Adiministrador de Sistemas de Integração de Dados";
            break;
            case "G":
               echo "Gerente do domínio";
            break;
            case "C":
               echo "Colaborador do domínio";
            break;
         }
      ?>
   </p>
   <br>
   </td>
  </tr>
  <tr>
   <td width="26%"><?= getmenu($_SESSION['tusu'])?></td>
   <td width="74%">
   <?
      if(isset($_SESSION['acao'])) include $_SESSION['acao'];
   ?>
   </td>
  </tr>
  <?}?>
</table>



<?//gerencia o login  loggin.php
include 'conexao.php';

session_start();
if($_POST['permitir']=="true"){
   if(isset($_POST['usu']) && strlen(trim($_POST['usu']))>0 && isset($_POST['senha']) &&
strlen(trim($_POST['senha']))>0){
            $aux = false;
            if($_POST['usu'] === "asid"){
                     $conexao = new conexao($_POST['usu'],$_POST['senha']);
         $con = $conexao->getConexao();
                     if($con !== false){
                              $uid = -1;
                              $aux = true;
                     }
            }
            else{
         $conexao = new conexao("publico",0);
         $con = $conexao->getConexao();
         if($con !== false){
            $sql = "select * from logginusuario('".trim($_POST['usu'])."','".md5(trim($_POST['senha']))."')";
                              $aux2 = pg_query($con,$sql);
                              $resultado = pg_fetch_array($aux2);
            if($resultado[0] !== null){
                                 $uid = $resultado[0];
```

```
                $aux = true;
            }
                    }
            }
        if($aux === false){
            $_SESSION['msg'] = "ERRO: Verifique se usuario e a senha estam corretos";
        }
            else{
                    $_SESSION['conexao'] = $conexao;
            $_SESSION['uid'] = $uid;
            header("Location: /bdterm/index.php?PHPSESSID=".session_id());
            exit;
        }
    }
    else{

        $_SESSION['msg'] = "ERRO: Informe o usuário e a senha";
    }
}
?>
<html>
<body>

<table border="1" width="100%"  height="92">
  <tr>
    <td width="100%" height="72">
    <p align="center">Informe o usuário e a senha para ter acesso as
    funcionalidades</td>
  </tr>
  <tr>
    <td width="100%" height="19"><font color="#FF0000">
                    <? if(isset($_SESSION['msg'])) echo $_SESSION['msg'];
          session_unset();
                    ?>
                    </font>
           </td>
  </tr>
  <tr>
    <td width="100%" height="19">
    <form method="POST" action="loggin.php<?echo"?PHPSESSID=".session_id();?>">
    <input type="hidden" name='permitir' value='true'>
      <!--webbot bot="SaveResults" U-File="fpweb:///_private/form_results.csv" S-Format="TEXT/CSV" S-Label-
Fields="TRUE" --><p>
      usuário: <input type="text" name="usu" ></p>
      <p>senha:  <input type="password" name="senha" ></p>
      <p><input type="submit" value="enviar" name="B1"></p>
    </form>
    <p> </td>
  </tr>
</table>
</body>
</html>



<? // index2.php para os upp
include_once 'conexao.php';
```

```
session_start();

if(!isset($_SESSION['usuario'])) $_SESSION['usuario'] = "participativo";
if(!isset($_SESSION['conexao'])){
   $conexao = new conexao($_SESSION['usuario'],0);
}
else{
   $conexao = $_SESSION['conexao'];
}

$sql = "select * from retornardominio()";
$aux = pg_query($conexao->getConexao(),$sql);
$aux3 = pg_query($conexao->getConexao(),$sql);

?>
<table border="0" width="100%" height="50%" cellpadding="2">

   <tr>
      <td width="50%">
         <form method="POST" action="index2.php?PHPSESSID=<?=session_id()?>">
         <input type="hidden" name='acao' value='exibetermo'>
         <p>Bem-vindo ao mecanismo de consulta do bdterm</p>
         <p>informe o termo que desejar procurar:<INPUT type="text" name="ter" value=""></p>
         <p>escolha um domínio:
         <select size="1" name="dominio">
            <option value = " " selected> </option>
            <?
            while($aux2 = pg_fetch_object($aux)){
            ?>
               <option value='<?echo $aux2->identificador?>'><? echo $aux2->nome;?></option>
            <?
            }
            ?>
         </select></p>
         <input type="submit" value="ok" name="B1"></p>
         </form>
      </td >

      <td width="50%" >

         Se o termo que está procurando não encontra-se registrado no domínio,
         você pode sugerí-lo.  Escolha o domínio e clique em prosseguir.
         <form name= "form2" method="POST" action="index2.php?PHPSESSID=<?=session_id()?>">
         <input type="hidden" name='acao' value='cadastrartermo'>
         <select size="1" name="dominio">
            <option value = " " selected> </option>
            <?
            while($aux2 = pg_fetch_object($aux3)){
            ?>
               <option value='<?echo $aux2->identificador?>'><? echo $aux2->nome;?></option>
            <?
            }
            ?>
         </select></p>
         <p><input type="submit" value="prosseguir" name="B1"></p>
```

```
        </form>
      </td>
   </tr>

   <tr>
      <td rowSpan="1" colspan="2">
      <?
      if(isset($_POST['acao'])){
         if(isset($_POST['dominio'])) $_SESSION['iddom'] = $_POST['dominio'];
         include $_POST['acao'].".php";
      }

?>
</td>
</tr>
</table>



<?  // menuesquerdo.php
session_start();
if(!(isset($_SESSION['conexao']) && isset($_SESSION['uid']))){
          $_SESSION['msg']="ERRO: Favor entrar no sistema";
          header("Location: /bdterm/loggin.php?PHPSESSID=".session_id());//rever aqui
}

function getMenu($tipousr){
   switch($tipousr){
            case "asid":
            ?>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=cadastrartermo">cadastrar
termo</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=removertermo">remover
termo</a></p>
        <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=alterartermo">alterar termo</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=crs">cadastrar relação
semântica</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=rrs">remover relação
semântica</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=listartermo">listar
termos</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=lrs">listar relações
semânticas</a><p>
            <?
            break;
            case "G":
      ?>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=cadastrarusuario">cadastrar
usuario</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=adu">alterar dados
usuário</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=aau">alterar atuação
usuário</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=rud">retornar usuários
domínio</a></p>
                     <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=cadastrartermo">cadastrar
```

```
termo</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=removertermo">remover
termo</a></p>
        <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=alterartermo">alterar termo</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=validartermo">validar
termo</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=crs">cadastrar relação
semântica</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=rrs">remover relação
semântica</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=vrs">validar relação
semântica</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=listartermo">listar
termos</a></p>
        <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=lrs">listar relações semânticas</a></p>
      <?
            break;
            case "C":
      ?>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=adu">alterar dados
usuário</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=rud">retornar usuários
domínio</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=cadastrartermo">cadastrar
termo</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=removertermo">remover
termo</a></p>
        <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=alterartermo">alterar termo</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=validartermo">validar
termo</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=crs">cadastrar relação
semântica</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=rrs">remover relação
semântica</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=vrs">validar relação
semântica</a></p>
                    <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=listartermo">listar
termos</a></p>
        <p><a href="index.php?PHPSESSID=<?=session_id()?>&opcao=lrs">listar relações semânticas</a></p>
      <?
            break;
   }
}
?>


<?// alterar atuacao usuario aau.php
$conexao = $_SESSION['conexao'];
if(isset($_POST['permitir'])){
   if(sizeof($_POST['validar']) > 0){
      $sql = "";
      $val = $_POST['validar'];
      foreach($val as $r){
         $sql = $sql."select alteraratuacaousuario(".$r.",".$_SESSION['iddom'].");";
      }
      $aux = pg_query($conexao->getConexao(),$sql);
      $aux = pg_fetch_array($aux);
```

```
        echo $aux[0];
    }
    else{
        echo "ERRO: Nenhum usuário foi escolhido";
    }
}
?>

<form method="POST" action="">
<p align='center'>Escolha o(s) usuário(s) que deve(m) ser promovido(s) a "Gerente(s)" do domínio</p>
<input type="hidden" name='permitir' value='true'>
<table border="1" height="100%" width= 100% >
    <tr>
        <td>Nome</td>
        <td>e-mail</td>
        <td>tipo</td>
    </tr>
        <?
            $sql = "select * from retornarusuariosdominio(".$_SESSION['iddom'].") where tipo = 'C'";
            $aux = pg_query($conexao->getConexao(),$sql);

            while($aux2 = pg_fetch_object($aux)){
        ?>
    <tr>
        <td>
            <input type="checkbox" name="validar[]" value=<?=$aux2->id_usuario?>> <?=$aux2->nome?>
        </td>
        <td >
            <?=$aux2->email?>
        </td>
        <td >
            <?
            echo "colaborador";
            ?>
        </td>
    </tr>
    <?
    }
    ?>
    <tr>
        <td>
            <input type="submit" value="enviar" name="B1">
        </td>
    </tr>
      <td></td>
      <td></td>
</table>
</form>



<?//alterar dados usuario adu.php
$conexao = $_SESSION['conexao'];

if(isset($_POST['permitir'])){
    if(strlen(trim($_POST['nome'])) > 0 && strlen(trim($_POST['email'])) > 0){
        $crp = 'null';
```

```
        if(strlen(trim($_POST['senha'])) > 0){
            if(trim($_POST['senha']) == trim($_POST['conf'])){
                $crp = md5(trim($_POST['senha']));
            }
            else{
                echo "ERRO: As senhas não conferem";
            }
        }

        if($crp !== 'null'){
            $sql = "select * from
alterardadosusuario(".$_SESSION['uid'].",'".trim($_POST['nome'])."','".trim($_POST['email'])."','".$crp."')";
        }
        else{
            $sql = "select * from
alterardadosusuario(".$_SESSION['uid'].",'".trim($_POST['nome'])."','".trim($_POST['email'])."',null)";
        }

        $aux = pg_query($conexao->getConexao(),$sql);
        $aux = pg_fetch_array($aux);
        echo $aux[0];
    }
    else{
        echo "ERRO: Pelo menos nome e e-mail devem ser informados";
    }
}

$sql = "select * from retornardadosusuario(".$_SESSION['uid'].")";
$aux = pg_query($conexao->getConexao(),$sql);
$aux = pg_fetch_object($aux);
?>
<form method="POST" action="">
<input type="hidden" name='permitir' value='true'>
<table border ='1'>
    <tr>
        <td>nome:<INPUT type="text" name="nome" value="<?=$aux->nome?>"></td>
        <td>email:<INPUT type="text" name="email" value="<?=$aux->email?>"></td>

        <td>senha:<INPUT type="password" name="senha"></td>
        <td>confirmar:senha<INPUT type="password" name="conf"></td>
    </tr>
    <tr>
        <td>
            <input type="submit" value="enviar" name="B1">
        </td>
        <td></td><td></td>
    </tr>
</table>
</form>



<?//cadastrartermo.php
session_start();

if(!isset($_SESSION['conexao'])){
    $_SESSION['usuario'] = "participativo";
```

```
    $conexao = new conexao($_SESSION['usuario'],0);
}
else{
    $conexao = $_SESSION['conexao'];
}

//if(isset($_POST['dominio']))

if(isset($_POST['permitir'])){
    if(strlen(trim($_POST['ter'])) > 0){
        $def = "null";
        $ter = trim($_POST['ter']);
        if(strlen(trim($_POST['def'])) > 0) $def = trim($_POST['def']);
        $sql = "select cadastrartermo('".$ter."','".$def."',".$_SESSION['iddom'].",current_user)";

        $aux = pg_query($conexao->getConexao(),$sql);
        $aux = pg_fetch_array($aux);
        $aux3 = explode(":",$aux[0]);
        echo $aux[0]."<br>";
        if($aux3[0] == "OK"){
            if(strlen(trim($_POST['semantica'])) > 0  && strlen(trim($_POST['termos'])) > 0){
                $sql = "select retornaridtermo('".$_POST['ter']."')";
                $aux2 = pg_query($conexao->getConexao(),$sql);
                $aux2 = pg_fetch_array($aux2);
                $sql = "select
cadastrarrelacaosemantica(".$aux2[0].",".$_POST['termos'].",".$_SESSION['iddom'].",".$_POST['semantica'].",cur
rent_user)";
                $aux2 = pg_query($conexao->getConexao(),$sql);
                $aux2 = pg_fetch_array($aux2);
                echo $aux2[0]."<br>";
            }
            else{
                if (strlen($_POST['semantica']) > 0  || strlen($_POST['termos']) > 0){
                    echo "ERRO: Um dos parâmetros para a relação semântica está faltando";
                }
            }
        }
    }
    else{
        echo "ERRO: O termo deve ser informado";
    }
}

if($_SESSION['usuario'] == "publico" || $_SESSION['usuario'] == "asid"){?>
    <FORM name = 'form' action = '<?echo "index.php?PHPSESSID=".session_id()."&opcao=cadastrartermo" ?>'
method="POST">
<?}
else{?>
    <FORM name = 'form' action = '<?echo "index2.php?PHPSESSID=".session_id() ?>' method="POST">
    <input type="hidden" name='acao' value='cadastrartermo'>
<?}
?>

    <input type="hidden" name='permitir' value='true'>

    <table border = 1 height="100%" width="100%">
        <tr align="center">
```

```
        <td>Informe o termo que deseja inserir<br></td>
      </tr>
      <tr>
        <td>
          Termo: <INPUT type="text" name='ter' size='30'><br>
          <br>
          definição:<p>
          <textarea name="def" cols="40" rows="5"></textarea></p><br>
        </td>
      </tr>
      <tr align="center">
        <td>Se desejar, pode relacionar semânticamente este termo a outro <br></td>
      </tr>
      <tr>
        <td>
          O termo a ser inserido é um:
          <?
            $sql = "select * from retornarsemantica()";
            $aux = pg_query($conexao->getConexao(),$sql);
          ?>
            <SELECT name="semantica">
              <option value = "></option>
            <?
            while($aux2 = pg_fetch_object($aux)){
            ?>
              <option value =<?=$aux2->identificador?>><?=$aux2->nome?></option>
            <?
            }
            ?>
            </SELECT>

          em relação:
          <?
            $sql = "select * from retornartermospublicocadastro(".$_SESSION['iddom'].")";
            $aux = pg_query($conexao->getConexao(),$sql);

          ?>
            <SELECT name="termos">
              <option value = "></option>
            <?
            while($aux2 = pg_fetch_object($aux)){
            ?>
              <option value =<?=$aux2->identificador?>><?=$aux2->termo?></option>
            <?
            }
            ?>
            </SELECT>
          <br>
        </td>
      </tr>
          <tr>
        <td>
          <INPUT type="submit" name="enviar" value="enviar">
        </td>
      </tr>
    </table>
</FORM>
```

```
<?//cadastrarusuario.php
include_once('Mail.php');
$conexao = $_SESSION['conexao'];

if(isset($_POST['permite'])){
   if(strlen(trim($_POST['nome'])) > 0 && strlen(trim($_POST['email'])) > 0 && strlen(trim($_POST['senha'])) > 0
&& strlen(trim($_POST['conf'])) > 0){
      if(trim($_POST['senha']) == trim($_POST['conf'])){
            $crp = md5(trim($_POST['senha']));
            $sql = "select * from cadastrarusuario('".trim($_POST['nome'])."','".trim($_POST['email'])."','".$crp."')";
            $aux = pg_query($conexao->getConexao(),$sql);
            $aux = pg_fetch_array($aux);
            $aux2 = explode(":",$aux[0]);
            if($aux2[0] == "OK"){
               echo $aux[0]."<br>";
               $sql = "select * from retornardadosusuario(".$_SESSION['uid'].")";
               $aux = pg_query($conexao->getConexao(),$sql);
               $aux = pg_fetch_object($aux);
               ///////envia email

               $recipients = trim($_POST['email']);
               $headers['From']    = 'xxxxxxxxxxx';
               $headers['To']      = trim($_POST['email']);
               $headers['Subject'] = 'Vc foi inserido no Sistema BDTerm';
               $headers['Date'] = date("r (T)");
               $body =
               'olá '.$_POST['nome'].' '.$sobrenome.'
               Você foi inserido por '.$aux->nome.'('.$aux->email.'),
               para participar do sistema BDTerm.
               Você pode acessar http://localhost/bdterm/loggin.php
               usuario = '.$_POST['email'].'
               senha = '.$_POST['senha'];
               $params['host'] = 'servidor de email';
               $mail_object =& Mail::factory('smtp',$params);
               $mail_object->send($recipients, $headers, $body);
               echo "O usuário receberá um mail com a confirmação do convite<br>";
               $sql = "select * from retornaridusuario('".trim($_POST['email'])."')";
               $aux = pg_query($conexao->getConexao(),$sql);
               $aux = pg_fetch_array($aux);
               $sql = "select * from cadastrarusuariodominio(".$aux[0].",".$_SESSION['iddom'].",'".$_POST['a']."')";
               $aux = pg_query($conexao->getConexao(),$sql);
               $aux = pg_fetch_array($aux);
               echo $aux[0]."<br>";
            }
            else{
               echo $aux[0];
            }
         }
         else{
            echo "ERRO: As senhas não conferem";
         }
   }
   else{
      echo "ERRO: O todos os parâmetros devem ser informados";
   }
```

```
}
?>
<form method="POST" action="">
<table border ='1'>
   <tr>
      <input type="hidden" name='permite' value='true'>
      <td>nome: <INPUT type="text" name="nome"></td>
      <td>email: <INPUT type="text" name="email" value=""></td>
      <td>senha: <INPUT type="password" name="senha"></td>
      <td>confirmar senha: <INPUT type="password" name="conf"></td>
   </tr>
   <tr>
      <td>
         <input type="submit" value="enviar" name="B1">
      </td>
      <td>
         Escolha a atuação deste usuário dentro do domínio:
         <select size="1" name="a">
            <option value='C' selected>Colaborador</option>
            <option value='G' >Gerente</option>
         </select>
      </td>
      <td>
      </td>
   </tr>
</table>
</form>



<?//cadastrar relação semântica  crs.php

$conexao = $_SESSION['conexao'];
if(isset($_POST['permitir'])){

   if(strlen($_POST['t1']) > 0 && strlen($_POST['s']) > 0 && strlen($_POST['t2']) > 0 ){

      $sql = "select
cadastrarrelacaosemantica(".$_POST['t1'].",".$_POST['t2'].",".$_SESSION['iddom'].",".$_POST['s'].",current_user
)";
      $aux = pg_query($conexao->getConexao(),$sql);
      $aux = pg_fetch_array($aux);
      echo $aux[0];
   }
   else{
      echo "ERRO: Parâmetros insuficientes para cadastrar uma relação semântica";
   }
}
?>
<form method="POST" action="">
<input type="hidden" name='permitir' value='true'>
<table border="1" height="100%" width= 100% >
<p align='center'>Escolha os termos e a relação semântica entre eles </p>
<table border="1" height="100%" width= 100% >

      <?
      if($conexao->usuario === "asid"){
```

```
        $sql = "select * from retornartermossid(".$_SESSION['iddom']."')";
    }
    else{
        $sql = "select * from retornartermospublico(".$_SESSION['iddom']."') where status = 'true'";
    }
    $sql2 = "select * from retornartermospublico(".$_SESSION['iddom']."') where status = 'true'";
    $aux = pg_query($conexao->getConexao(),$sql);
    $aux3 = pg_query($conexao->getConexao(),$sql2);
    $sql = "select * from retornarsemantica()";
    $aux2 = pg_query($conexao->getConexao(),$sql);


    ?>
<tr>
    <td>
        O termo: 
        <SELECT name="t1">
            <option value=""></option>
        <?
        while($aux4 = pg_fetch_object($aux)){
        ?>
            <option value='<?=$aux4->id_termo?>'><?=$aux4->nome?></option>
        <?
        }
        ?>
        </SELECT>
    </td>
    <td>
        é um: 

        <SELECT name="s">
            <option value=""></option>
        <?
        while($aux4 = pg_fetch_object($aux2)){
        ?>
            <option value='<?=$aux4->identificador?>'><?=$aux4->nome?></option>
        <?
        }
        ?>
        </SELECT>
    </td>
    <td>
        de: 
        <SELECT name="t2">
            <option value=""></option>
        <?
        while($aux4 = pg_fetch_object($aux3)){
        ?>
            <option value='<?=$aux4->id_termo?>'><?=$aux4->nome?></option>
        <?
        }
        ?>
        </SELECT>
    </td>
</tr>

<tr>
```

```
      <td>
         <input type="submit" value="enviar" name="B1">
      </td>
   </tr>
     <td></td>
     <td></td>
</table>
</form>


<?//exibe relacao semantica exiberelacao.php
   if(strlen($_POST['dominio']) > 0 && strlen($_POST['idter']) > 0 && strlen($_POST['idrel']) > 0){
      $sql = "select * from
consultarrelacaotermo('".trim($_POST['idter'])."',".$_POST['dominio'].",".$_POST['idrel'].")";
      $aux = pg_query($conexao->getConexao(),$sql);
      if (pg_num_rows($aux) > 0){
         echo "O(s) ".$_POST['nomerel']."(s) de ".$_POST['nometer']." é(são): ";
         while($aux2 = pg_fetch_object($aux)){

         ?>
            <form name="<?echo "a".$aux2->id_termo."a";?>"
action="index2.php?PHPSESSID=<?=session_id()?>" method="POST">
            <input type="hidden" name='dominio' value='<?=$_POST['dominio']?>'>
            <input type="hidden" name='ter' value='<?=$aux2->nome?>'>
            <input type="hidden" name='acao' value='exibetermo'>
            <input type="hidden" name='permitir' value='true'>
            <p>Termo: <a href="<?echo"javascript:document.a".$aux2->id_termo."a.submit()";?>"> <?=$aux2-
>nome?></a></p>
            </form>
         <?

         }
      }
      else{
         echo "Nada foi encontrado";
      }
   }
?>



<?// exibetermo.php
   if(strlen($_POST['dominio']) > 0 && strlen($_POST['ter']) > 0){
      $sql = "select * from consultardefinicaotermo('".trim($_POST['ter'])."',".$_POST['dominio'].")";
      $aux = pg_query($conexao->getConexao(),$sql);
      $aux2 = pg_fetch_object($aux);
      if(strlen($aux2->nome) > 0){
         $sql = "select * from retornarsemantica()";
         $aux = pg_query($conexao->getConexao(),$sql);
      ?>

        <table border="1" width="100%" height="100%">
           <tr>
              <td width="20%" height="10%">
              <?
              while($aux3 = pg_fetch_object($aux)) {
              ?>
```

```
                <p>
                <form method="POST"  name="<?echo "a".$aux3->identificador."a"?>"
action="index2.php?PHPSESSID=<?=session_id()?>">
                <input type="hidden" name='acao' value='exiberelacao'>
                <input type="hidden" name='permitir' value='true'>
                <input type="hidden" name='idter' value='<?=$aux2->id_termo?>'>
                <input type="hidden" name='nometer' value='<?=$aux2->nome?>'>
                <input type="hidden" name='dominio' value='<?=$_POST['dominio']?>'>
                <input type="hidden" name='idrel' value='<?=$aux3->identificador?>'>
                <input type="hidden" name='nomerel' value='<?=$aux3->nome?>'>
                <a href="<? echo "javascript:document.a".$aux3->identificador."a.submit()";?>"><?=$aux3-
>nome?></a>

                 </form>
                <?
                }
                ?>
                </td>

                <td>
                   <p>Termo: <?=$aux2->nome?></p>
                   <p>Definição:</p>
                   <p><?=trim($aux2->definicao)?></p>
                </td>
             </tr>
          </table>
       <?
       }
       else{
          echo "Nenhuma definição para este termo dentro do domínio foi encontrada";
       }
    }
    else{
       echo "ERRO: Parâmetros insuficientes para realizar a busca";
    }
    ?>


<?// listartermo.php
$conexao = $_SESSION['conexao'];

if($conexao->usuario === "asid"){
    $sql = "select * from retornartermossid(".$_SESSION['iddom'].") where status = 'true'";
    $aux = pg_query($conexao->getConexao(),$sql);
?>
    Lista de termos que pertencem ao domínio (validados) com finalidade de integração de dados
    <table border = '1' height="100%" width= 100% >
       <tr>
          <td>Termo</td>
          <td>Descrição</td>
       </tr>
       <?
       while($aux2 = pg_fetch_object($aux)){
       ?>
       <tr>
          <td><?=$aux2->nome?></td>
          <td><?=$aux2->definicao?></td>
```

```
        </tr>
        <?
        }
        ?>
    </table>
<?
}
?>
<br>
<br>

Lista de termos que pertencem ao domínio (validados) com finalidade de informação
    <table border = '1' height="100%" width= 100% >
        <tr>
            <td>Termo</td>
            <td>Descrição</td>
        </tr>
        <?
        $sql = "select * from retornartermospublico(".$_SESSION['iddom'].") where status = 'true' ";
        $aux = pg_query($conexao->getConexao(),$sql);

        while($aux2 = pg_fetch_object($aux)){
        ?>
        <tr>
            <td><?=$aux2->nome?></td>
            <td><?=$aux2->definicao?></td>
        </tr>
        <?
        }
    ?>
    </table>



<?// listar relação semântica lrs.php

$conexao = $_SESSION['conexao'];

    $sql = "select * from retornartermospublico(".$_SESSION['iddom'].") where status = 'true'";
    $aux = pg_query($conexao->getConexao(),$sql);
 ?>
<form method="POST" action="">
<p>Escolha um termo:
            <SELECT name='tn1'>
                <option value=''></option>
            <?
            while($aux4 = pg_fetch_object($aux)){
            ?>
                <option value='<? echo $aux4->id_termo.",".$aux4->nome?>'><?=$aux4->nome?></option>
            <?
            }
            ?>
            </SELECT>
<input type="submit" value="ok" name="B1"></p>
</form>
     </p>
```

```
<?
     if(strlen($_POST['tn1']) > 0){
       $aux = explode(",",$_POST['tn1']);
       $t1 = $aux[0];
       $n = $aux[1];

       $sql = "select * from retornarrelacoessemanticaspublico(".$t1.",".$_SESSION['iddom'].") where rstatus =
'true'";
       $aux = pg_query($conexao->getConexao(),$sql);
       if( pg_num_rows($aux) == 0){
          echo "<p>ERRO: Nenhuma relação para este termo foi encontrada</p>";
       }
       else{

?>
<table border="1" height="100%" width= 100% >


   <tr>
      <td colspan="5">O termo <b><?=$n?></b>, possuí as seguintes relações semânticas</td>
   </tr>
<input type="hidden" name="t1" value="<?=$t1?>" >
<input type="hidden" name="n" value="<?=$n?>" >
   <tr>
      <td>Termo</td>
      <td>Definição</td>
      <td>é um</td>
   </tr>
      <?
         while($aux2 = pg_fetch_object($aux)){
            $id = $t1.",".$aux2->id_termo.",".$_SESSION['iddom'];
      ?>
   <tr>
      <td>
         <?=$aux2->termo?>
      </td>
      <td >
         <?=$aux2->definicao?>
      </td>
      <td >
         <?=$aux2->relacao?>
      </td>

   </tr>
   <?
         }
         ?>
   </table>
   <?

   }
}
?>



<?//remover termo.php
```

```
$conexao = $_SESSION['conexao'];

if(isset($_POST['permitir'])){
   if(sizeof($_POST['remover'])){
            $sql = "";
            $rem = $_POST['remover'];

            foreach($rem as $r){
                     $sql = $sql."select removertermo(".$r.",".$_SESSION['iddom'].");";
            }
            $aux = pg_query($conexao->getConexao(),$sql);
            $aux = pg_fetch_array($aux);
            echo $aux[0]."<br>";
   }
   else{
            echo "ERRO: Nenhum termo foi selecionado para ser removido";
   }
}

?>

<form method="POST" action="">
<input type="hidden" name='permitir' value='true'>
<p align='center'>Escolha o(s) termo(s) que deseja remover</p>
<table border="1" height="100%" width= 100% >
   <tr>
                     <td>Termo</td>
                     <td>Definição</td>
                     <td>Status</td>
          </tr>
                     <?
      if($conexao->usuario === "asid"){
          $sql = "select * from retornartermossid(".$_SESSION['iddom'].")";
                     }
                     else{
                              $sql = "select * from retornartermospublico(".$_SESSION['iddom'].")";
                     }
                     $aux = pg_query($conexao->getConexao(),$sql);
                     while($aux2 = pg_fetch_object($aux)){
                     ?>
          <tr>
             <td>
                              <input type="checkbox" name="remover[]" value=<?=$aux2-
>id_termo?>> <?=$aux2->nome?>
                     </td>
                     <td >
                              <?=$aux2->definicao?>
                     </td>
                     <td >
                              <?
          if($aux2->status == "f"){
               echo "Falso";
            }
            else{
               echo "Verdadeiro";
            }
          ?>
```

```
                </td>
        </tr>
        <?
        }
        ?>
        <tr>
                <td>
                        <input type="submit" value="enviar" name="B1">
                </td>
        </tr>
          <td></td>
          <td></td>
</table>
</form>


<?// remover relação semântica  rrs.php

$conexao = $_SESSION['conexao'];
if($_POST['permitir']=="true" && strlen($_POST['t1']) > 0){

   if(sizeof($_POST['remover'])){
      $sql = "";
      $rem = $_POST['remover'];
      foreach($rem as $r){
         $sql = $sql."select removerrelacaosemantica(".$r.");";
      }
      $aux = pg_query($conexao->getConexao(),$sql);
      $aux = pg_fetch_array($aux);
      echo $aux[0];
   }
   else{
      echo "ERRO: Nenhuma relação semântica foi selecionada para ser removida";
   }
}
?>
<?
   if($conexao->usuario === "asid"){
      $sql = "select * from retornartermossid(".$_SESSION['iddom'].")";
   }
   else{
      $sql = "select * from retornartermospublico(".$_SESSION['iddom'].")";
   }
   $aux = pg_query($conexao->getConexao(),$sql);
 ?>
<form method="POST" action="">

   <p>Escolha um termo</p>
   <SELECT name='tn1'>
      <option value=''></option>
      <?
      while($aux4 = pg_fetch_object($aux)){
      ?>
      <option value='<? echo $aux4->id_termo.",".$aux4->nome?>'><?=$aux4->nome?></option>
      <?
      }
      ?>
```

```
</SELECT>
<?
if(strlen($_POST['tn1']) > 0){
   $aux = explode(",",$_POST['tn1']);
   $t1 = $aux[0];
   $n = $aux[1];
   ?><input type="hidden" name='permitir' value='true'><?    }
else{
   if(strlen($_POST['t1']) > 0){
      $t1 = $_POST['t1'];
      $n = $_POST['n'];
   }
}
if(sizeof($t1) > 0){
   if($conexao->usuario === "asid"){
      $sql = "select * from retornarrelacoessemanticasasid(".$t1.",".$_SESSION['iddom'].")";
   }
   else{
      $sql = "select * from retornarrelacoessemanticaspublico(".$t1.",".$_SESSION['iddom'].")";
   }
   $aux = pg_query($conexao->getConexao(),$sql);
   if( pg_num_rows($aux) == 0){

      echo "<p>ERRO: Nenhuma relação para este termo foi encontrada</p>";
   }
   else{

   ?>
   <p align='center'>Escolha a relação semântica que deseja remover:</p>
   <input type="hidden" name="t1" value="<?=$t1?>" >
   <input type="hidden" name="n" value="<?=$n?>" >
   <table border="1" height="100%" width= 100% >
      <tr>
         <td colspan="5">O termo <b><?=$n?></b>, possuí as seguintes relações semânticas</td>
      </tr>
      <tr>
         <td>Termo</td>
         <td>Definição</td>
         <td>Status do termo</td>
         <td>é um</td>
         <td>Status da relação</td>
      </tr>
      <?
      while($aux2 = pg_fetch_object($aux)){
         $id = $t1.",".$aux2->id_termo.",".$_SESSION['iddom'];
      ?>
      <tr>
         <td>
            <input type="checkbox" name="remover[]" value=<?=$id?>> <?=$aux2->termo?>
         </td>
         <td >
            <?=$aux2->definicao?>
         </td>
         <td>
         <?
            if($aux2->tstatus == "f"){
               echo "Falso";
```

```
                }
                else{
                    echo "Verdadeiro";
                }
            ?>
            </td>
            <td >
                <?=$aux2->relacao?>
            </td>
            <td>
            <?
                if($aux2->rstatus == "f"){
                    echo "Falso";
                }
                else{
                    echo "Verdadeiro";
                }
            ?>
            </td>
        </tr>
        <?
        }
        ?>
    </table>
<?
    }
}
?>
<p>
<input type="submit" value="ok" name="B1"></p>
</form>


<?// retorna usuarios do dominio rud.php
$conexao = $_SESSION['conexao'];

    $sql = "select * from retornarusuariosdominio(".$_SESSION['iddom'].")";
    $aux = pg_query($conexao->getConexao(),$sql);
?>
Usuários que fazem parte do domínio e suas atuações
<table border = '1' height="100%" width= 100% >
    <tr>
        <td>Nome</td>
        <td>e-mail</td>
        <td>atuação</td>
    </tr>
    <?
        while($aux2 = pg_fetch_object($aux)){
    ?>
        <tr>
            <td><?=$aux2->nome?></td>
            <td><?=$aux2->email?></td>
            <td>
            <?
                if($aux2->tipo == 'G'){
                    echo "Gerenciador";
                }
```

```
            else{
                echo "Colaborador";
            }

          ?>
          </td>
      </tr>
      <?
      }
      ?>
</table>


<?validartermo.php
$conexao = $_SESSION['conexao'];
if(isset($_POST['permite'])){

   if(sizeof($_POST['validar'])){
      $sql = "";
      $val = $_POST['validar'];

      foreach($val as $r){
         $sql = $sql."select validartermo(".$r.",".$_SESSION['iddom'].");";
      }
      $aux = pg_query($conexao->getConexao(),$sql);
      $aux = pg_fetch_array($aux);
      echo $aux[0];
   }
   else{
      echo "ERRO: Nenhum termo foi selecionado para ser validado";
   }
}
?>

<form method="POST" action="">
<input type="hidden" name='permite' value='true'>
<p align='center'>Escolha o(s) termo(s) que deseja validar no domínio</p>
<table border="1" height="100%" width= 100% >
   <tr>
      <td>Termo</td>
      <td>Definição</td>
      <td>Status</td>
   </tr>
      <?
         $sql = "select * from retornartermospublico(".$_SESSION['iddom'].") where status = 'false'";
         $aux = pg_query($conexao->getConexao(),$sql);
         while($aux2 = pg_fetch_object($aux)){
      ?>
   <tr>
      <td>
         <input type="checkbox" name="validar[]" value=<?=$aux2->id_termo?>> <?=$aux2->nome?>
      </td>
      <td >
         <?=$aux2->definicao?>
      </td>
      <td >
         <?
```

```
            if($aux2->status == "f"){
                echo "Falso";
              }
              else{
                echo "Verdadeiro";
              }
            ?>
          </td>
      </tr>
      <?
      }
      ?>
      <tr>
          <td>
            <input type="submit" value="enviar" name="B1">
          </td>
      </tr>
        <td></td>
        <td></td>
</table>
</form>


<?// validar relação semantica vrs.php

$conexao = $_SESSION['conexao'];
if($_POST['permitir']=="true" && strlen($_POST['t1']) > 0){
    if(sizeof($_POST['validar'])){
        $sql = "";
        $rem = $_POST['validar'];
        foreach($rem as $r){
            $sql = $sql."select validarrelacaosemantica(".$r.");";
        }
        $aux = pg_query($conexao->getConexao(),$sql);
        $aux = pg_fetch_array($aux);
        echo $aux[0];
    }
    else{
        echo "ERRO: Nenhum relação semântica foi selecionada para ser validada";
    }
}

?>
<?
    $sql = "select * from retornartermospublico(".$_SESSION['iddom'].") where status = 'true'";

    $aux = pg_query($conexao->getConexao(),$sql);
 ?>
<form method="POST" action="">
<p>Escolha um termo</p><br>
            <SELECT name='tn1'>
              <option value=''></option>
            <?
            while($aux4 = pg_fetch_object($aux)){
            ?>
              <option value='<? echo $aux4->id_termo.",".$aux4->nome?>'><?=$aux4->nome?></option>
            <?
```

```
            }
            ?>
            </SELECT>
    <?
    if(strlen($_POST['tn1']) > 0){
        $aux = explode(",",$_POST['tn1']);
        $t1 = $aux[0];
        $n = $aux[1];
        ?><input type="hidden" name='permitir' value='true'><?    }
    else{
        if(strlen($_POST['t1']) > 0){
            $t1 = $_POST['t1'];
            $n = $_POST['n'];
        }
    }
    if(sizeof($t1) > 0){
        $sql = "select * from retornarrelacoessemanticaspublico(".$t1.",".$_SESSION['iddom'].") where rstatus =
'false'";
        $aux = pg_query($conexao->getConexao(),$sql);
        if( pg_num_rows($aux) == 0){

            echo "<p>ERRO: Nenhuma relação para este termo foi encontrada</p>";
        }
        else{
?>
<p align='center'>Escolha a relação semântica que deseja validar:</p><br>
<table border="1" height="100%" width= 100% >


    <tr>
        <td colspan="5">O termo <b><?=$n?></b>, possuí as seguintes relações semânticas</td>
    </tr>
<input type="hidden" name="t1" value="<?=$t1?>" >
<input type="hidden" name="n" value="<?=$n?>" >
    <tr>
        <td>É um</td>
        <td>Termo</td>
        <td>Definição</td>
        <td>Status do termo</td>
    </tr>
        <?
         while($aux2 = pg_fetch_object($aux)){
            $id = $t1.",".$aux2->id_termo.",".$_SESSION['iddom'];
        ?>
    <tr>
        <td>
            <input type="checkbox" name="validar[]" value=<?=$id?>> <?=$aux2->relacao?>
        </td>
        <td >
            <?=$aux2->termo?>
        </td>
        <td >
            <?=$aux2->definicao?>
        </td>
        <td>
            <?
            if($aux2->tstatus == "f"){
```

```
                echo "Falso";
            }
            else{
                echo "Verdadeiro";
            }
          ?>
        </td>
      </tr>
        </tr>
          <?
          }
          ?>
    </table>
   <?
      }
   }
?>
<input type="submit" value="ok" name="B1">
</form>
```