UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

André Guerreiro Cantarelli

D2R EDITOR: HABILITANDO A PUBLICAÇÃO AUTOMÁTICA DE ANOTAÇÕES SEMÂNTICAS DE SITES DINÂMICOS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Rogério Cid Bastos, Dr.
Professor Orientador

Florianópolis, Fevereiro de 2005

D2R Editor: Habilitando a Publicação Automática de Anotações Semânticas de Sites Dinâmicos

André Guerreiro Cantarelli

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Área de Concentração em Sistemas de Computação e aprovada em sua forma final pelo programa de Pós-Graduação em Ciência da Computação.

_____
Prof. Raul Sidnei Wazlawick, Dr.
Coordenador do programa de pós-graduação
em Ciência da Computação.

Banca Examinadora:

_____
Prof. Rogério Cid Bastos, Dr.
Orientador

_____
Prof. Raul Sidnei Wazlawick, Dr.

_____
Prof. Fernando Ostuni Gauthier, Dr.

_____
Profa. Anita Maria da Rocha Fernandes, Dra.

Sumário

Lista de Figuras

Lista de Quadros

Resumo

O objetivo da *Web* Semântica é fazer com que sistemas entendam as informações publicadas na *Internet* para oferecer aos usuários um melhor aproveitamento no uso do computador. Para expressar uma semântica formal e viabilizar esta compreensão automatizada, os conteúdos devem estar descritos com a linguagem RDF (*Resource Description Framework*), tendo seus significados definidos e interligados através de relações ontológicas. Neste trabalho é desenvolvida uma ferramenta que, através de uma interface *Web*, permite o registro de informações sobre a estrutura de um banco de dados relacional e, com isto, gera um mapa de migração que facilita a geração automática de anotações semânticas sobre os conteúdos deste banco de dados. Adicionalmente, o sistema constrói um vocabulário RDF, que formaliza e compartilha a semântica dos termos utilizados nas anotações que serão geradas através do mapa. Com esta aplicação, diferentes sites dinâmicos se tornam aptos a publicar informações que possam ser igualmente compreendidas e relacionadas por agentes inteligentes.

Palavras-chave: Web Semântica; RDF; XML; Bancos de Dados Relacionais; Integração de Dados.

1 INTRODUÇÃO

A *Web* Semântica tem por objetivo fazer com que a *Internet* funcione como uma grande base de conhecimento que possa ser compreendida pelo computador, o qual estará apto a realizar tarefas mais proveitosas para os usuários. Neste novo ambiente, as aplicações serão capazes de trocar informações entre si, fazendo análises mais eficazes e chegando a conclusões que irão contribuir diretamente com o rendimento das pessoas que utilizam o computador.

Para que este processamento avançado seja viável, é necessário que o conteúdo da *Internet* seja estendido com representações formais de seu significado, isto é, adicionar declarações textuais que obedeçam a um mesmo padrão sintático e estrutural, onde os termos utilizados são definidos em vocabulários formais e compartilhados. Somente a partir desta padronização e compartilhamento de definições, um software é capaz de relacionar informações que antes só eram compreendidas por seres humanos. A Seção 2 aborda com mais detalhes as tecnologias utilizadas na *Web* Semântica para a representação formal do conhecimento.

Para chegar ao estágio onde o conteúdo da *Internet* esteja nos padrões da *Web* Semântica, é necessário utilizar alguns procedimentos que variam de acordo com a estrutura de cada fonte de informação. As principais estruturas identificadas são os *sites* estáticos, cujas informações encontram-se no formato HTML (*Hiper Text Markup Language* [RAG 99]), e os *sites* dinâmicos, que possuem seus dados armazenados e estruturados principalmente em bancos de dados relacionais.

O foco deste trabalho está voltado para um método o qual permite que conteúdos armazenados em bancos de dados relacionais possam ser publicados em um formato passível de processamento inteligente, ou seja, dentro dos padrões estebelecidos pela *Web* Semântica. Para que esta publicação aconteça adequadamente, o método deve ser aplicado de forma que as anotações semânticas sejam produzindas no mesmo momento que houver uma solicitação. Estas anotações devem, ainda, estar vinculadas a um vocabulário formal que defina e relacione seus termos com outras fontes de dados do mesmo domínio de conhecimento.

Chris Bizer [BIZ 03] definiu uma linguagem para a especificação de mapas de migração semântica. Estes mapas contêm as instruções necessárias para que declarações formais sejam criadas em tempo real. Adicionalmente, Chris Bizer desenvolveu um sistema capaz de realizar esta migração, gerando arquivos que declaram formalmente o conteúdo de tal banco de dados.

Contudo, esta arquitetura proposta por Chris Bizer não possui nenhum método automatizado para a criação de vocabulários que definem os termos utilizados nas anotações ou declarações geradas. Também é possível dizer que este método deveria ser mais simples de ser aplicado, pois, através desta abordagem, o mapa de migração deve ser criado manualmente pelo administrador do banco de dados.

A ferramenta desenvolvida no presente trabalho chama-se *D2R Editor*, e permite que o usuário registre informações sobre a estrutura de determinado banco de dados, através de uma interface gráfica. Com isto, o sistema realiza a geração automática de um mapa de migração de tal banco de dados, na linguagem especificada por Chris Bizer. Como resultado adicional, a ferramenta produz um vocabulário de definições ricas em uma semântica formal, o qual define o conceito de todos os termos utilizados neste banco de dados, viabilizando a integração com outras fontes de informação.

1.1 Importância e Justificativa

A publicação, por Tim Berners-Lee, do artigo "*The Semantic Web*" [BER 01] movimentou pesquisadores de várias áreas, pois, além da proposta de revolucionar o uso da *Internet*, foram mencionadas formas de representação do conhecimento de modo a viabilizar a compreensão por parte de novos agentes inteligentes. Deste então, a organização que regulamenta todos os padrões utilizados na *Web* [W3C 04] deu início a um grupo oficial de pesquisas sobre a *Web* Semântica e, além disso, muitos outros trabalhos vêm sendo desenvolvidos, com as mais diversas abrangências dentro deste tema.

Handschuh [HAN 03] apresenta uma abordagem onde usuários utilizam ferramentas para ajustar seus próprios vocabulários formais com as anotações semânticas encontradas em outros *sites* que já produzem conteúdos ricos em uma semântica formal. Os componentes disponibilizados contribuem muito para eliminar

ambigüidades entre fontes de dados distintas. Um outro trabalho, publicado por Grau [GRA 04], apresenta uma proposta para simplificar a arquitetura da *Web* Semântica, onde são sugeridas algumas modificações importantes aos padrões aceitos pela W3C.

Já sobre a migração de bases de dados relacionais para anotações semânticas, o trabalho publicado por Stojanovic [STO 02] apresenta um sistema onde o usuário, primeiramente, informa toda a estrutura de um modelo relacional e, em seguida, acompanha um processo semi-automático de geração de um vocabulário, que representa formalmente todo o modelo relacional informado. O sistema ainda viabiliza a criação das instâncias de declarações formais contendo as informações do banco de dados de acordo com a estrutura do vocabulário.

As tecnologias que circundam o desenvolvimento de uma *Web* inteligente são importantes para muitas áreas da ciência da computação, fazendo com que todas trabalhem em cooperação, desde a representação formal do conhecimento, passando pela pesquisa sobre novos modelos de interface com o usuário final, até a interpretação inteligente de conteúdos.

Com relação à aplicabilidade desta tecnologia, pode-se dizer que o uso do computador poderá se tornar uma experiência nova em todos os domínios. Isto por que o usuário não receberá mais informações desconexas, mas sim resultados de processamentos inteligentes, os quais relacionaram dados de diversos locais. Sendo assim, a utilização da *Internet* ficará mais rápida, prática, e eficiente.

1.2 Objetivos

Permitir com que administradores de sites dinâmicos e bancos de dados relacionais possam disponibilizar na *Internet*, com facilidade, conteúdos de acordo com os padrões da *Web* Semântica. Assim, as informações poderiam ser trocadas e processadas entre agentes inteligentes, o que tornaria a *Internet* um meio de comunicação ainda mais útil em diversos segmentos da sociedade.

1.2.1 Objetivos Específicos

- Desenvolver um sistema capas de gerar mapas de migração e esquemas RDF [MAN 03] de quaquer banco de dados relacional;

- Desenvolver uma interface gráfica para o registro de informações sobre a estrutura de um banco de dados;
    - Armazenar informações sobre mais de uma base de dados;
    - Permitir que o usuário aponte detalhes sobre a estrutura relacional da fonte de dados;
    - Registrar relacionamentos entre os termos utilizados neste banco de dados com termos definidos em outros vocabulários da *Web*;
- Criar um mapa de migração de acordo com as especificações definidas por Chris Bizer, baseando-se nas informações registradas sobre a estrutura de um banco de dados;
- Gerar um vocabulário formal o qual defina os termos utilizados no mapa de migração, baseando-se nas mesmas características informadas sobre determinado banco de dados;
    - Relacionar as definições deste vocabulário com termos definidos em outros vocabulários da *Web*;

1.3 Limitações

- O trabalho possui o objetivo de gerar mapas de migração e vocabulários semânticos sobre informações contidas apenas em bancos de dados relacionais, não abrangendo nenhum processo de anotação semântica de informações que estejam em qualquer outro formato;
- Com o *D2R Editor* não é possível gerar vocabulários contendo descrições em mais de um idioma;
- O sistema não faz nenhum relacionamento automático com outras definições de vocabulários espalhados pela *Web*, mas fornece uma interface para que o usuário os faça manualmente;
- O processo de reconhecimento da estrutura do banco de dados em questão não é automático, mas sim através de uma interface onde o

usuário registra quais informações deverão constar no mapa de migração e no vocabulário;

- O *D2R Editor* não é capaz de produzir as declarações formais dos dados contidos em um banco de dados. O sistema gera apenas um vocabulário formal e um mapa de migração que irá viabilizar este processo, que pode ser realizado pelo sistema apresentado por Chris Bizer.

1.4 Organização do Texto

Este trabalho está estruturado em sete capítulos.

O primeiro faz uma introdução do tema abordado, identificando objetivos, apontando limitações, ressaltando a importância e justificando o trabalho realizado.

O segundo capítulo apresenta a *Web* Semântica, explicando seu funcionamento e descrevendo as tecnologias que a envolvem.

O terceiro capítulo identifica formas de construir a base da *Web* Semântica ressaltando, inicialmente, a necessidade da existência de métodos simples para a descrição da semântica formal dos conteúdos da *Internet*.

O quarto capítulo apresenta o sistema *D2R Editor*, descrevendo sua estrutura, funcionamento e implementação. Ao fim do capítulo é apresentado um exemplo de uso desta aplicação.

No quinto capítulo são descritas as conclusões do trabalho desenvolvido, apresentado os resultados e indicando possibilidades para trabalhos futuros.

O sexto capítulo contém todas as referências bibliográficas utilizadas, e o sétimo capítulo é composto por alguns anexos que contribuem para a compreensão do presente texto.

2 A *WEB* SEMÂNTICA

A *Internet* contém uma enorme quantidade de informações criadas pelas mais diversas organizações, pessoas e comunidades, cada qual com seus objetivos específicos. Para interagir com este universo, o usuário deve, ou especificar o endereço do *site* desejado, ou filtrar resultados em mecanismos de busca, ou clicar em *links* relacionados, dentre outras ações simples. Esta simplicidade foi definitiva para tanta popularidade e crescimento deste meio de comunicação.

Todavia, o crescimento da *Internet* trouxe uma série de conseqüências relacionadas ao acesso e organização de seu conteúdo. Com uma gama cada vez maior de dados disponíveis, é comum que informações relevantes sejam perdidas em uma busca, ou que conteúdos não relacionados à mesma sejam retornados. Isto ocorre por que a *Web* se tornou uma grande base de dados escrita e estruturada de maneiras distintas e, sendo assim, está sendo cada vez mais difícil a realização eficiente de buscas e relacionamentos automatizados nos conteúdos da *Internet*.

A *Web* Semântica pode ser definida como uma extensão da *Web* atual, onde as informações serão complementadas por descrições ricas de uma semântica formal, e os sistemas, trabalhando em cooperação, estarão habilitados a “entender” e integrar estes conteúdos, podendo realizar tarefas muito mais complexas do que as executadas atualmente [BER 01].

Representar os conteúdos da *Web* utilizando uma semântica formal significa, primeiramente, adicionar declarações textuais utilizando um único padrão sintático e estrutural e, posteriormente, estendê-las com a criação de vocabulários que definam e compartilhem o significado dos termos destas declarações. Adicionalmente, devem ser realizadas descrições lógicas sobre fatos que envolvem estes termos. A seguir são citadas as camadas utilizadas na representação da informação de forma a torná-la “compreensível” pelo computador.

- *Extensible Markup Language* (XML) [BRA 04]: provê uma sintaxe padronizada que permite a troca de informações entre diferentes *softwares* e plataformas;

- *Resource Description Framework* (RDF): fornece um modelo uniforme para descrever dados sobre qualquer recurso na W*eb*, viabilizando a integração de informações de origens distintas;
- Ontologias: eliminam ambigüidades e adicionam detalhes semânticos estendendo declarações RDF, criando vocabulários comuns sobre conceitos de um mesmo domínio.

Nas seções 2.1, 2.2, 2.3 e 2.4 são detalhadas estas camadas utilizadas na descrição da semântica formal de conteúdos da *Web*.

Além destas camadas apresentadas, ainda existem outras duas em nível superior, sendo uma responsável pela definição de regras para inferências lógicas, e a outra determina níveis de confiança para informações com procedências diferentes. Entretanto, estes dois níveis de representação não são abordados em detalhes, visto que não fazem parte do escopo deste trabalho.

Com as informações devidamente representadas, será viável o funcionamento de aplicações e agentes de *software* preparados para ler e interpretar dados providos de uma semântica formal. Assim, estes sistemas estarão prontos para trocar informações entre si, processá-las de forma homogênea, tomar decisões inteligentes baseados nas descrições lógicas de fatos, e realizar operações que possam reduzir e otimizar o trabalho do usuário no uso da *Internet*.

2.1 XML

A Linguagem de Marcação Extensível (*Extensible Markup Language*) XML foi projetada para permitir que qualquer um possa projetar e escrever sua própria estrutura de documento [BRA 04]. Assim como HTML, o documento XML contém um texto, que é marcado com *tags*. Esta marcação permite que uma aplicação interprete as partes (chamadas de elementos) do conteúdo. Em HTML, o conjunto de *tags* permitidas é finito, já com XML os usuários podem definir suas próprias *tags* e a estrutura hierárquica na qual elas estarão dispostas no documento. O Quadro 2-1 é um exemplo simples de um fragmento de código XML.

```
<declaracao><pessoa identificador="http://example.com/andre#">André
Cantarelli</pessoa>
comeu uma <fruta>maçã</fruta>.</declaracao>
```

Quadro 2-1 - Exemplo de um código XML.

Os elementos delimitados pelas *tags* são utilizados para refletir uma estrutura particular associada à sentença. As *tags* permitem que o documento seja processado por um *software* desenvolvido para entender especificamente a estrutura destes elementos. Por exemplo, um dos elementos do Quadro 2-1 é `<fruta>maçã</fruta>`, e consiste na *tag* de abertura `<fruta>`, seguido do conteúdo `maçã`, e da *tag* de fechamento `</fruta>`. O elemento `fruta`, juntamente com o elemento `pessoa`, estão aninhados como parte do elemento `declaracao`.

Em alguns casos um elemento pode não ter um conteúdo. Isto pode ser escrito de duas formas, onde uma se dá no uso de uma *tag* de abertura, seguida do seu fechamento (`<fruta></fruta>`). A outra ocorre simplesmente com a *tag* de abertura junto da barra de fechamento no final (`<fruta />`).

Existem casos onde a *tag* de abertura pode ser acrescida de informações de caracterização ao invés de somente uma outra *tag* dentro de sua estrutura. É o exemplo do elemento `pessoa` que possui o atributo `identificador=http://example.com/#andre`. Um atributo consiste em um nome, um sinal de igualdade e um valor dentro de aspas.

O código do Quadro 2-1 usa termos particulares (`declaracao`, `pessoa`, `fruta`) como nomes de *tags* visando dar um significado para a leitura do conteúdo. Esta semântica teria efeito somente em casos onde o consumidor desta informação fosse, ou um ser humano, ou um *software* programado para interpretar, especificamente, os termos deste documento. Se a situação não for nenhuma destas duas, não é possível o reconhecimento semântico ou estrutural.

A linguagem XML apresenta uma estrutura totalmente flexível, permitindo que se crie qualquer outra linguagem com uma estrutura própria, de forma que sua sintaxe possa ser lida por qualquer *software*. Esta característica viabiliza a troca de informações entre aplicações e, por este motivo, o padrão de sintaxe XML foi escolhido para o desenvolvimento das linguagens de representação de informações na *Web* Semântica [BEC 04].

2.1.1 URIs e Namespaces XML

URI (*Uniform Resource Identifier*) é uma forma de identificação de recursos na *Web* [BER 98]. Diferentemente de URL, um URI não é limitado a identificar somente elementos que possuem um local remoto válido. Para representar um URI, é necessário um prefixo e um nome de identificação. Existem muitos formatos de URIs, sendo que os mais comuns servem para identificar páginas da *Web* (`http://www.example.org`), endereços de *e-mail* (`mailto:andre@example.org`), endereços de servidores de transferências de arquivos (`ftp://ftp.example.org`), e nomes de recursos diversos (`urn:isbn:8-556-55225-5`).

Na declaração formal de informações na *Web* existem situações onde são realizadas mais de uma definição de elementos que possuem o mesmo nome. Ocorre então, situações onde um termo possui um significado em um determinado documento, e outro completamente diferente em um segundo documento.

Para solucionar esta questão, foi implantado na estrutura da linguagem XML um mecanismo adicional objetivando manter únicos os vocabulários, tendo os elementos de marcação identificados sem duplicidades, através do uso de *Namespaces* XML [BRA 03]. Um n*amespace* é uma forma de identificar (com uma abreviação que representa um URI) um espaço ou parte da *Web* que corresponde a um conjunto específico de nomes ou termos. Através da qualificação de nomes de *tags* com um *namespace*, é possível criar vocabulários e torná-los únicos na *Internet.* O Quadro 2-2 apresenta um exemplo de uso de *namespace* para identificar os termos usados em sentenças XML.

```
<user:declaração
   xmlns:user="http://example.com/usersdef#"
>
   <user:pessoa user:identificador="http://example.com/andre#">
      André Cantarelli
   </user:pessoa>
   comeu uma <user:fruta>maçã</user:fruta>.
</user:declaracao>
```

Quadro 2-2 - Exemplo do uso de namespace e URI em um documento XML.

2.2 RDF – Resource Description Framework

O RDF é uma linguagem para representação de informações sobre recursos na *Internet* [KLY 04]. Inicialmente era utilizada somente para representar metadados sobre recursos específicos da *Web*, ou seja, propriedades como título, autor, ou data de modificação de documentos disponíveis na *Internet*. Entretanto, através da generalização do conceito "recurso da *Web*", o RDF pode ser usado para fazer declarações sobre objetos das mais diversas naturezas (cd, pessoa, automóvel, livro, etc.) que podem ser identificados na *Web*, sem a necessidade do acesso direto pelo *browser* ou navegador *Web*.

O RDF é utilizado quando informações sobre um ou mais recursos devem ser processadas por aplicações diferentes, independente da forma como o conteúdo é apresentado. A linguagem provê um *framework* comum para representação de dados, de modo que estes possam ser trocados entre aplicações sem haver perda de significado.

O RDF foi projetado para representar informações com o mínimo de restrições na expressão de seus significados, oferecendo grande flexibilidade na declaração dos fatos de forma que se possa chegar tão perto quanto possível da realidade. Este modelo de representação uniforme pode ser usado isoladamente em uma única aplicação, no entanto, sua estrutura facilita o compartilhamento, agregando valor à informação, uma vez que esta pode ser acessada por várias aplicações na *Internet*.

O RDF pode ser amplamente utilizado na descrição formal do significado de informações na W*eb*, pelas seguintes características [KLY 04]:

- Modelo de dados simples: sua estrutura permite uma fácil interpretação por parte das aplicações, sendo independente de qualquer sintaxe;
- Semântica e inferência formais: possui uma semântica formal que provê uma base para o raciocínio sobre o significado de uma expressão RDF, bem como para a definição de regras de confiança e inferência;
- Vocabulário extensível baseado em URIs: apresenta uma grande flexibilidade para utilizar tipos de objetos diferentes, que podem ser identificados de forma única em toda a *Web*;

- Total integração com XML: é possível realizar declarações RDF utilizando a sintaxe XML, juntamente com os valores que estão de acordo com os tipos de dados definidos no esquema XML, facilitando mais ainda a troca de informações entre aplicações;
- Qualquer um pode realizar declarações sobre qualquer recurso: para facilitar operações na escala da *Internet*, o RDF é aberto no sentido de que a informação pode ser construída por todos que a compartilham.

2.2.1 O Modelo RDF

O modelo RDF assume que cada recurso tem uma ou mais propriedades que possuem valores. Sendo assim, os dados são representados pela estrutura "Sujeito – Predicado – Objeto"; Sujeito é o alvo da declaração, um recurso real ou virtual que sempre será representado por um URI; Predicado identifica a propriedade, a característica do sujeito; e Objeto determina o valor da propriedade, podendo assumir uma forma literal ou referenciar um outro recurso através de um URI [DEC 00].

Utilizando URIs para referenciar recursos, é possível realizar várias declarações sobre um mesmo conceito nos mais diferentes locais da *Web*, permitindo o surgimento uma rede de conhecimento.

O modelo de dados do RDF pode ser expresso de uma forma abstrata através de grafos titulados e dirigidos, conforme mostra a Figura 2-1.

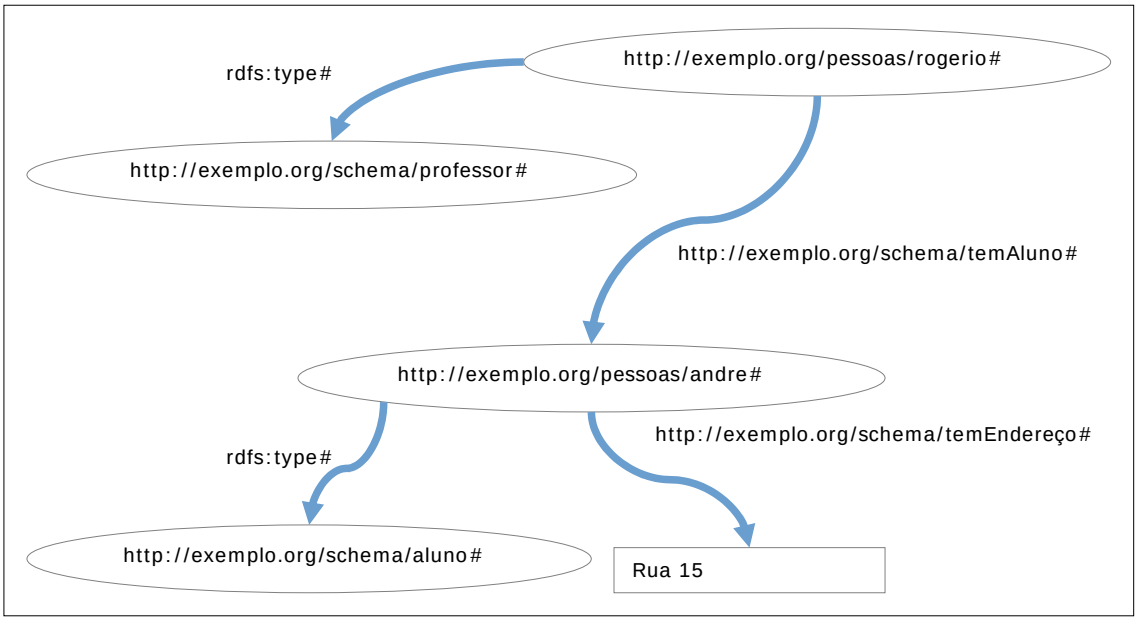

Figura 2-1 - Declarações RDF na forma de grafos titulados e dirigidos

As declarações RDF da Figura 2-1 formam as seqüências "sujeito-predicado-objeto" apresentadas no Quadro 2-3.

Sujeito	Predicado	Objeto
http://exemplo.org/rogerio#	é do tipo	Professor
http://exemplo.org/rogerio#	possui o aluno	http://exemplo.org/andre#
http://exemplo.org/andre#	é do tipo	Aluno
http://exemplo.org/andre#	possui o endereço	Rua 15

Quadro 2-3 - Declarações RDF

2.2.2 Identificação de Recursos

Uma referência URI é um identificador único que é atribuído a um recurso, objeto, ou propriedade, permitindo que este seja referenciado ou apontado em qualquer declaração RDF, independente de sua identificação remota. No modelo RDF, um grafo ou nodo pode ter como valor uma referência URI, um literal ou um valor em branco (*blank node*). Um literal ou uma referência URI usados como um nodo, identificam o

que este nodo representa. Uma propriedade ou predicado é também uma referência URI, e identifica um relacionamento entre dois nodos de forma global.

Um nodo em branco (*blank node*) é um nodo que não é nem um literal, nem uma referência URI. Serve como uma referência local para identificar um conjunto de declarações sobre um determinado recurso.

2.2.3 Tipos de Dados

Tipos de dados são utilizados no RDF para representação de valores, tais como números inteiros, datas e *strings*. No entanto, não existe nenhum conceito pré-definido sobre o uso de tipos de dados comuns, ao invés disso, tipos de dados em RDF são definidos separadamente, identificados com referências URI. Os tipos de dados definidos no Esquema XML [FAL 04] são utilizados como referências para este propósito, pois, além de possuírem as definições dos tipos de dados mais conhecidos, apresentam recursos que viabilizam novas especificações.

2.2.4 Literais

Literais são usados para identificar valores como números e datas, através do significado de uma representação léxica. Qualquer coisa representada por um literal pode também ser representado por um URI. Um literal pode ser o objeto de uma declaração RDF, mas não o sujeito de um predicado. Um literal pode ser plano (*plain*) ou com um tipo definido (*typed*). Um literal plano é uma string convencional podendo ser combinada com uma *tag* que determina a linguagem, e um literal com um tipo definido é uma string combinada com um URI que aponta para um tipo de dados específico.

2.2.5 Sintaxe XML para RDF

Representações de declarações que utilizam o modelo RDF podem ser representadas através de grafos com nodos e arcos dirigidos e titulados, onde cada declaração contém três elementos: um sujeito, um predicado e um objeto. Os nodos podem ser referências URI, valores literais, ou nodos em branco. Entretanto, para viabilizar a interoperabilidade entre aplicações diferentes, é necessário serializar grafos RDF utilizando a sintaxe XML, ou seja, transformá-los em uma seqüência de caracteres fazendo uso de uma lógica estrutural e sintática. Para isto, os predicados e nodos podem

ser representados como termos XML – nomes de elementos, nomes de atributos, conteúdos de elementos e valores de atributos.

Um grafo de declarações RDF é um conjunto de nodos que representam sujeitos e objetos, interligados por arcos que representam predicados. Com RDF/XML, estas declarações tornam-se seqüências de elementos dentro de outros elementos que representam os nodos e os arcos. Um nodo que inicia uma seqüência em um grafo torna-se o elemento raiz em RDF/XML, já o arco que está ligado a este nodo torna-se um elemento filho, e assim por diante [BEC 04].

2.2.5.1 Elementos Nodos e Propriedades

O caminho da esquerda, destacado no grafo da Figura 2-2, é composto por 3 nodos (sujeitos) e 2 arcos (predicados). Para representá-los em XML é necessário o uso de elementos dos tipos nodos e propriedades, respectivamente. No Quadro 2-4, os elementos `rdf:Description` representam os nodos (sujeitos), ao passo que `ex:editor` e `ex:site` representam as propriedades (arcos).

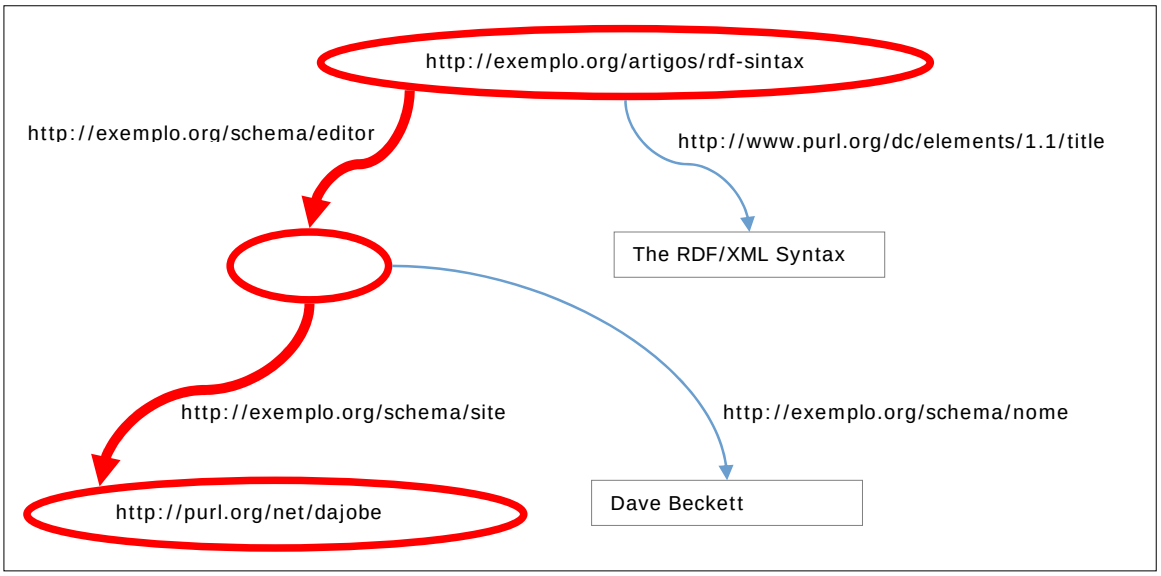

Figura 2-2 - Exemplo de grafo RDF

O RDF/XML usa nomes qualificados (XML *QName* - definido no *namespace* XML) para representar e abreviar referências URI. Todo o nome qualificado é uma pequena seqüência de caracteres a qual está atribuída a uma referência URI. Estes nomes qualificados podem ser utilizados para abreviar as referências URI de predicados, sujeitos e objetos dentro de um documento RDF/XML. O conjunto de

nomes qualificados que abreviam referências URI em um documento XML forma *namespace* deste documento.

```
<rdf:Description>
   <ex:editor>
      <rdf:Description>
         <ex:site>
            <rdf:Description>
            </rdf:Description>
         </ex:site>
      </rdf:Description>
   </ex:editor>
</rdf:Description>
```

Quadro 2-4 - Nodos e Arcos RDF/XML

Alguns dos nodos da Figura 2-2 são referências URI. Estas são declaradas com RDF/XML usando o atributo `rdf:about` nos elementos `rdf:Description`, como mostra o Quadro 2-5.

```
<rdf:Description rdf:about="http://exemplo.org/artigos/rdf-sintax">
   <ex:editor>
       <rdf:Description>
          <ex:site>
             <rdf:Description rdf:about="http://purl.org/net/dajobe">
             </rdf:Description>
          </ex:site>
       </rdf:Description>
   </ex:editor>
</rdf:Description>
```

Quadro 2-5 - Nodos com Referências URI

Adicionando os dois caminhos restantes, que não foram destacados na Figura 2-2, é possível notar que o nodo em branco dá origem a dois caminhos, ou seja, possui duas propriedades. O mesmo acontece com o nodo raiz, identificado pela referência URI `http://exemplo.org/artigos/rdf-sintax`. O Quadro 2-6 exibe a descrição completa de todos os caminhos do grafo.

```
<rdf:Description rdf:about="http://exemplo.org/artigos/rdf-sintax">
   <ex:editor>
       <rdf:Description>
          <ex:site>
             <rdf:Description rdf:about="http://purl.org/net/dajobe">
             </rdf:Description>
          </ex:site>
       </rdf:Description>
   </ex:editor>
</rdf:Description>

<rdf:Description rdf:about="http://exemplo.org/artigos/rdf-sintax">
   <ex:editor>
       <rdf:Description>
              <ex:nome>Dave Beckett</ex:nome>
       </rdf:Description>
   </ex:editor>
</rdf:Description>

<rdf:Description rdf:about="http://exemplo.org/artigos/rdf-sintax">
   <dc:titulo>The RDF/XML Syntax</dc:titulo>
</rdf:Description>
```

Quadro 2-6 - Grafo RDF serializado com a sintaxe XML

2.2.5.2 Abreviaturas RDF/XML

Muitos tipos de abreviaturas podem e devem ser utilizados para simplificar as serializações de grafos RDF, pelo fato de que os nodos (sujeitos) geralmente possuem vários arcos (predicados) os quais descrevem suas características. Sem abreviaturas, o código XML apresentará muitas repetições, o que aumentará, desnecessariamente, o tamanho e a complexidade dos arquivos.

2.2.5.2.1 Elementos com mais de uma propriedade

No Quadro 2-6, o elemento utilizado para representar o sujeito que possui a referência URI `http://exemplo.org/artigos/rdf-sintax` é declarado duas vezes, pois possui duas propriedades: `ex:editor` e `dc:titulo`. O mesmo ocorre com o elemento que representa o nodo em branco, o qual possui `ex:site` e `ex:nome`.

Dentre os recursos de abreviatura providos pelo RDF/XML, está o uso de mais de um elemento XML que representa uma propriedade dentro de um elemento XML que representa um sujeito. Assim, é possível a definição de várias propriedades para um sujeito, sem a necessidade de várias declarações semelhantes, como é apresentado no Quadro 2-7.

```
<rdf:Description rdf:about="http://exemplo.org/artigos/rdf-sintax">
   <ex:editor>
      <rdf:Description>
         <ex:site>
            <rdf:Description rdf:about="http://purl.org/net/dajobe">
            </rdf:Description>
         </ex:site>
         <ex:nome>
            Dave Beckett
         </ex:nome>
      </rdf:Description>
   </ex:editor>
   <dc:titulo>The RDF/XML Syntax</dc:titulo>
</rdf:Description>
```

Quadro 2-7 - RDF/XML utilizando várias propriedades para um único elemento

2.2.5.2.2 Elementos de propriedades em branco

Quando um arco predicado aponta para um nodo que não aponta para nenhum sujeito, este nodo é representado no RDF/XML como um elemento nodo em branco `<rdf:Description rdf:about="http://www.exemplo.com/#b"> </rdf:Description>` (ou `<rdf:Description rdf:about="http://www.exemplo.com/#b" />`). Estas declarações RDF/XML podem ser abreviadas usando o URI do elemento nodo em branco como valor de um atributo chamado `rdf:resource`, instanciado no elemento que contém o nodo em branco.

No Quadro 2-7 esta abreviatura pode ser aplicada no elemento identificado pela referência URI `http://purl.org/net/dajobe`, que é um elemento de propriedade em branco. Neste caso o elemento `ex:site` ganhará o atributo `rdf:resource`, e seu elemento nodo filho desaparecerá, conforme o Quadro 2-8.

```
<rdf:Description rdf:about="http://exemplo.org/artigos/rdf-sintax">
   <ex:editor>
      <rdf:Description>
         <ex:site rdf:resource="http://purl.org/net/dajobe" />
         <ex:nome>Dave Beckett</ex:nome>
      </rdf:Description>
   </ex:editor>
   <dc:titulo>The RDF/XML Syntax</dc:titulo>
</rdf:Description>
```

Quadro 2-8 - RDF/XML com a abreviatura para elementos de propriedades em branco

2.2.5.3 Propriedades Atributo

Quando um elemento que representa uma propriedade contém uma *String* Literal, é possível que este elemento se torne um atributo do seu elemento raiz, desde que não existam ocorrências repetidas desta propriedade no mesmo elemento nodo.

No exemplo atual, é possível notar que os valores das propriedades `ex:titulo` e `ex:nome` são *Strings* Literais e não se repetem (não existem mais de um título e, tampouco, mais de um nome), portanto podem ser incorporadas como propriedades atributos de seus elementos nodo, de acordo com o Quadro 2-9.

```
<rdf:Description
rdf:about="http://exemplo.org/artigos/rdf-sintax"
ex:titulo="The RDF/XML Syntax">
   <ex:editor>
      <rdf:Description ex:nome="Dave Beckett">
         <ex:site rdf:resource="http://purl.org/net/dajobe" />
      </rdf:Description>
   </ex:editor>
</rdf:Description>
```

Quadro 2-9 - Exemplo do uso de propriedades atributos

2.2.6 Completando o Documento RDF/XML

Para criar um documento RDF/XML completo, a serialização do grafo em XML deve estar dentro do elemento XML `rdf:RDF`, que é o elemento raiz de todo o documento. A especificação do XML também exige uma declaração XML no topo do documento que define a versão da Linguagem XML e, opcionalmente, o tipo de

codificação do conteúdo. O Quadro 2-10 mostra o código RDF/XML do Quadro 2-9 juntamente com os elementos necessários para a boa formatação de um documento RDF/XML. Na listagem a seguir possível notar, também, a existência das atribuições dos nomes qualificados utilizados no documento.

```
<?xml version="1.0" encoding="utf-8">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc=" http://purl.org/dc/elements/1.1/"
xmlns:ex="http://exemplo.org/rdfschema/">

   <rdf:Description rdf:about="http://exemplo.org/artigos/rdf-sintax"
   ex:titulo="The RDF/XML Syntax">
      <ex:editor>
         <rdf:Description ex:nome="Dave Beckett">
            <ex:site rdf:resource="http://purl.org/net/dajobe" />
         </rdf:Description>
      </ex:editor>
   </rdf:Description>

</rdf:RDF>
```

Quadro 2-10 - Documento RDF/XML completo

2.2.7 Descrevendo Recursos em Diversas Línguas

O RDF/XML permite o uso do atributo `xml:lang` [BRA 04] , para viabilizar a identificação do idioma do conteúdo dos elementos XML. O atributo `xml:lang` pode ser utilizado dentro de qualquer elemento nodo ou elemento propriedade para indicar que o conteúdo ali incluído está escrito em um determinado idioma, de acordo com o Quadro 2-11.

```
<ex:titulo xml:lang="en">
   The RDF/XML Syntax
</ex:titulo>
<ex:titulo xml:lang="pt-br">
   A Sintaxe RDF/XML
</ex:titulo>
```

Quadro 2-11 - Uso do atributo xml:lang

2.2.8 Tipos de Literais

Com RDF/XML é possível especificar o tipo de dados dos valores literais de um predicado. Para isto, é adicionado o atributo `rdf:datatype="datatypeURI"` ao respectivo elemento propriedade. O Quadro 2-12 mostra uma referência URI, definida no esquema XML [FAL 04], sendo utilizada para identificar o tipo dos possíveis valores da propriedade `ex:tamanho`. Entretanto, qualquer referência URI pode ser utilizada para especificar tipos de dados.

```
<rdf:Description rdf:about="http://exemplo.org/item1">
   <ex:tamanho rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
      115
   </ex:tamanho>
</rdf:Description>
```

Quadro 2-12 - Literais com tipos de dados definidos

2.2.9 Identificação de Nodos em Branco

Nodos em branco são elementos que não possuem nenhuma referência URI como identificador. São utilizados quando for necessário agrupar um conjunto de outras declarações em um único elemento Este grupo de elementos é identificado através do atributo `rdf:nodeID`, que só é reconhecido por elementos do documento onde é definido. Um exemplo de aplicação é apresentado no Quadro 2-13.

```
<rdf:Description rdf:about="http://exemplo.org/artigos/rdf-sintax"
ex:titulo="The RDF/XML Syntax">
   <ex:editor rdf:nodeID="abc"/>
</rdf:Description>

<rdf:Description rdf:nodeID="abc" ex:nome="Dave Beckett">
   <ex:site rdf:resource="http://purl.org/net/dajobe" />
</rdf:Description>
```

Quadro 2-13 - Identificação de nodos em branco

2.2.10 Conjuntos de Membros de Elementos Propriedades

O RDF/XML possui as classes de elementos nodos `rdf:Seq`, `rdf:Bag` e `rdf:Alt` que podem servir como conjunto de elementos propriedades. Estas propriedades podem ser enumeradas e ordenadas, sendo representadas pelos elementos da classe `rdf:_1`,

`rdf:_2` e assim sucessivamente, ou da classe `rdf:li`. O Quadro 2-14 apresenta um exemplo de uso destes recursos.

```
<rdf:Seq rdf:about="ex:fruta_favorita">
   <rdf:_1 rdf:Resource="ex:banana"/>
   <rdf:_2 rdf:Resource="ex:melao"/>
   <rdf:_3 rdf:Resource="ex:mamao"/>
</rdf:Seq>
```

Quadro 2-14 - Conjunto de membros de elementos propriedades

2.2.11 Coleções

O RDF/XML permite também que o elemento propriedade seja definido como uma coleção de elementos nodos, para isto basta adicionar `rdf:parseType="Collection"` como um atributo desta propriedade, conforme é demonstrado no Quadro 2-15.

```
<rdf:Description rdf:about="ex:cesta">
   <ex:temFruta rdf:parseType="Collection">
      <rdf:Description rdf:about="ex:banana"/>
      <rdf:Description rdf:about="ex:laranja"/>
      <rdf:Description rdf:about="ex:mamao"/>
   </ex:temFruta>
</rdf:Description>
```

Quadro 2-15 - Propriedade que contém uma coleção de nodos

2.2.12 Reificação de Declarações

O atributo `rdf:ID` pode ser usado em um elemento propriedade para reificar uma declaração em um documento RDF/XML. Este identificador serve para construir uma referência URI que aponta para a declaração. Esta referência URI será o resultado da concatenação do URI do documento, do símbolo "`#`", e do valor do atributo `rdf:ID`. No Quadro 2-16 é demonstrada a declaração de uma propriedade que pode ser referenciada diretamente por qualquer outro documento na *Web* através do URI `http://exemplo.org/#dave_beckett`.

```
<ex:editor>
   <ex:nome rdf:ID="dave_beckett">Dave Beckett</ex:nome>
</ex:editor>
```

Quadro 2-16 – Reificação de uma declaração RDF/XML

2.3 O Esquema RDF

Declarações RDF são realizadas utilizando termos de domínios específicos, tais como `ex:nome`, `ex:editor`, entre outros. Porém, para viabilizar o processamento inteligente destas informações, é necessário que estes termos estejam previamente estruturados e relacionados. Estas definições são realizadas através da construção de vocabulários RDF, com a utilização de termos do Esquema RDF [BRI 03].

O Esquema RDF é uma extensão do modelo RDF, onde, o conjunto de termos de seu vocabulário é incrementado com alguns itens específicos. Com este conjunto definido de nomes, é possível criar novos vocabulários RDF compostos por termos de qualquer domínio, dependendo de cada necessidade. Sendo assim, estas novas estruturas podem ser instanciadas em declarações RDF convencionais sobre os mais diversos recursos [MAN 03].

Todos os termos utilizados na descrição de vocabulários RDF devem ser previamente conhecidos por aplicações projetadas para interpretar dados no formato RDF/XML. Estes termos são capazes de descrever formalmente a estrutura de recursos na *Web*, através de uma abordagem semelhante à orientada a objetos, onde são declaradas Classes, Subclasses e Propriedades.

Da mesma forma que o RDF, o Esquema RDF possui um documento no qual estão definidos todos os seus termos. Este vocabulário possui a referência URI `http://www.w3.org/2000/01/rdf-schema#` e, convencionalmente, é abreviado pelo prefixo (*QName*) `rdfs`.

2.3.1 Definição de Classes

O primeiro passo no processo de descrever vocabulários que definem a estrutura recursos RDF na *Web* é identificar o tipo dos objetos que serão mencionados, ou seja, as categorias as quais os recursos pertencem. O Esquema RDF trata estes “tipos de objetos” como classes. Uma classe no Esquema RDF é um conceito genérico de um tipo ou categoria. Para realizar declarações RDF sobre diferentes tipos de “veículos”, por exemplo, é necessária a definição de uma classe que representa objetos (ou recursos) que são “veículos”. As declarações sobre os veículos em si serão chamadas de instâncias da classe “veículo”.

Para definir uma classe com o Esquema RDF, basta apenas declarar que um determinado recurso, que representa tal classe, possui a propriedade `rdf:Type` com o valor `rdfs:Class`, conforme o Quadro 2-17.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://example.org/schemas/veiculos">
   <rdf:Description rdf:ID="Veículo">
      <rdf:type rdf:resource="rdfs:Class"/>
   </rdf:Description>
</rdf:RDF>
```

Quadro 2-17 - Declaração de uma classe com o Esquema RDF

No exemplo do Quadro 2-17, a propriedade `rdf:Type` é usada para indicar que o recurso “Veículo” é uma instância de uma classe chamada `rdfs:Class`. Esta mesma declaração também pode ser realizada da uma forma abreviada, como mostra o Quadro 2-18.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://example.org/schemas/veiculos">
   <rdfs:Class rdf:ID="Veículo"/>
</rdf:RDF>
```

Quadro 2-18 - Declaração abreviada de uma classe com o Esquema RDF

Agora, tendo a classe “Veículo” definida, é possível fazer declarações sobre recursos do mundo real que são da categoria “veículo”. O Quadro 2-19 apresenta um exemplo de instância de “Veículo”.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ex="http://example.org/schemas/veiculos"
xml:base="http://example.org/objetos">
   <ex:Veículo rdf:ID="carro1"/>
</rdf:RDF>
```

Quadro 2-19 - Instância de uma classe definida em um Esquema RDF

Depois da criação da classe “Veículo”, é possível a definição de outras subcategorias de veículos, ou seja, classes de objetos que também são veículos, mas que apresentam algumas características especiais, como veículos de passageiros, veículos de carga, ônibus, caminhões, etc. Estas classes podem ser, primeiramente, definidas da mesma forma que a classe “Veículo” e, em seguida, declaradas como subclasses, como mostra o Quadro 2-20.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://example.org/schemas/veiculos">
   <rdfs:Class rdf:ID="Veículo"/>
   <rdfs:Class rdf:ID="VeículoDeCarga">
      <rdfs:subClassOf rdf:resource="#Veículo" />
   </rdfs:Class>
   <rdfs:Class rdf:ID="VeículoDePasseio">
      <rdfs:subClassOf rdf:resource="#Veículo" />
   </rdfs:Class>
   <rdfs:Class rdf:ID="Caminhao">
      <rdfs:subClassOf rdf:resource="#VeículoDeCarga" />
   </rdfs:Class>
   <rdfs:Class rdf:ID="Onibus">
      <rdfs:subClassOf rdf:resource="#VeículoDeCarga" />
   </rdfs:Class>
</rdf:RDF>
```

Quadro 2-20 - Declaração de classes e subclasses com o Esquema RDF

2.3.2 Definição de Propriedades

Para diferenciar cada uma das classes de um Esquema RDF, é preciso definir seus respectivos conjuntos de propriedades de acordo com as características reais de cada tipo de objeto. Para criar uma definição de propriedade em um vocabulário RDF, é utilizada a classe `rdf:property`, associada as propriedades rdfs:domain, `rdfs:range` e, opcionalmente, `rdfs:subPropertyOf`.

Todas as propriedades em RDF são descritas como instâncias da classe `rdf:property`. Sendo assim, para definir uma nova propriedade em um Esquema RDF, deve-se declarar que um determinado recurso com um URI definido tem a propriedade `rdf:type` com o valor `rdf:property`. O Quadro 2-21 mostra duas formas de declaração

de uma propriedade RDF. A primeira na forma convencional, e a segunda no formato abreviado.

```
<!—- Formato convencional -->
<rdf:Description rdf:ID=“temNome”>
   <rdf:type rdf:resource=“rdf:Property” />
</rdf:Description>

<!—- Formato Abreviado -->
<rdf:Property rdf:ID=“temNome” />
```

Quadro 2-21 - Formas de declarações de uma propriedade RDF

Além da definição de propriedades, o Esquema RDF ainda provê recursos específicos para a especificação dos tipos de valores permitidos nas instâncias RDF. Estes tipos podem ser identificados através da propriedade `rdfs:range`, conforme o exemplo do Quadro 2-22, onde é declarado que a propriedade `temNome` pode receber valores do tipo `http://www.w3.org/2001/XMLSchema#String`.

```
<rdf:Property rdf:ID=“temNome”>
   <rdfs:range rdf:resource=” http://www.w3.org/2001/XMLSchema#String” />
</rdf:Property>
```

Quadro 2-22 - Restringindo os tipos de dados de uma propriedade

Os tipos de dados que podem ser apontados pela propriedade `rdfs:range` não estão restritos apenas às definições do esquema XML, tais como inteiros, datas, *strings*, etc. Adicionalmente, é comum declarar que uma determinada propriedade deve assumir valores que são instâncias de outras classes definidas em qualquer esquema ou vocabulário RDF, por exemplo, a propriedade `temAluno` pode assumir valores que são do tipo `ex:Aluno`, como mostra o exemplo do Quadro 2-23.

```
<rdf:Property rdf:ID=“temAluno”>
   <rdfs:range rdf:resource=”ex:Aluno” />
</rdf:Property>
```

Quadro 2-23 - Uma Classe RDF como tipo de uma propriedade.

Para definir por completo uma propriedade em um vocabulário RDF, além de determinar quais os tipos de valores, devem ser especificadas quais as classes que têm esta propriedade como uma característica.

Ao contrário da modelagem orientada a objetos convencional, onde a definição da classe indica quais propriedades ela possui, o modelo RDF permite que a definição

de uma propriedade determine quais classes ela está vinculada. Esta especificação é chamada de definição do domínio de uma propriedade, e com isto é possível encontrar propriedades em vocabulários RDF que são características de mais de uma classe, podendo ou não possuir *namespaces* diferentes, viabilizando a formação de uma rede de definições espalhada pela *Web*.

Para identificar classes as quais uma determinada propriedade está vinculada, é utilizado o termo `rdfs:domain`, que significa declarar, por exemplo que a propriedade `temAluno` está dentro do domínio `ex:Professor`, como exemplifica o Quadro 2-24.

```
<rdf:Property rdf:ID="temAluno">
   <rdfs:domain rdf:resource="ex:Professor" />
   <rdfs:domain rdf:resource="ex2:Mestre" />
   <rdfs:range rdf:resource="ex:Aluno" />
</rdf:Property>
```

Quadro 2-24 - Definição de domínios de uma propriedade

2.4 Ontologias

Ontologia é uma palavra originalmente utilizada pela filosofia, onde seu conceito é relacionado à "descrição da existência das coisas" [CHA 99]. No entanto, o termo vêm sendo empregado também em áreas da computação, onde uma ontologia é vista como uma "*especificação formal e explícita de uma conceitualização compartilhada*" [GRU 93]. O termo "conceitualização" refere-se a um modelo abstrato o qual representa a definição de algum elemento ou fenômeno do mundo real. "Explícita" significa que os tipos de conceitos utilizados, juntamente com seus relacionamentos e restrições, são explicitamente definidos. O uso do termo "formal" é justificado pela necessidade da ontologia possuir um padrão estrutural e sintático, de modo a viabilizar o seu reconhecimento em um computador.

O modelo RDF juntamente com a sintaxe XML, permite que diferentes declarações sobre um mesmo elemento possam ser postadas em locais distintos da *Web*, com a possibilidade de apontarem umas para as outras estabelecendo relações. Isto significa que a linguagem RDF/XML viabiliza a definição formal de características e relações simples entre objetos do mundo real de uma forma compartilhada, o que caracteriza a criação de ontologias simplificadas.

Todavia, para que as futuras aplicações da *Web* Semântica possam ser realmente "inteligentes", as representações de informações realizadas com o modelo RDF/XML que alimentarão estes *softwares*, deverão ser estendidas de forma a enriquecer a semântica formal contida em suas declarações. Para isto, foi definida uma linguagem chamada OWL (*Ontology Web Language*) [SMI 03], que pode ser relacionada ou até mesmo utilizada em conjunto com declarações de vocabulários RDF/XML convencionais. A OWL é uma evolução da linguagem DAML+OIL [CON 01] e se tornou um padrão reconhecido e recomendado pela *W3C* [W3C 04]. A linguagem possui a finalidade de especificar formalmente, com o maior nível de detalhe possível, todas as características de elementos, relacionamentos, e restrições das informações sobre um determinado domínio de conhecimento.

Com o esquema RDF pode-se definir classes e subclasses, com suas propriedades, que podem ter suas sub-propriedades, domínios, e tipos de valores específicos. Através destes recursos é possível, por exemplo, compartilhar a seguinte declaração formal: "um quarteto de cordas é composto por músicos". Embora estas definições sejam suficientes para que um *software* possa relacionar informações sobre quartetos de cordas provenientes de diversas fontes, poderíamos ter problemas para definir informações tais como o número de integrantes que compõem um quarteto de cordas. Com o enriquecimento semântico de vocabulários RDF é possível compartilhar e reutilizar declarações do tipo – "um quarteto de cordas é composto por exatamente quatro músicos" – ou até mesmo – "um músico é o mesmo que um instrumentista" – dentre muitas outras definições formais. O Quadro 2-25 mostra o exemplo de um esquema RDF enriquecido com definições semânticas mas detalhadas, através de termos da linguagem OWL.

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

<rdf:Property rdf:ID="temMusico">
   <rdfs:domain rdf:resource="#QuartetoCordas" />
   <rdfs:range rdf:resource="#Pessoa" />
   <owl:sameAs rdf:resource="http://musicos.org/#temInstrumentista"/>
</rdf:Property>

<owl:Restriction>
   <owl:onProperty rdf:resource="#temMusico" />
   <owl:maxCardinality
   rdf:datatype="&xsd;nonNegativeInteger">4</owl:maxCardinality>
   <owl:minCardinality
   rdf:datatype="&xsd;nonNegativeInteger">2</owl:minCardinality>
</owl:Restriction>
```

Quadro 2-25 – Fragmento de uma ontologia definida com as linguagens OWL e RDF/XML

2.5 Um Caso de Uso da Web Semântica

A *Web* Semântica disponibiliza agentes com capacidade de realizar tarefas mais complexas para os usuários, entendendo e integrando informações provenientes de várias fontes. Para exemplificar esta efetividade, pode ser citado um exemplo de serviço na *Web* que funciona como agente planejador de atividades sociais, o qual recebe preferências de um usuário, tais como tipos de filmes, comida e esportes prediletos, e os utiliza para planejar e sugerir as atividades desta pessoa durante um dia, classificando-as por serviços mais utilizados, mais procurados e melhor conceituados no mercado.

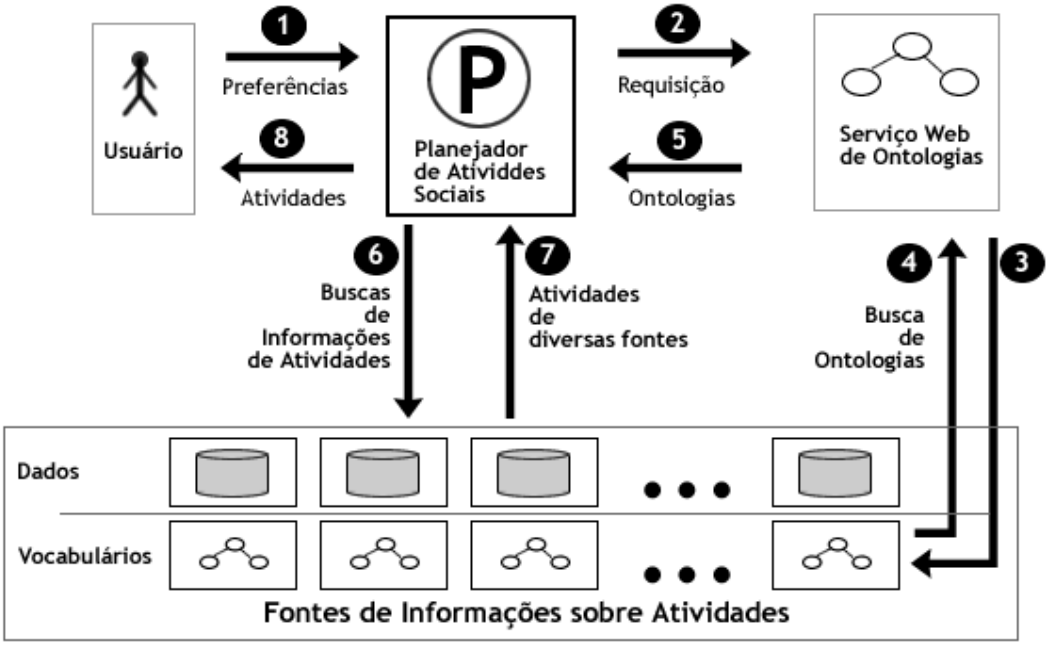

Figura 2-3 - Funcionamento de um agente planejador de atividades

Primeiro o agente recebe as preferências do usuário e comunica-se com um *software* (Serviço *Web* de Ontologias) que provê vocabulários sobre os domínios relativos a hotéis, restaurantes, cinemas, e outros serviços. Com esta informação o agente consegue realizar uma busca por serviços em várias fontes distintas, utilizando os termos corretos nas requisições e filtrando as respostas de forma eficiente, pois a ontologia esclarece formalmente o significado dos termos utilizados em cada origem, como mostra a Figura 2-3.

Tendo informações sobre as atividades de locais diferentes, em conjunto com os vocabulários que alinham o significado dos termos, é possível realizar comparações com as preferências informadas pelo usuário.

Depois de decidir quais serviços se enquadram nas preferências do usuário, pode ser feita ainda uma classificação levando em conta a qualidade. Para isto, baseando-se novamente nas ontologias, são consultados agentes que disponibilizam resultados de avaliações sobre serviços, permitindo que o usuário receba um planejamento diário composto pelas melhores atividades que se enquadram nas suas preferências.

Muitos outros exemplos poderiam ser citados para destacar os benefícios do uso da *Web* Semântica, no entanto, o fato relevante é que as ontologias permitem que várias

declarações sobre domínios do conhecimento sejam agrupadas e reaproveitadas, tendo seus significados semanticamente alinhados por agentes de *software*, permitindo que os dados sejam processados igualmente, não importando sua identificação URI ou estrutura de origem.

2.6 Considerações Finais

A *Web* atual é uma ferramenta muito eficiente na troca de informações em qualquer ramo de atividade, entretanto, com o seu crescimento, surgiu a necessidade de especializar ainda mais as suas funcionalidades. Esta especialização pode ser realizada através da criação de *softwares* que possam entender as informações disponíveis na *Internet* e processá-las uniformemente, com o objetivo de prover informações mais interessantes para os usuários.

Para se chegar a este estágio, as informações da *Web* precisam ser estendidas, e estas extensões devem ser representadas por textos ricos em uma semântica formal, composta por camadas que determinam padrões de sintaxe, estrutura, significado, lógica e confiança. Só assim a informação poderá ser interpretada pelos futuros agentes de *software* da *Web* Semântica.

3 CONSTRUÇÃO DA WEB SEMÂNTICA

A existência da *Web* Semântica depende, inicialmente, de representações formais e compartilhadas dos significados das informações que estão disponíveis na *Internet*. Para isto é necessário criar ontologias para a definição dos vocabulários de termos comuns, e utilizar estes termos em declarações na linguagem RDF/XML, estendidas pela OWL. Depois desta etapa, será viável a plena utilização de *softwares* inteligentes os quais irão consumir toda esta informação, e o processo evolutivo da *Web* estará iniciado [HAN 02].

Para realizar estas representações ou anotações semânticas, as quais são determinantes para a evolução da nova *Web*, é necessário primeiro identificar os tipos de informações que já existem e, em seguida, utilizar técnicas de anotações específicas para cada tipo identificado.

A informação na *Web* encontra-se principalmente armazenada em *sites* estáticos ou em *sites* dinâmicos e, em cada caso, são necessárias técnicas diferentes para dar significado formal aos conteúdos disponibilizados.

3.1 Anotações de Sites Estáticos

Quando um navegador *Web* tenta localizar uma página, através de um determinado endereço, na verdade ele está enviando uma requisição para um servidor *Web* que está em outro local da *Internet*. Esta requisição contém, dentre outras informações, um endereço (URL) de resposta e um nome de arquivo (com o respectivo caminho de diretórios) que deverá existir no servidor. O servidor, por sua vez, quando encontra o arquivo no caminho especificado, lê o seu conteúdo e o envia como resposta para o cliente, conforme ilustra a Figura 3-1. Estes conteúdos encontram-se no formato HTML, o qual determina as características de apresentação da informação, e não o seu significado formal.

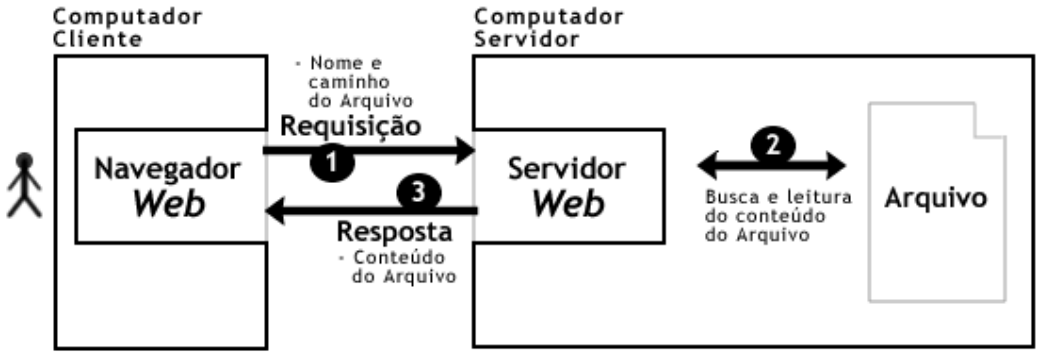

Figura 3-1 - Arquitetura de um *site* estático.

Para viabilizar a interpretação inteligente dos dados de *sites* estáticos da *Internet*, os arquivos HTML devem conter informações adicionais que descrevam formalmente a semântica de seus conteúdos. Estas anotações podem estar localizadas no cabeçalho do documento ou até mesmo serem referenciadas, caso estejam em um arquivo separado (localizado em um outro URL). Assim, o mesmo arquivo pode ser processado tanto por um navegador *Web*, quanto por um agente inteligente.

Siegfried Handschuh apresenta uma ferramenta chamada "*Ont-O-Mat*" [HAN 02], a qual permite que páginas HTML sejam editadas e anotadas ao mesmo tempo. O sistema conta com uma interface funcional e intuitiva, onde o usuário cria o conteúdo definindo o seu formato de apresentação (HTML) juntamente com o seu significado (RDF/XML).

3.2 Anotações de Sites Dinâmicos

Na arquitetura de *sites* dinâmicos, as tarefas executadas no lado cliente são exatamente as mesmas realizadas nos sites estáticos. Entretanto, no lado do servidor o procedimento é um pouco mais complexo. Quando o arquivo solicitado é encontrado, o aplicativo servidor varre o seu conteúdo em busca de instruções e as executa. Estas instruções podem ter as mais diversas funcionalidades, mas sempre com o objetivo de realizar algum processamento que retorne conteúdo HTML o qual será entregue para o cliente HTTP. Uma grande parte do *sites* da *Internet* são compostos por instruções que incluem no conteúdo HTML informações provenientes de bancos de dados relacionais, como mostra a Figura 3-2.

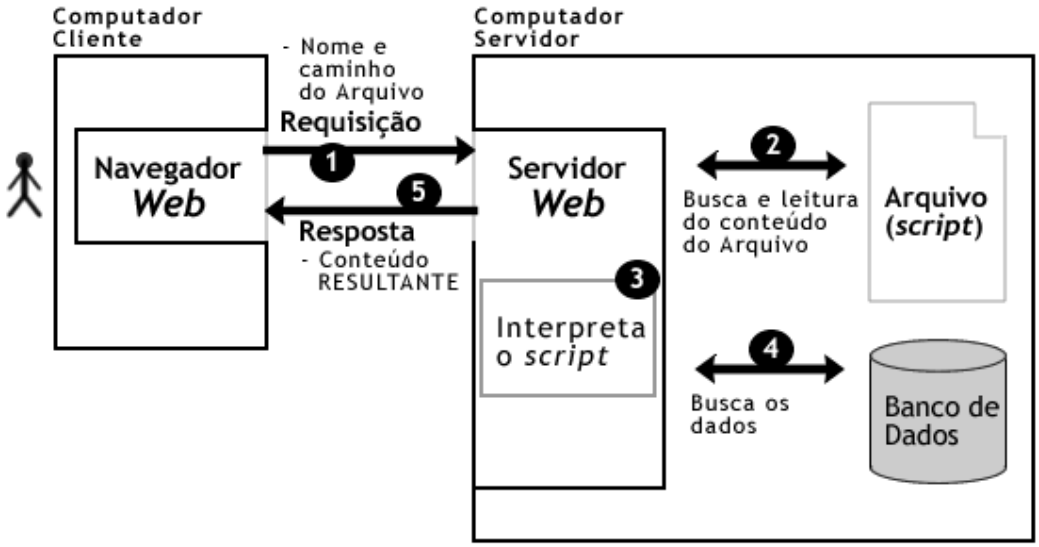

Figura 3-2 - Arquitetura de um *site* dinâmico.

Este tipo de arquitetura é a mais utilizada nos *sites* da *Internet*, pois assim, estes podem funcionar como verdadeiras aplicações distribuídas, onde usuários de qualquer lugar podem acessar e interagir ao mesmo tempo. Por este motivo, a criação de um mecanismo eficiente para a inclusão destas informações nos padrões da *Web* Semântica se torna uma tarefa muito importante para a evolução desta nova *Web* [HAN 03].

A estrutura de anotação semântica de informações de *sites* dinâmicos deve estar de acordo com as seguintes premissas.

- O cliente HTTP deve ser capaz de acessar as informações, tanto no formato de apresentação convencional (HTML), quanto na estrutura rica em semântica formal (RDF/XML);
- Os dados do banco de dados devem permanecer protegidos e acessíveis somente pelo servidor *Web*;
- A estrutura do banco de dados não deve ser alterada;
- O conteúdo RDF deve ser formado na hora da requisição HTTP, contendo as informações atuais do banco de dados, como acontece com o conteúdo HTML;

- O processo, depois de estruturado e implementado, deve funcionar automaticamente, só assim será possível entregar informações sempre atualizadas;
- As instâncias RDF geradas automaticamente devem estar vinculadas a um vocabulário ou ontologia que defina os significado de seus termos e os relacione com outros existentes, para viabilizar o processamento inteligente das informações;
- É desejável que este mecanismo possa ser aplicado por administradores de bancos de dados, para que possa ser disseminado com mais facilidade.

Na Seção 1.1 são mencionados trabalhos que estão relacionados com estas diretivas, mas nenhum chega a atender todos os objetivos citados. Entretanto Chris Bizer [BIZ 03] implementa um mecanismo que pode ser considerado o início de um *framework* capaz de atingir tais objetivos de anotação semântica em tempo real.

3.2.1 D2R MAP e Processor

Chris Bizer definiu uma linguagem XML (*D2R Map*), a qual permite a descrição de mapeamentos entre esquemas de bases de dados relacionais e esquemas RDF. O objetivo principal da linguagem é permitir mapeamentos flexíveis e automatizados de estruturas relacionais complexas, viabilizando a geração de dados escritos no modelo RDF, sem alterar o esquema da base de dados existente. Esta flexibilidade é alcançada empregando consultas SQL diretamente nas regras de mapeamento [BIZ 03].

O conteúdo do mapa serve como entrada para um sistema (*D2R Processor*), o qual executa operações de acordo com as instruções contidas no mapa, conforme mostra a Figura 3-3. Os resultados deste processamento são primeiramente agrupados em tabelas relacionais e posteriormente, utilizados para criar instâncias RDF. No entanto estas duas etapas são transparentes para o usuário, que recebe apenas as instâncias RDF resultantes. A especificação da estrutura dos elementos e atributos da linguagem *D2R Map* está no Anexo I.

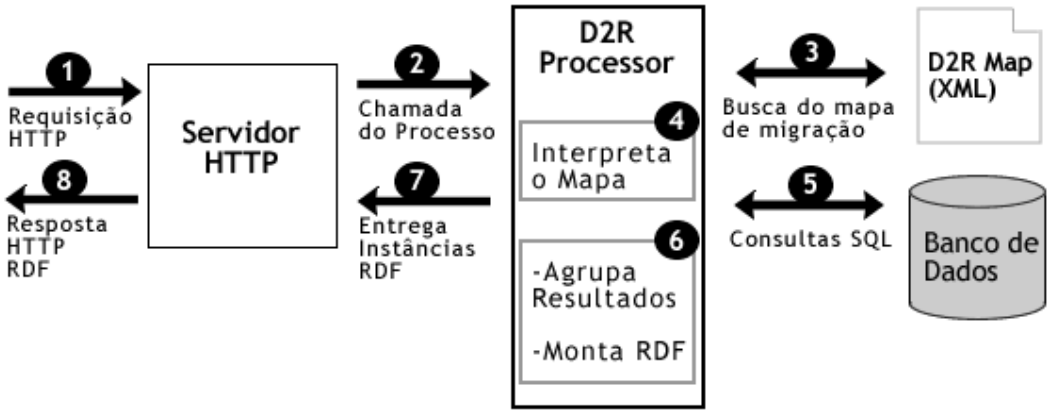

Figura 3-3 – Arquitetura de funcionamento do D2R *Processor*.

O mapa de migração é utilizado por um *software* específico, o qual utiliza as informações descritas em XML para executar operações que irão resultar na geração automática de declarações RDF sobre o conteúdo do banco de dados. O processo de mapeamento realizado pelo *D2R Processor* consiste em quatro passos, conforme a seguinte especificação [BIZ 03]:

- Seleção de um conjunto de registros da base de dados a partir de uma consulta SQL;
- Agrupamento de cada conjunto de registros utilizando a coluna indicada em `d2r:groupBy`;
- Criação de instâncias de classes e identificadores;
- Mapeamento dos dados dos conjuntos de registros para instâncias de propriedades.

O *D2R Processor* foi implementado com a linguagem de programação *Java* [JAV 04], baseado na *API Jena* [MCB 01]. O sistema exporta dados no formato RDF, no modelo *Jena*, dentre outros. Pode acessar qualquer banco de dados através de conexões ODBC (*Open DataBase Connectivity*) ou JDBC (*Java DataBase Connectivity*), podendo ser usando como um *servlet java* (para publicar dinamicamente conteúdo RDF como resposta a requisições HTTP), ou como um conector de bancos de dados em aplicações que utilizam o modelo *Jena*, ou até mesmo ser executado manualmente no sistema operacional.

3.3 Considerações Finais

O *D2R Processor* disponibiliza declarações RDF sobre o conteúdo de bases relacionais, entretanto, surge a necessidade de uma ferramenta direcionada a administradores de bancos de dados, a qual facilite a implantação desta tecnologia. Os seguintes itens podem ser apontados como oportunidades deixadas pelo trabalho de Chirs Bizer:

- Necessidade de geração simples e rápida de vários mapas de migração, de acordo com cada necessidade;
- Os vocabulários das instâncias RDF não são gerados automaticamente;
- Relacionamentos entre as informações geradas e outras ontologias não são realizados de forma automatizada;
- Necessidade de um controle mais claro sobre as informações que serão publicadas.

Com uma ferramenta que supra estas necessidades, pode ser mais simplificado o processo de criação do mapa de migração (*D2R Map*), bem como dos vocabulários RDF que definem os termos usados nas declarações geradas, de forma a viabilizar a integração dos conceitos ali definidos com outras ontologias espalhadas pela *Web*.

4 D2R EDITOR

D2R Editor é o sistema desenvolvido neste trabalho, o qual permite que administradores de bancos de dados especifiquem, através de uma interface gráfica, informações sobre a estrutura relacional de seu banco de dados. Com estas instruções, o sistema é capaz de gerar um mapa de migração (*D2R Map*), que determina as diretivas para a publicação automática de declarações RDF. O sistema gera, também, um vocabulário RDF para viabilizar o processamento inteligente destas declarações.

Este mecanismo possui o objetivo principal de facilitar a aplicação do processo apresentado por Chris Bizer [BIZ 03]. Com estes mapas e esquemas RDF, é preciso apenas integrar um *site* dinâmico ao *D2R Processor* para que instâncias RDF sobre conteúdos de um banco de dados possam ser publicadas automaticamente na *Web*. A seguir são destacadas as principais características do sistema *D2R Editor*.

- Interface gráfica para a criação de mapas de migração, permitindo a publicação automática na Web de anotações semânticas de informações de um determinado banco de dados;
    - Os mapas de migração são gerados de acordo com o padrão do sistema *D2R Processor*;
- Facilidade para geração de vocabulários RDF que descrevem formalmente conceitos da estrutura das informações extraídas de bancos de dados;
- Possibilidade de relacionamento dos termos do esquema RDF gerado, com conceitos de outras ontologias existentes na *Web*;
- Possibilidade de filtrar o conteúdo que estará disponível através de consultas SQL personalizadas, protegendo informações que não devem ser publicadas;
- Gerência de mapas de migração de mais de um banco de dados, separadamente.

4.1 Arquitetura do Sistema

O sistema é uma aplicação baseada na *Web*, e deve ser instalado em um computador que possua conexão com o banco de dados "alvo", que armazena as informações que devem ser publicadas no formato RDF/XML. Através de uma interface *Web*, o usuário passa para o sistema informações sobre a estrutura dos dados que serão publicados. O *D2R Editor*, com base nestas instruções, realiza consultas no banco de dados "alvo" para gerar os dois arquivos (*RDF Schema* e *D2R Map*) que habilitarão o *D2R Processor* a publicar instâncias RDF sobre o conteúdo do banco de dados. A Figura 4-1 apresenta uma ilustração desta arquitetura de funcionamento.

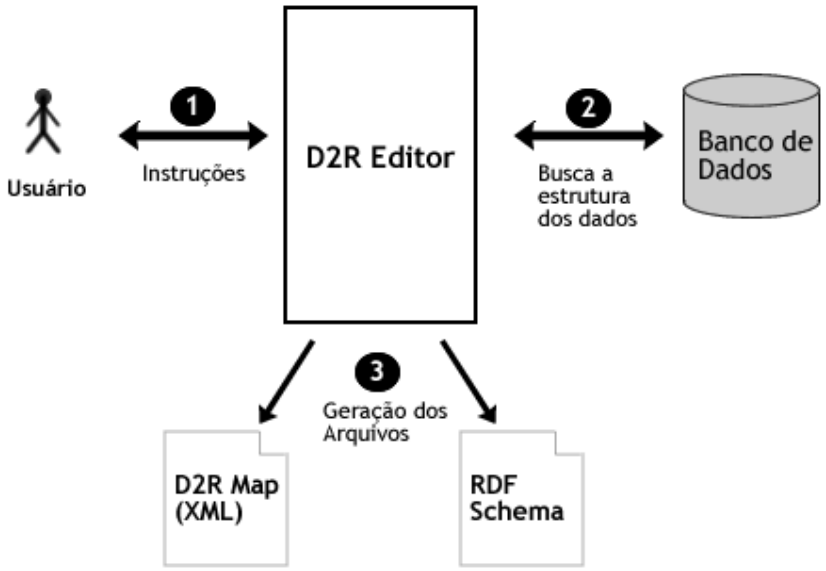

Figura 4-1 – Arquitetura do Sistema *D2R Editor*.

4.2 Funcionamento

O processo de "mapeamento" do *D2R Editor* é dividido em 5 passos, onde são registrados os dados essenciais para a criação de um mapa de migração, e um vocabulário RDF sobre tabelas de um banco de dados. Após a criação destes dois arquivos o *D2R Processor* poderá ser utilizado.

4.2.1 Passo 1 – Database Connection

Através da seção *Database Conection* ou Passo 1, o usuário configura a conexão com a fonte de dados para a geração do *D2R Map* e das instâncias RDF. Sem estas informações de conexão o sistema não consegue acessar a base de dados e, por isto, cada vez que o usuário muda estas configurações, o sistema automaticamente verifica se a fonte de dados é válida ou não. A Figura 4-2 mostra a interface de atualização dos dados de conexão com o banco de dados, onde são solicitadas as seguintes informações:

Tipo de Conexão, que pode ser ODBC ou JDBC;

Nome da Fonte de Dados ODBC, caso o tipo de conexão seja ODBC;

Nome do *Driver* e da fonte de dados JDBC, para conexões do tipo JDBC;

Dados de autenticação na fonte de dados.

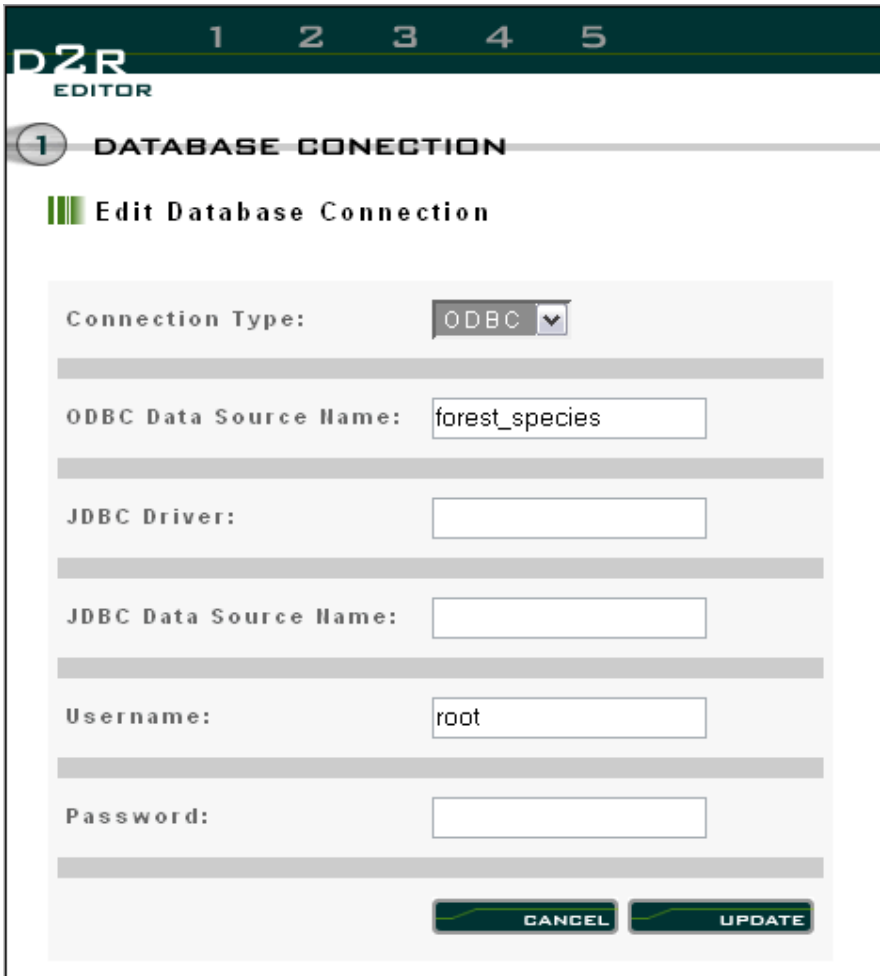

Figura 4-2 - D2R Editor: Conexão com o banco de dados

4.2.2 Passo 2 – Database Queries

Com a configuração da fonte de dados definida, o próximo passo é a definição de quais informações serão disponibilizadas no processo de geração automática de anotações semânticas. Isto é realizado através do cadastro de consultas SQL, como mostra a Figura 4-3. Estas consultas serão executadas quando o processo de migração entrar em funcionamento. Neste cadastro também é informado um nome conceitual para a consulta SQL.

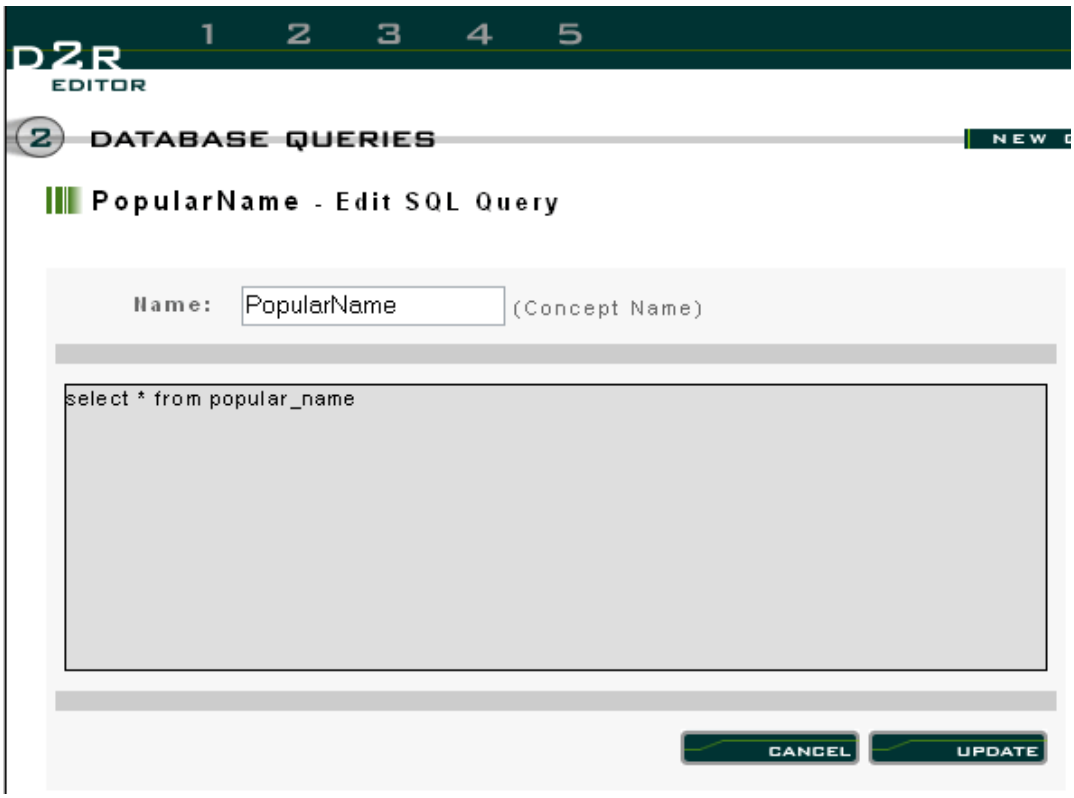

Figura 4-3 – Cadastro de uma consulta SQL no sistema.

O resultado gerado por cada consulta SQL é uma visão de uma ou mais tabelas do banco de dados analisado, sendo que estas visões são tratadas como tabelas do banco de dados pelo *D2R Editor*. Isto permite que informações banco de dados real sejam preservadas, pois, as consultas mascaram e filtram conteúdos da fonte de dados.

O conjunto de consultas informadas no sistema formam uma pseudo-modelagem relacional do banco de dados, contendo somente campos selecionados pelas instruções SQL.

Quando o usuário informa a consulta SQL, o sistema, além de registrar a informação, executa a consulta cadastrada extraindo os nomes das colunas da tabela de resultados. Tais colunas são armazenadas no banco de dados do *D2R Editor*, assim, o sistema tem acesso às colunas resultantes de cada consulta SQL informada.

Após o cadastro da consulta, o sistema apresenta ao usuário as colunas resultantes, a exemplo da Figura 4-4. O usuário, então, para cada coluna, informa um nome conceitual, assinala se a coluna faz parte da chave primária, e indica se a coluna é uma chave estrangeira. Para cada chave estrangeira informada, devem ser identificadas a tabela e a coluna correspondentes. A Figura 4-5 mostra a interface desta etapa.

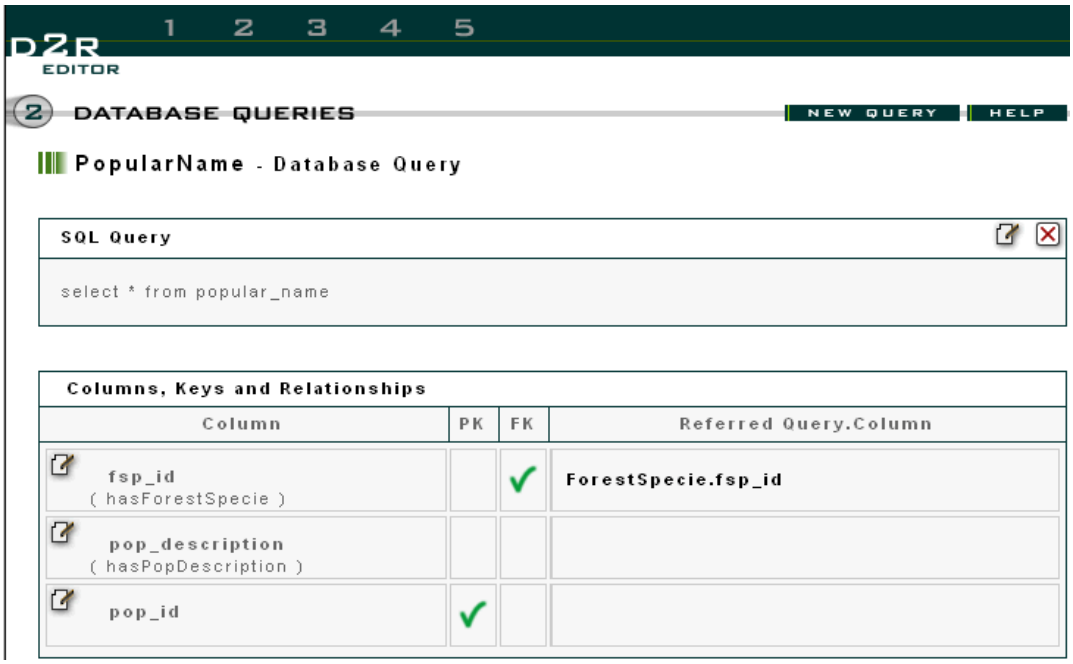

Figura 4-4 – Visualização das colunas resultantes de uma consulta SQL

Uma tabela correspondente indicada em um relacionamento de chave estrangeira é outra consulta SQL previamente registrada no sistema. Somente com todas as informações de relacionamentos entre as tabelas resultantes das consultas SQL, o *D2R Editor* chega a uma pseudo-modelagem do banco de dados real.

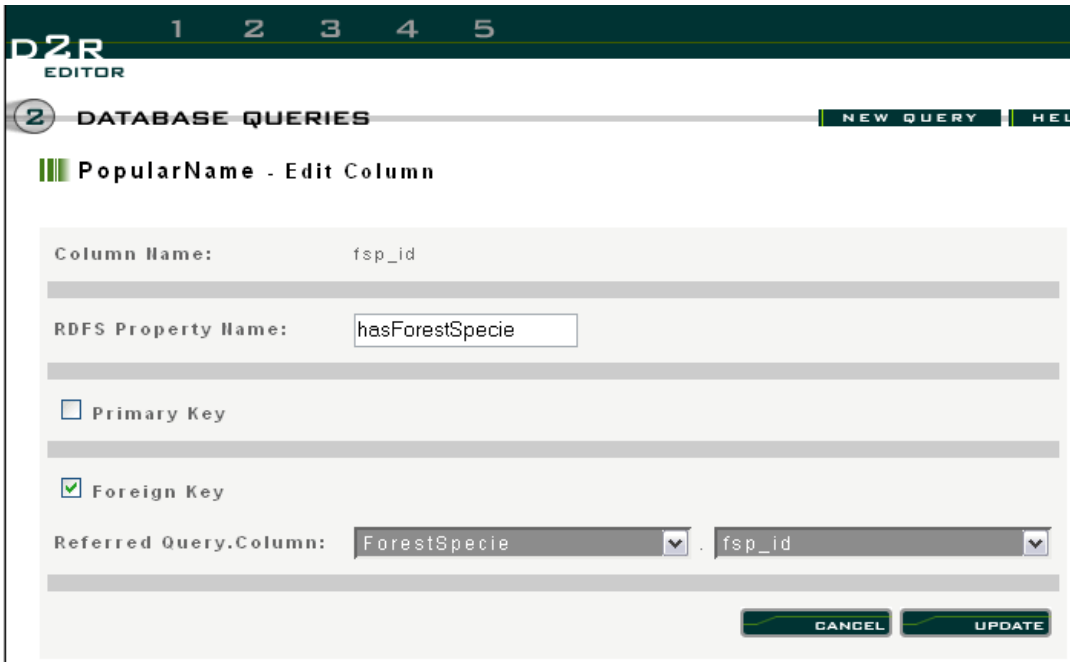

Figura 4-5 - Propriedades de uma coluna resultante

4.2.3 Passo 3 - Namespaces

No terceiro passo são cadastradas as referências URI de outras ontologias que se relacionam com os conceitos das informações que serão mapeadas automaticamente. Estes identificadores URI servem como base para busca de termos de ontologias no próximo passo, também estando presentes no *namespace* do vocabulário RDF gerado no quinto passo. A Figura 4-6 mostra a interface de cadastro de um URI.

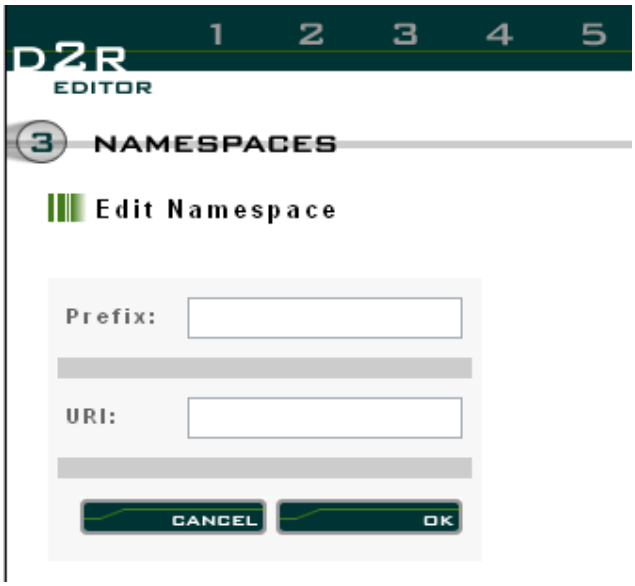

Figura 4-6 - Cadastro de informações de uma referência URI.

4.2.4 Passo 4 – RDF Schema

Na quarta etapa do processo, são apresentadas para o usuário as características básicas de cada classe RDF gerada pelo sistema. Elas são derivadas do pseudo-modelo relacional obtido no registro e configuração das consultas SQL da segunda etapa. A interface de apresentação das características de uma classe RDF está ilustrada na Figura 4-7.

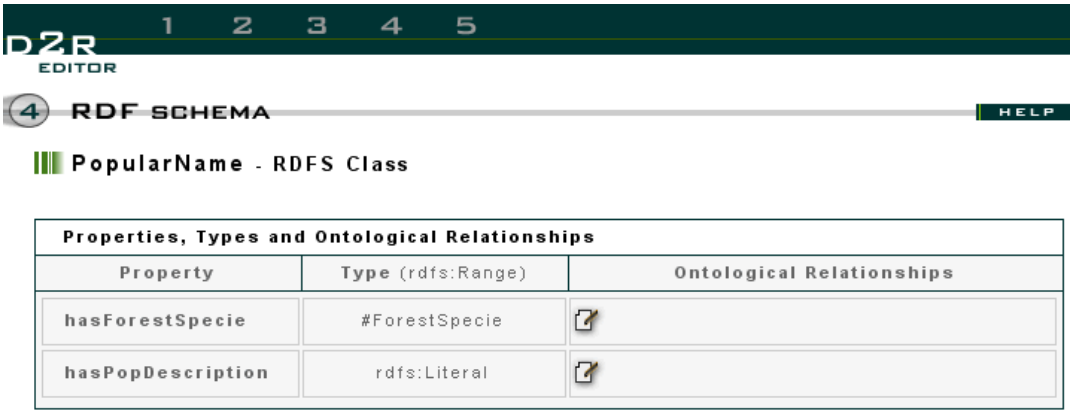

Figura 4-7 - Características de uma classe RDF resultante.

Cada classe apresentada nesta etapa é composta por uma ou mais propriedades derivadas das colunas resultantes de uma consulta SQL. Mais do que antecipar informações das classes RDF que serão criadas, o objetivo principal desta etapa do

processo é permitir que o usuário aponte relacionamentos ontológicos entre as propriedades das classes apresentadas e termos de outras ontologias, cujas referências URI foram registradas na terceira etapa. Isto é importante para que as informações migradas do banco de dados para instâncias RDF possam estar relacionadas com outros conceitos, viabilizando comparações com dados de outras fontes, que é uma das principais finalidades da *Web* Semântica.

No caso desta seção, uma propriedade é composta por um nome, um valor que determina o seu tipo (rdfs:Range), e por um ou mais relacionamentos ontológicos. Estes relacionamentos podem ser incluídos ou editados através da interface apresentada na Figura 4-8. Cada relacionamento é comporto por três valores:

- **Tipo de relacionamento**: determina qual o termo OWL que irá relacionar a propriedade indicada com o outro termo de outra ontologia.
- **Ontologia externa**: determina o prefixo previamente cadastrado na etapa três, o qual representa uma outra ontologia que pode estar em qualquer lugar da *Web*.
- **Termo externo relacionado**: determina especificamente o termo externo que está sendo relacionado na sentença.

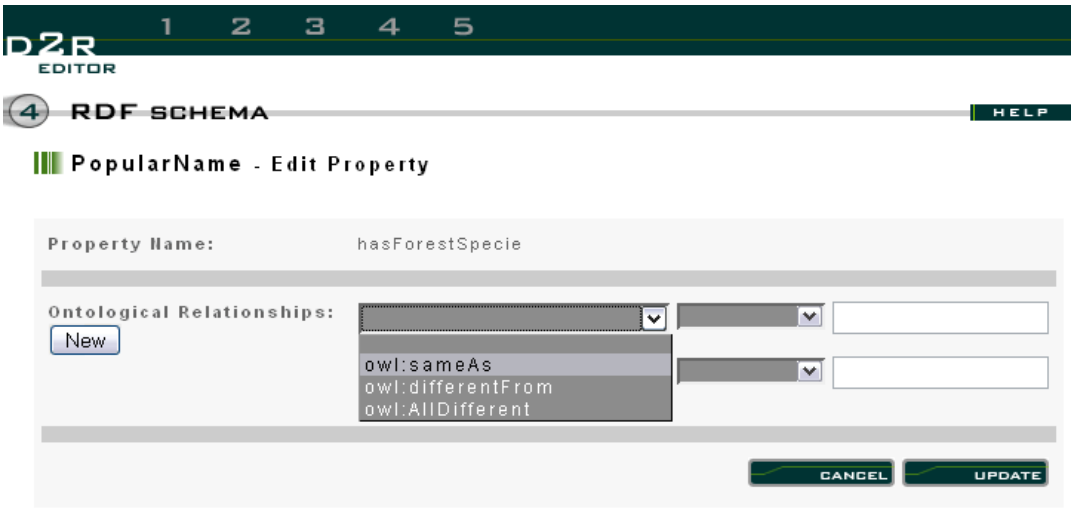

Figura 4-8 – Edição dos relacionamentos ontológicos de uma propriedade

4.2.5 Passo 5 - Geração do Mapa de Migração e do Esquema RDF

Realizadas todas as configurações necessárias, no quinto passo o usuário apenas ajusta os últimos parâmetros utilizados, e ordena a execução do processo automático de geração do mapa de migração e do esquema RDF, conforme é mostrado na Figura 4-9. Os parâmetros ajustados nesta fase final são:

- Prefixo do esquema RDF;
- URI do esquema RDF;
- Nome do arquivo RDF onde as instâncias RDF serão salvas pelo *D2R Processor*;
- Formato das instâncias RDF geradas pelo *D2R Processor*;
- Conteúdo adicional que poderá estar no início e no fim do arquivo gerado pelo *D2R Processor*.

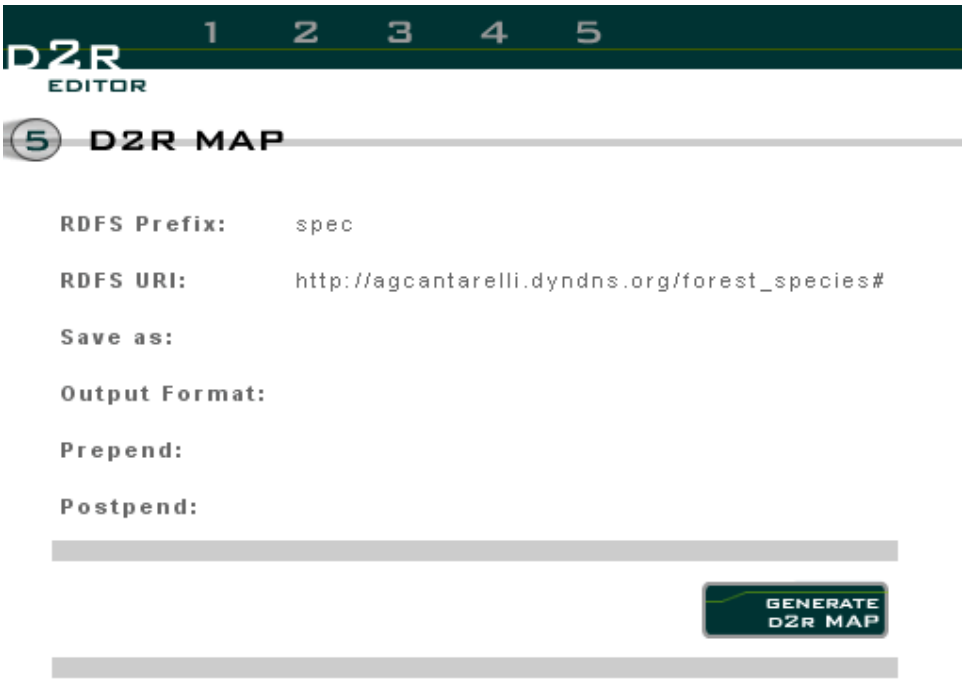

Figura 4-9 - Interface de geração do mapa de migração e do esquema RDF

Ao fim do processo, o sistema cria dois arquivos, um contento declarações RDF os quais definem o vocabulário dos termos do banco de dados em questão, e outro

contendo especificações XML as quais definem o mapa de migração *D2R Map*. O Quadro 4-1 apresenta exemplos resumidos de informações geradas nos dois tipos de arquivos.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
EXEMPLO 1 – CONTEÚDO DE UM MAPA DE MIGRAÇÃO
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
<?xml version="1.0"?>
<d2r:Map
xmlns:d2r="http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RMap/0.1#"
d2r:versionInfo="v1">
<d2r:ProcessorMessage d2r:outputFormat="RDF/XML-ABBREV"/>
<d2r:DBConnection d2r:odbcDSN="fruit_db" d2r:username="root" />
<d2r:Namespace d2r:prefix="fru"
d2r:namespace="http://agcantarelli.dyndns.org/fruits#" />
<d2r:Namespace d2r:prefix="rdf"
d2r:namespace="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
<d2r:ClassMap d2r:type="fru:Fruit"
d2r:sql="select FRU_ID, FRU_NOME from fruta" d2r:groupBy="FRU_ID"
d2r:uriPattern="fru:Fruit@@FRU_ID@@" >
<d2r:DatatypePropertyBridge d2r:property="fru:hasFruitName"
d2r:column="FRU_NOME" />
</d2r:ClassMap>
</d2r:Map>
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
EXEMPLO 2 – CONTEÚDO DE UM ESQUEMA RDF
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
<?xml version="1.0"?>
<rdf:RDF
xml:base="http://agcantarelli.dyndns.org/fruits#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#">
<rdfs:Class rdf:ID="Fruit"/>
<rdf:Property rdf:ID="hasFruitName">
<rdfs:domain rdf:resource="#Fruit"/>
<rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
</rdf:RDF>
```

Quadro 4-1 - Exemplo de conteúdos gerados pelo D2R Editor.

4.3 Integração com o D2R Processor

O objetivo final de um usuário administrador de banco de dados, ao utilizar o *D2R Editor* é, além de gerar um mapa de migração com seu respectivo esquema RDF, viabilizar a produção automática de anotações semânticas contendo informações de seu banco de dados.

Para atingir este objetivo, é necessário integrar o *site* com o *D2R Processor*. *A*ssim, quando uma requisição HTTP é feita, o *site*, além de gerar o conteúdo HTML, faz uma solicitação para o *D2R Processor* passando como parâmetro o caminho do mapa de migração, recebendo como resposta instâncias RDF com informações do banco de dados. A Figura 4-10 apresenta detalhes deste procedimento.

Para permitir a integração com o *D2R Processor*, o *site* deve fazer uma chamada para este gerador de conteúdo RDF, através de uma interface D2R. Esta interface pode ser implementada de várias maneiras, dependendo da linguagem de programação do *site*. Como o *D2R Processor* foi desenvolvido na linguagem Java, *sites* que utilizam esta linguagem somente precisam invocar o método do *D2R Processor* que a integração estará implementada. No entanto, quando as linguagens são diferentes, é necessário desenvolver um módulo de interface. Este módulo poderá chamar o *D2R Processor* através de uma rotina no sistema operacional, ou até mesmo com *Web Services* [BOO 04]. A versão atual do *D2R Editor* ainda não cria automaticamente esta interface de integração, então, isto deve ser programando manualmente pelo administrador do *site*.

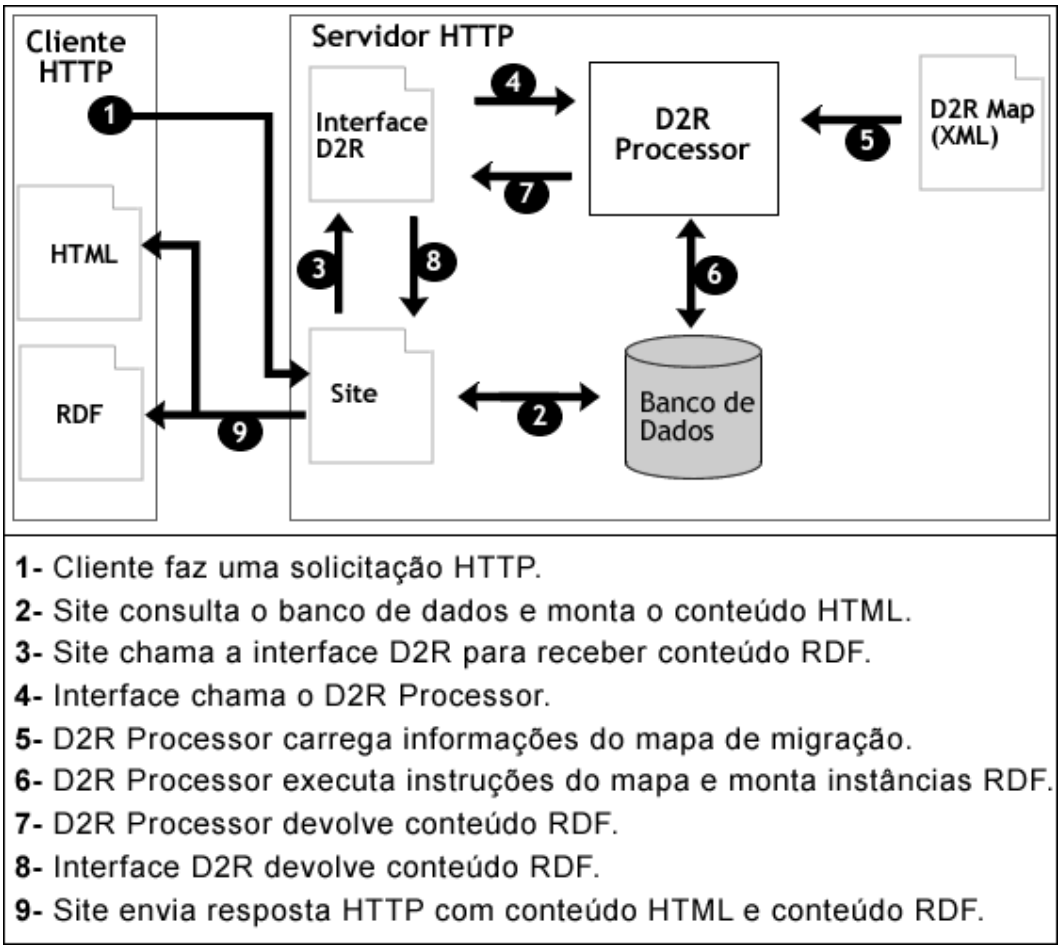

Figura 4-10 - Chamada do D2R Processor em um site dinâmico.

4.4 Implementação

O *D2R Editor* é uma aplicação baseada na *Web* e foi desenvolvido na linguagem *PHP* [PHP 04], pelos motivos de ter licença gratuita, funcionar em mais de um sistema operacional, e estar amplamente difundida. Para armazenar as informações sobre os bancos de dados "alvo" e gerar mapas de migração e esquemas RDF, o *D2R Editor* utiliza o banco de dados relacional *MySQL* [MYS 04], pelo fato de ser multi-plataforma, gratuito, rápido e muito utilizado principalmente em aplicações *Web*. A implementação de suas principais funções encontram-se no Anexo II.

4.4.1 Banco de Dados do Sistema

Os dados sobre conexão, consultas SQL, classes e propriedades RDF, gerenciados pelos usuários através da interface do sistema, são armazenados em um

banco de dados relacional. Estas informações servem como entrada para os métodos de geração do mapa de migração e do esquema RDF. A criação deste modelo foi baseada nas seguintes premissas:

- Uma base de dados alvo possui várias tabelas;
- Uma tabela de um determinado banco de dados possui várias colunas;
- Uma base de dados pode estar relacionada com várias referências URI;
- Uma coluna de uma determinada tabela pode ser oriunda de outra tabela (chave estrangeira);
- Uma coluna pode ter vários relacionamentos ontológicos.

A Figura 4-11 ilustra o modelo relacional da base de dados do sistema.

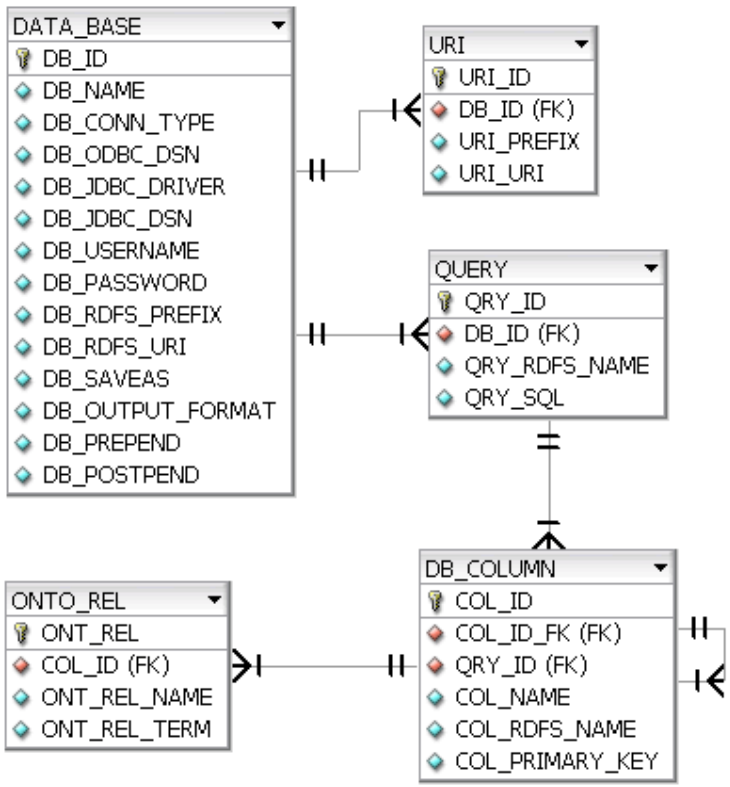

Figura 4-11 - Modelo relacional do banco de dados do D2R Editor.

4.4.2 Método para Geração de Esquema RDF

A geração de um esquema RDF no *D2R Editor* acontece de forma automatizada, onde o sistema recupera informações cadastradas pelo usuário e gera um conteúdo que é gravado em um arquivo. Este conteúdo deve, através do formato RDF/XML, realizar definições de classes e propriedades de forma a representar o pseudo-modelo relacional formado pelas tabelas resultantes das consultas SQL registradas no sistema. Para montar o conteúdo de um esquema RDF referente a um determinado banco de dados "alvo", o sistema precisa buscar as seguintes informações:

- Referências URI;
- Nomes de classes;
- Propriedades com seus respectivos relacionamentos.

As informações utilizadas na formação do *namespace* do vocabulário RDF encontram-se na tabela `URI`, e os valores dos prefixos e destas referências são originados dos campos `URI_PREFIX` e `URI_URI`, respectivamente. A coluna `DB_ID` é utilizada para filtrar os valores relacionados somente com um banco de dados "alvo" específico.

As classes do vocabulário RDF são obtidas através de uma consulta na tabela `QUERY`, de onde a coluna `QRY_RDFS_NAME` provê os valores dos nomes de cada classe. O Quadro 4-2 apresenta um exemplo de definições de *namespaces* e de uma classe RDFS, onde os valores destacados são provenientes do banco de dados do *D2R Editor*.

```
<rdf:RDF
xml:base="http://agcantarelli.dyndns.org/fruits#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
>
<rdfs:Class rdf:ID="Fruit"/>
```

Quadro 4-2 - Definições de *namespaces* e de uma classe RDF gerados automaticamente

As propriedades do vocabulário RDF são oriundas da tabela `COLUMN`, onde são filtradas informações relacionadas com as tabelas de um banco de dados "alvo" específico, através da coluna `QRY_ID`.

Dentre os resultados dos filtros, somente originam uma propriedade aqueles que, possuem o valor de `COL_ID_FK` diferente de "0", ou `COL_PRIMARY_KEY` igual a "0" pois,

para se transformar em uma propriedade RDFS, uma coluna de um banco de dados deve ser, ou uma chave estrangeira, ou não pode ser uma chave primária.

Uma propriedade RDF, além de seu nome, deve ter no mínimo dois valores de propriedades, um para `rdfs:range` e outro para `rdfs:domain`. O valor de `rdfs:range` é determinado pelo tipo de valor que esta propriedade poderá assumir, e o valor de `rdfs:domain` depende de qual classe esta propriedade pertence.

O método para encontrar o valor de `rdfs:domain` de uma propriedade, realiza as seguintes operações:

1. Com o valor de `QRY_ID` da coluna, busca o valor de `QRY_RDFS_NAME` nos registros da tabela `QUERY`;
2. O valor de `rdfs:domain` é determinado pelo valor de `QRY_RDFS_NAME`.

Quando o valor de `COL_ID_FK` for “0”, significa que a coluna não é chave primária nem chave estrangeira, portanto se transformará numa propriedade convencional, e o valor de `rdfs:range` será sempre `rdfs:Literal`. Quando o valor de `COL_ID_FK` for diferente de “0”, significa que a coluna é uma chave estrangeira e está relacionada com uma outra coluna de uma outra tabela. O valor de `COL_ID_FK` indica, então, o valor de `COL_ID` da coluna relacionada. Para estes casos de chaves estrangeiras, o valor de `rdfs:range` é determinado da seguinte maneira:

1. O sistema busca o valor de `QRY_ID` nos registros da tabela `COLUMN`, onde o valor de `COL_ID` é igual ao de `COL_ID_FK`;
2. Com o valor de `QRY_ID` da coluna que é chave estrangeira, o sistema busca o valor de `QRY_RDFS_NAME` nos registros da tabela `QUERY`, onde `QRY_ID` é igual ao valor de `QRY_ID` obtido na consulta anterior;
3. O valor de `rdfs:range` será determinado pelo valor de `QRY_RDFS_NAME`.

O sistema ainda faz uma busca nos relacionamentos ontológicos de cada coluna. Para isto a tabela `ONTO_REL` é consultada, utilizando como filtro o valor de `COL_ID`, para obter somente os valores da coluna especificada. Depois destas operações o sistema pode completar a montagem das propriedades, como mostra o exemplo do Quadro 4-3, onde os dados automáticos são destacados.

```
<?xml version="1.0"?>
<rdf:RDF
xml:base="http://agcantarelli.dyndns.org/fruits#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
>
<rdfs:Class rdf:ID="Fruit"/>
<rdf:Property rdf:ID="hasFruitName">
<rdfs:domain rdf:resource="#Fruit"/>
<rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
</rdf:RDF>
```

Quadro 4-3 - Exemplo de um esquama RDF gerado automaticamente.

4.4.3 Método para a Geração de D2R MAP

A criação de um arquivo cujo conteúdo é um mapa de migração *D2R Map* acontece baseada nos mesmos princípios da geração do esquema RDF, pois as consultas realizadas no banco de dados são muito semelhantes. O que difere estes processos é o tratamento que os resultados recebem, ou seja, as informações são utilizadas de forma diferente.

O *D2R Map* é composto por um elemento mestre (`d2r:Map`), o qual possui vários elementos filhos, que determinam suas características. O processo de geração de um mapa de migração é o conjunto de rotinas executadas para criar cada um destes elementos. A estrutura dos elementos da linguagem *D2R Map* está especificada com detalhes no Anexo I.

Para a formação dos elementos `d2r:ProcessorMessage`, `d2r:DBConnection`, `d2r:Prepend` e `d2r:Postpend`, a tabela do banco de dados `DATA_BASE` é consultada. Em seguida, um elemento `d2r:Namespace` é criado para cada registro da tabela “URI” relacionado com o banco de dados “alvo”. Este relacionamento é determinado pela coluna `DB_ID`.

Após a geração de cabeçalho do mapa de migração, são buscadas informações sobre os elementos `d2r:ClassMap`. Um mapa de classe (`d2r:ClassMap`) é gerado para

cada registro encontrado na tabela QUERY que esteja relacionado com o banco de dados “alvo”, e os valores dos suas propriedades são obtidos das seguintes formas:

- O valor da propriedade d2r:type é determinado pela junção do valor encontrado na coluna DB_NAME.DB_RDFS_PREFIX, mais o caractere “:”, e o valor da coluna QUERY.QRY_RDFS_NAME;
- A propriedade d2r:sql receberá o conteúdo da coluna QRY_SQL;
- O elemento dr2:ClassMap ainda contém duas propriedades, d2r:groupBy e d2r:uriPattern. Os seus valores são determinados através de uma busca realizada na tabela COLUMN, a qual retorna as colunas que fazem parte da chave primária da tabela resultante da consulta em questão.

Para determinar os elementos filhos de um mapa de classe, é realizada uma busca na tabela COLUMN, que retorna informações sobre as colunas da tabela em questão. Cada coluna resultante pode gerar um, dentre dois tipos de elementos possíveis, dependendo do valor de COL_ID_FK.

Quando o valor de COL_ID_FK for “0”, significa que a coluna não é chave primária nem chave estrangeira, portanto dará origem ao elemento d2r:DatatypePropertyBridge, e os valores de suas propriedades são como segue:

- A propriedade d2r:property é determinada pela concatenação do valor de DATA_BASE.DB_RDFS_PREFIX, mais o caractere “:”, seguido do valor COL_RDFS_NAME;
- A propriedade d2r:column é determinada por COL_NAME.

Quando o valor de COL_ID_FK for diferente de “0”, significa que a coluna é uma chave estrangeira e está relacionada com uma outra coluna de uma outra tabela. Neste caso o elemento gerado será ObjectPropertyBridge, contento as seguintes propriedades:

- A propriedade d2r:property é determinada pela concatenação do valor de DATA_BASE.DB_RDFS_PREFIX, mais o caractere “:”, seguido do valor COL_RDFS_NAME. A partir de então o sistema faz uma busca para descobrir qual a classe relacionada, e determinar o valor de d2r:referredClass;

- Em seguida busca o valor da propriedade relacionada, determinando valor de `d2r:referredGroupBy`. A busca toma por base o valor de `COL_ID_FK`, o qual indica o valor de "`COL_ID`" da coluna relacionada.

4.5 Exemplo de Uso – Rede Semente Sul

A Rede Semente Sul [RSS 03] é o resultado de um projeto liderado pela Universidade Federal de Santa Catarina, onde foi formada uma rede de troca de conhecimento entre instituições, entidades, empresas e pessoas ligadas à pesquisa ou comercialização de sementes. O objetivo do projeto foi criar, comunicar e aplicar procedimentos para auxiliar a conservação das espécies florestais nativas da mata atlântica do sul do Brasil. Para transmitir estas informações, foram desenvolvidos mecanismos de disseminação de dados técnicos e comerciais sobre sementes, para que os envolvidos pudessem realizar de forma correta as atividades de cultivo, manejo, venda e troca criando, assim, ambientes sustentáveis de produção.

Uma das principais ferramentas desenvolvidas na Rede Semente Sul é um sistema *Web* o qual facilita a troca de informações entre todas as partes interessadas. O sistema conta com um grande banco de dados que apresenta diversas funcionalidades:

- Descrição das informações gerais do projeto;
- Divulgação de eventos relacionados com a área;
- Publicação de artigos e notícias relevantes;
- Fórum virtual para discussões sobre os mais diversos temas;
- Gerência de informações sobre todos os envolvidos no projeto;
- Cadastros de informações detalhadas sobre as principais espécies florestais da mata atlântica, com fotos, distribuição, clima, ecologia, utilidades, informações técnicas de manejo, bibliografias, etc;
- Sistema para gerência de informações comerciais de sementes, onde os participantes da rede podem cadastrar seus lotes de sementes e trocar informações de compra, troca e venda entre si, criando um grande leilão interativo.

Com este sistema é possível compartilhar conhecimentos sobre espécies florestais entre muitos ramos da sociedade, fazendo com que pessoas dos mais diversos segmentos se mantenham informadas, o que facilita muito o processo de conservação. A ferramenta de comercialização, além de auxiliar o desenvolvimento econômico de produtores da região, permite que as espécies sejam distribuídas pelo país.

Embora o sistema da Rede Semente Sul apresente funcionalidades muito úteis para a troca de informações entre diversas entidades, este processo ainda é centralizado, ou seja, a informação fica dependente de um só ponto, o *site* da Rede Semente Sul. Com isto, muitos dados armazenados em bancos de dados de outras instituições participantes, os quais poderiam ser muito úteis, ficam isolados e não participam deste processo de disseminação de informações, pois não existe um mecanismo automatizado para realizar tal integração.

As tecnologias utilizadas na *Web* Semântica poderiam ser aplicadas no caso da Rede Semente Sul, pois, permitiriam que informações provenientes de instituições diferentes pudessem ser processadas de forma inteligente, e entregues aos usuários de forma transparente. Para viabilizar este processo de integração de informações em um ambiente da *Web* Semântica, antes da implementação de agentes inteligentes que processarão as informações, é necessária a preparação dos dados, ou seja, fazer com que os bancos de dados de todos envolvidos tornem-se aptos a publicar conteúdos ricos em uma semântica formal.

Na seção 4.5.1 é apresentada uma demonstração de uso do sistema *D2R Editor* para habilitar um banco de dados sobre espécies florestais a publicar automaticamente conteúdos descritos no formato RDF/XML, permitindo que sejam interpretados por agentes inteligentes. Para viabilizar a integração de informações de fontes diferentes, este processo deverá ser aplicado em cada banco de dados, entre todas as entidades participantes da Rede Semente Sul.

4.5.1 Geração de Anotações Semânticas sobre Espécies Florestais

Este exemplo de uso apresenta está baseado em um banco de dados de espécies florestais de uma instituição participante da Rede Semente Sul, como mostra a Figura 4-12. Tal banco de dados contém informações que devem ser publicadas e integradas,

como nomes populares, famílias das espécies, grupos ecológicos, tipos de frutos, utilidades e quantidades disponíveis para comercialização. Entretanto existem, também, dados que somente dizem respeito à instituição, tais como informações dos compradores dos lotes, datas e quantidades vendidas.

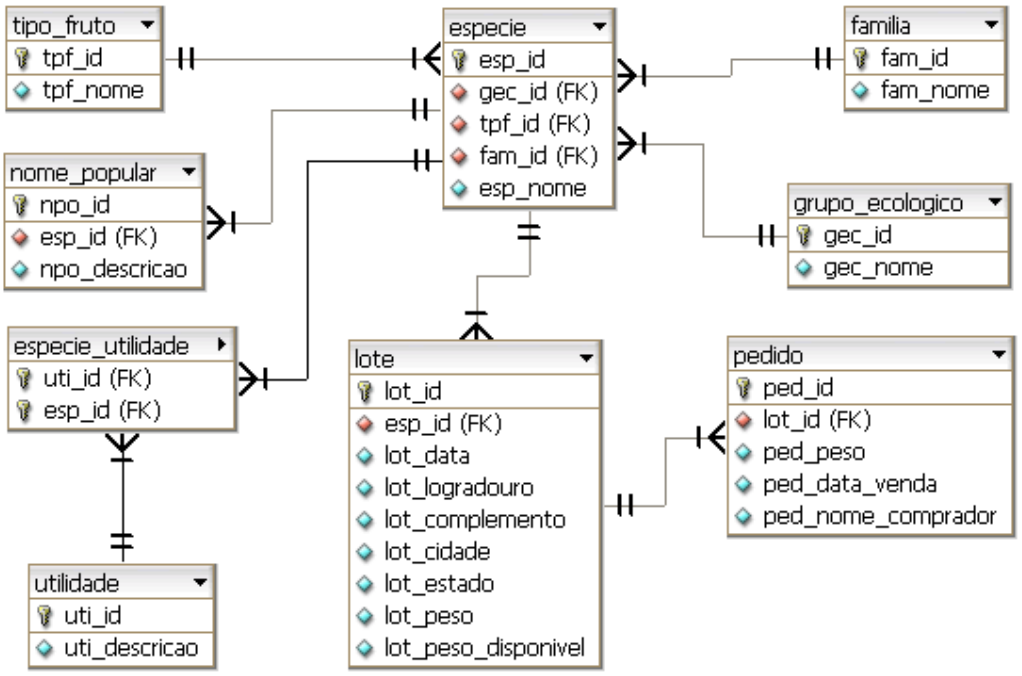

Figura 4-12 - Modelagem relacional do banco de dados de espécies florestais.

Para permitir que o *D2R Processor* publique declarações RDF sobre as informações de espécies florestais, é necessária a criação de um arquivo que serve como um mapa de migração, o qual define as consultas SQL que serão executadas e como será a estrutura do documento RDF criado. Adicionalmente deve ser criado um arquivo contendo o esquema RDF que define formalmente o vocabulário das declarações RDF geradas. Para facilitar a criação destes dois arquivos, deve ser utilizado o sistema *D2R Editor*.

No primeiro passo do *D2R Editor* são definidas informações de conexão com o banco de dados de espécies florestais e, no segundo passo, o usuário informa as consultas SQL que irão compor o mapa de migração. Esta tarefa deve ser realizada de tal forma que as consultas SQL selecionem apenas aquelas informações passíveis de publicação, escondendo os dados considerados sigilosos. Para atingir este objetivo, foram utilizadas quatro seleções SQL, conforme o Quadro 4-4.

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//====== Consulta Espécies Florestais
SELECT
   especie.esp_id,
   esp_nome nome_cientifico,
   sum(lot_peso_disponivel) peso_disponivel,
   tpf_nome tipo_fruto,
   fam_nome familia,
   gec_nome grupo_ecologico
FROM
   especie, lote, tipo_fruto, familia, grupo_ecologico
WHERE
   especie.esp_id = lote.esp_id
   and tipo_fruto.tpf_id = especie.tpf_id
   and familia.fam_id = especie.fam_id
   and grupo_ecologico.gec_id = especie.gec_id
GROUP BY
   especie.esp_id

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//====== Consulta Utilidades
SELECT * FROM utilidade

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//====== Consulta Relacionamento entre Espécies e suas Utilidades
SELECT * FROM especie_utilidade

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
====== Consulta Nomes Populares das Espécies Florestais
SELECT * FROM nome_popular
```

Quadro 4-4 - Consultas SQL registrados no D2R Editor

Depois de cadastrar as consultas SQL, e ainda no segundo passo, o usuário deve identificar quais campos resultantes de cada consulta fazem o papel de chave primária e chave estrangeira. Através destas informações o sistema irá formar uma pseudo-modelagem do banco de dados real, que servirá como base para a criação do Esquema RDF. A Figura 4-13 ilustra a pseudo-modelagem extraída das consultas SQL registradas neste caso de uso.

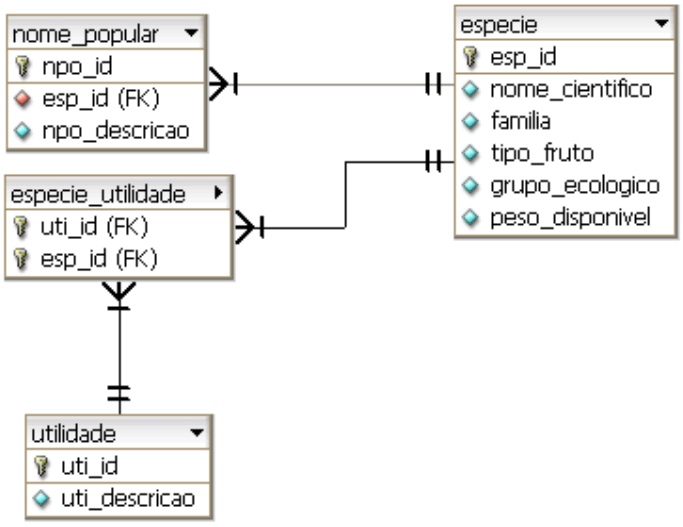

Figura 4-13 - Pseudo-modelagem do banco de dados de espécies florestais.

No terceiro passo podem ser cadastradas URI's que apontam para ontologias já definidas sobre espécies florestais, e no quarto passo, conforme a Figura 4-14, o sistema apresenta para usuário as classes e propriedades que foram derivadas a partir da pseudo-modelagem relacional, permitindo que sejam informados relacionamentos ontológicos entre termos descritos nas ontologias localizadas nas URI's registradas no terceiro passo. É possível notar que, nesta etapa, sempre existirá uma classe para cada consulta SQL cadastrada no segundo passo.

Figura 4-14 - Classes e propriedades de espécies florestais.

Com todas as informações relevantes registradas, o *D2R Editor* gera, no quinto passo, tanto o esquema RDF, quanto o arquivo que viabiliza o *D2R Processor* a criar instâncias RDF sobre espécies florestais. O conteúdo destes dois arquivos encontram-se nos Anexos III e IV, respectivamente. Um exemplo de instâncias RDF geradas automaticamente pelo *D2R Processor* é apresentado no Anexo V.

5 CONCLUSÕES

A *Web* Semântica é uma nova visão da *Web*, onde os conteúdos ganham descrições formais, permitindo a ação de softwares inteligentes que trabalhando em cooperação, podem integrar informações de diversas fontes, realizando tarefas que só poderiam ser feitas por humanos. Todavia, para viabilizar este processo, é necessário estender as informações disponíveis atualmente, adicionando descrições formais de seus significados, as quais permitirão tal processamento inteligente.

Uma das tarefas mais importantes para viabilizar o funcionamento da *Web* Semântica, é a criação de métodos os quais permitem que bancos de dados relacionais possam publicar seus conteúdos, enriquecidos com uma semântica formal. O trabalho de Chriz Bizer especifica uma linguagem que permite a criação de mapas de migração e, adicionalmente, apresenta um sistema que interpreta estes mapas, executa as instruções e gera instâncias RDF sobre o conteúdo do banco de dados.

O presente trabalho tem como produto o *D2R Editor*, uma ferramenta para a geração automática de mapas de migração e esquemas RDF. Além de complementar as funcionalidades do sistema apresentado por Chris Bizer, o D2R Editor facilita a aplicação do processo em um *site* dinâmico que esteja sendo preparado para fazer parte do ambiente da *Web* Semântica, uma vez que qualquer adminsitrador de banco de dados é capaz de operar o sistema.

5.1 Resultados Gerais

Foi desenvolvido um sistema chamado *D2R Editor*, capaz de gerar de mapas de migração e esquemas RDF sobre um determinado banco de dados relacional. Através desta ferramenta, administradores de bancos de dados podem registrar informações sobre modelos relacionais e obter como resultado, os respsctivos mapas de migração semântica e vocabulários RDF os quais descrevem formalmente os conceitos utilizados.

Através deste software, é possível que mais *sites* dinâmicos da *Internet* disponibilizaem suas informações estruturadas de forma com que agentes inteligentes possam “compreendê-las”, trazendo resultados mais eficientes para o usuário final.

5.2 Resultados Específicos

Foi desenvolvido um sistema *Web* com uma interface que viabiliza o registro de informações sobre a estrutura de um banco de dados. Através desta interface o usuário pode gerenciar informações sobre diversos bancos de dados separadamente, informando tabelas e relacionamentos. O sistema ainda permite que sejam registrados relacionamentos dos os termos do banco de dados com conceitos definidos em ontologias de outros locais da Web.

Com base nas informações sobre o banco de dados, o sistema gera um arquivo XML que funciona como um mapa de migração, utilizando a linguagem definida por Chris Bizer. Este mapa, além de indicar as consultas que devem ser executadas no banco de dados na hora da geração de instâncias RDF, aponta quais classes e propriedades RDF devem ser instanciadas de acordo com os resultados das consultas. Estas instruções são necessárias para que o *D2R Processor* gere declarações RDF automaticamente.

De acordo com as informações registradas pelo usuário, o *D2R Editor* cria um arquivo XML que funciona como um vocabulário RDF, o qual define os termos da estrutura relacional do banco de dados contendo, também, extensões que descrevem o relacionamento de tais termos os de outras ontologias. Este vocabulário é utilizado na interpretação do significado dos conteúdos que serão gerados automaticamente pelo *D2R Processor*.

O sistema foi testado junto ao banco de dados da Rede Semente Sul, onde foi verificada a geração de um mapa de migração e um esquema RDF que define alguns conceitos sobre espécies florestais. Como parte final dos testes, foi utilizado o sistema *D2R Processor*, que recebeu como entrada o mapa de migração gerado pelo *D2R Editor*, resultando em declarações RDF referentes às espécies cadastradas no banco de dados em questão.

5.3 Recomendações para Trabalhos Futuros

- Adicionar ao D2R Editor funcionalidades para a criação de vocabulários RDF que definam a estrutura de um banco de dados relacional em vários idiomas;

- Implementar um módulo para buscar ontologias espalhadas pela Web e relacionar automaticamente com os termos do banco de dados gerenciado pelo *D2R Editor*;
- Aperfeiçoar o processo de registro de consultas SQL que informam a estrutura do banco de dados relacional, apresentando uma interface gráfica que permita a navegação pelas tabelas e campos do banco de dados;
- Incorporar os métodos desenvolvidos por Chris Bizer dentro do *D2R Editor*. Assim, todo o processo de geração automática de anotações semânticas de informações de bancos de dados relacionais seria realizado em um único sistema.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[BEC 04] BECKETT, D.; MCBRIDE, B.; RDF/XML Syntax Specification. W3C Recommendation. http://www.w3.org/TR/rdf-syntax-grammar/. Acessado em 08/2003.

[BER 01] BERNERS-LEE, T.; HENDLER, J.; LASSILA, O.; The Semantic Web. Scientific American, New York, USA. Maio/2001. Acessado em 01/2003

[BER 98] BERNERS-LEE, T.; URI Generic Syntax. Standards Track – RFC 2396. http://www.isi.edu/in-notes/rfc2396.txt. Acessado em 11/2003.

[BIZ 03] BIZER, C.; D2R MAP – A Database to RDF Mapping Language. WWW2003. Maio/2003. Budapest, Hungary. http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rmap/www2003-D2R-Map.pdf. Acessado em 07/2003.

[BOO 04] BOOTH, D.; HAAS, H.; MCCABE, F.; Web Services Architeture. W3C Working Group Note 11 Febuary 2004. http://www.w3.org/TR/ws-arch/. Acessado em 05/2003.

[BRA 04] BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M.; MALER, E.; YERGEAU, F.; Extensible Markup Language (XML) 1.0. Terceira edição. World Wide Web Consortium. http://www.w3.org/TR/REC-xml/. Acessado em 07/2002.

[BRA 03] BRAY, T.; HOLLANDER, D.; LAYMAN, A.; Namespaces in XML. World Wide Web Consortium. http://www.w3.org/TR/REC-xml-names/. Acessado em 08/2003.

[BRI 03] BRICKLEY, D.; GUHA, R. V.; MCBRIDE, B.; RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. http://www.w3.org/TR/rdf-schema/. Acessado em 01/2003.

[CHA 99] CHANDRASEKARAN, B.; JOSEPHSON, J. R.; BENJAMINS, V. R.; What Are Ontologies, and Why Do We Need Them?. In IEEE Intelligent Systems. 1999. Magazine. pp 20 – 26.

[CON 01] CONNOLLY, D.; HARMELEN, F., V.; HORROCKS, I.; MCGUINNESS, D., L.; PATEL-SCHNEIDER, P., F.; STEIN, L., A; DAML+OIL

Reference Description. W3C Note. Dezembro/2001. http://www.w3.org/TR/daml+oil-reference. Acessado em 05/2003.

[DEC 00] DECKER, S.; MITRA, P.; MELNIK, S.; Framework for the Semantic Web: An RDF Tutorial. In IEEE Internet Computing. 2000. Magazine. pp 68 – 73.

[FAL 04] FALLSIDE, D., C.; WALMSLEY, P.; XML Schema Part 0: Primer. Segunda Edição. World Wide Web Consortium. http://www.w3.org/TR/xmlschema-0/. Acessado em 07/2003.

[GRA 04] GRAU, B. C.; A possible simplification of the semantic web architecture. International World Wide Web Conference; Proceedings of the 13th international conference on World Wide Web; SESSION: Semantic web foundations. pp 704 - 713. ACM Press. New York, USA. 2004.

[GRU 93] GRUBER, T. R.; A translation approach to portable ontology specifications. Knowledge Acquisition, 1993. 5:199-220.

[HAN 02] HANDSCHUH, S.; STAAB, S.; Authoring and annotation of web pages in CREAM. In International conference on World Wide Web, 11, 2002 , Honolulu, Hawaii, USA. Proceedings. New York, USA: ACM Press , 2002. p.p. 1100–1107.

[HAN 03] HANDSCHUH, S.; STAAB, S.; VOLZ, R.; On Deep Annotation. In Twelfth International Conference on World Wide Web. 2003. Proceedings. pp 431-438. ACM Press. New York-USA.

[JAV 04] JAVA TECNOLOGY. Web Site. http://java.sun.com/. Acessado em 09/2003.

[KLY 04] KLYNE, G.; CARROLL, J., J.; MCBRIDE, B.; Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. http://www.w3.org/TR/rdf-concepts/. Acessado em 10/2002.

[MAN 03] MANOLA, F.; MILLER, E.; MCBRIDE, B.; RDF Primer. W3C Recommendation. http://www.w3.org/TR/rdf-primer/. Acessado em 11/2002.

[MCB 01] MCBRIDE, B.; Jena: Implementing the RDF Model and Syntax Specification. Semantic Web Workshop, WWW2001. http://www-uk.hpl.hp.com/people/bwm/papers/20001221-paper/. Acessado em 03/2004.

[MYS 04] MYSQL - The world's most popular open source database. Web Site. http://www.mysql.com/. Acessado em 07/2003.

[NOY 01] NOY, N. F.; SINTEK, M.; DECKER, S.; CRUBÉZY, M.; FERGERSON, R. W.; MUSEN, M. A.; Creating Semantic Web Contents with Protégé-2000. In IEEE Intelligent Systems. 2001. Magazine. pp 70 – 71.

[PHP 04] PHP: Hypertext Preprocessor. Web Site. http://www.php.net/. Acessado em 02/2004.

[RAG 99] RAGGETT, D.; LE HORS, A.; JACOBS, I.; HTML 4.01 Specification. W3C Recommendation. W3C. 1999. http://www.w3.org/TR/html4/. Acessado em 01/2003

[RSS 03] Rede Semente Sul – Universidade Federal de Santa Catarina. Web Site. http://www.sementesul.ufsc.br. Acessado em 01/2003.

[SMI 03] SMITH, M. K.; WELTY, C.; MCGUINNESS, D. L.; OWL Web Ontology Language Guide. W3C Proposed Recommendation. W3C. 2003. http://www.w3.org/TR/owl-guide/. Acessado em 08/2003.

[STO 02] STOJANOVIC, L.; STOJANOVIC, N.; VOLZ, R.; Migrating data-intensive Web Sites into de Semantic Web. In ACM symposium on Applied computing, 2002, Madrid, Spain. Proceedings. New York, USA: ACM Press, 2002. pp 1100–1107.

[W3C 04] World Wide Web Consortium. Web Site. http://www.w3.org/. Acessado em 09/2002.

ANEXOS

I - Especificação da Estrutura da Linguagem D2R Map.

<table>
<tr><td colspan="2">Elemento <code>d2r:Map</code></td><td colspan="2">Descrição:
Elemento Raiz.</td></tr>
<tr><td rowspan="3">Atributos</td><td>Nome</td><td colspan="2">Descrição</td></tr>
<tr><td><code>xmlns:d2r</code></td><td colspan="2">URI do Esquema XML que define os termos da linguagem D2R.</td></tr>
<tr><td><code>d2r:versionInfo</code></td><td colspan="2">Informações sobre a versão do mapa.</td></tr>
<tr><td>Exemplo</td><td colspan="3"><code><d2r:Map
xmlns:d2r="http://www.wiwiss.fu-
berlin.de/suhl/bizer/D2RMap/0.1#"
d2r:versionInfo=”$Id:map.d2r,v1.0 2004/12/12
19:44:09 André$”>
<!— D2R Map -->
</d2r:Map></code></td></tr>
</table>

Elemento `d2r:DBConnection`	Descrição: Especifica os parâmetros de conexão ODBC ou JDBC.	
Atributos	Nome	Descrição
	`d2r:odbcDSN`	Nome da fonte de dados ODBC.
	`d2r:jdbcDriver`	Nome do *driver* JDBC.
	`d2r:jdbcDSN`	Nome da fonte de dados JDBC.
	`d2r:username`	Nome de usuário da fonte de dados.
	`d2r:password`	Senha da fonte de dados.
Exemplo	`<d2r:ODBConnection d2r:odbcDSN="semsul" />`	

Elemento `d2r:Namespace`		Descrição: Mapeia um prefixo para um N*amespace* específico, o qual pode ser usado em mapeamentos de classes e propriedades.	
	Atributos	Nome	Descrição
		`d2r:prefix`	Prefixo.
		`d2r:namespace`	URI que irá ser representada pelo prefixo.

	Exemplo	`<d2r:Namespace`
		`d2r:prefix="spec"`
		`d2r:namespace="http://www.sementesul.ufsc.br`
		`/species_schema#" />`	

Elemento `d2r:Prepend`	Descrição: Determina o conteúdo estático (HTML ou XML) que será adicionado no início do arquivo de saída.

Exemplo	`<d2r:Prepend>`
	`ABCD`
	`</d2r:Prepend>`

Elemento `d2r:ProcessorMessage`		Descrição: Define uma mensagem que será ao *D2R Processor*, especificando algumas opções para a geração do arquivo de saída.	
	Atributos	Nome	Descrição
		`d2r:saveAs`	Caminho onde o arquivo de saída deve ser salvo.
		`d2r:outputFormat`	Formato do arquivo (RDF/XML, RDF/XML-ABBREV, N-TRIPLE ou N3).
	Exemplo	`<d2r:ProcessorMassage` `saveAs="c:\test.rdf" outputformat="RDF/XML" />`	

Elemento `d2r:ClassMap`	Descrição: Usado para mapear o resultado de uma consulta SQL para uma classe ou um grupo de classes similares.	
Atributos	Nome	Descrição
	`rdf:id`	Identificador do ClassMap. Usado para referenciar o mapa nos atributos **d2r:referredClass**. Se um **d2r:type** é definido, também pode ser usado como identificador.
	`d2r:type`	URI de uma classe OWL ou de uma classe RDFS.
	`d2r:sql`	Consulta SQL para seleção de dados do banco de dados. Uso obrigatório.
	`d2r:groupBy`	Coluna ou lista de colunas usadas para agrupar as linhas do conjunto de registros resultantes. Todas as linhas com o mesmo valor nas colunas especificadas no d2r:groupBy irão formar a mesma instância. Uso obrigatório.
	`d2r:uriColunm`	Coluna do banco de dados que contém URIs de instâncias.
	`d2r:uriPattern`	Regra para a criação do URI que identificará cada instância.

Exemplo	`<d2r:ClassMap`
	`d2r:type="spec:PopularName"`
	`d2r:sql="select * from popularname"`
	`d2r:groupBy="pop_id, fsp_id"`
	`d2r:uriPattern="spec:PopularName@@pop_id@@-`
	`@@fsp_id@@"`
	`>`
	`<!– Property Bridges -->`
	`</d2r:ClassMap>`	

Elemento `d2r:DatatypePropertyBridge`	Descrição: Elemento utilizado dentro de um ClassMap. Define um relacionamento entre uma coluna do conjunto de registros resultante e uma propriedade literal das instâncias criadas.	
Atributos	Nome	Descrição
	`d2r:property`	Nome da propriedade. Uso obrigatório.
	`d2r:column`	Nome da coluna do conjunto de registros resultante.
	`d2r:pattern`	Padrão para criar o valor da propriedade.
	`d2r:value`	Cria uma propriedade adicional que fixa um valor para todas as instâncias da propriedade.
	`d2r:translate`	Identificador de um elemento **d2r:TranslationTable** usado para converter valores do banco de dados para valores de propriedades.
	`d2r:lang`	Identificador de linguagem.

Exemplo	`<d2r:DatatypePropertyBridge`
	`d2r:property="spec:hasNameDescription"`
	`d2r:column="pop_description"`
	`/>`	

Elemento `d2r:ObjectPropertyBridge`	Descrição: Elemento utilizado dentro de um ClassMap. Define um relacionamento entre uma coluna do conjunto de registros resultante e uma propriedade das instâncias criadas a qual aponta para outro objeto.	
Atributos	Nome	Descrição
	`d2r:property`	Nome da propriedade. Uso obrigatório.
	`d2r:column`	Nome da coluna do conjunto de registros resultante.
	`d2r:pattern`	Padrão para criar o valor da propriedade.
	`d2r:value`	Cria uma propriedade adicional que fixa um valor para todas as instâncias da propriedade.
	`d2r:translate`	Identificador de um elemento **d2r:TranslationTable** usado para converter valores do banco de dados para valores de propriedades.
	`d2r:referredClass`	Referencia a outro **d2r:ClassMap**. Aponta para as instâncias criadas dinamicamente do **d2r:ClassMap** indicado. Deve ser utilizado juntamente com **d2r:referredGroupBy**.
	`d2r:referredGroupBy`	Coluna ou lista de colunas que identificam as instâncias alvo de **d2r:referredClass**. Se mais de uma coluna é usada pra identificar a instância referenciada, a ordem das colunas deve estar de acordo com a ordem definida em **d2r:referredClass**.

Exemplo	`<d2r:ObjectPropertyBridge`
	`d2r:property="spec:hasForestSpecie"`
	`d2r:referredClass="spec:ForestSpecie"`
	`d2r:referredGroupBy="fsp_id"`
	`/>`	

Elemento `d2r:Postpend`	Descrição: Determina o conteúdo estático (HTML ou XML) que será adicionado no final do arquivo de saída.

Exemplo	`<d2r:Postpend>`
	`ABCD`
	`</d2r:Postpend>`

Elemento `d2r:TranslationTable`	Descrição: Pode ser usado para substituir valores de colunas antes destes serem usados como valores de propriedades. É um *container* para os elementos **d2r:Translation**	
Atributos	Nome	Descrição
	`d2r:id`	Identificador. Uso obrigatório.

Elemento `d2r:Translation`	Descrição: Define a conversão de uma chave para um valor.	
Atributos	Nome	Descrição
	`d2r:key`	Nome da chave. Uso obrigatório.
	`d2r:value`	Valor. Uso obrigatório.

Exemplo	<pre><d2r:TranslationTable d2r:id="Products2URI">
	<d2r:Translation
	d2r:key="12"
	d2r:value="http://www.a.com/Prod#1" />
	<d2r:Translation
	d2r: key ="1"
	d2r: value ="http://www.a.com/Prod#2" />
	<d2r:Translation
	d2r: key ="124"
	d2r: value ="http://www.abc.de/p#124" />
	</d2r:TranslationTable></pre>	

II - Principais Funções do D2R Editor

```
function get_rdfs(){
//MONTA O XML BÁSICO E CHAMA A FUNÇÃO QUE VAI PEGAR OS DADOS DAS CLASSES E
SUAS PROPRIEDADES
   $str = '<?xml version="1.0"?>
<rdf:RDF xml:base="http://agcantarelli.dyndns.org/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#" '.rdfs_get_str_uris().'>
'.rdfs_get_str_classes().'
</rdf:RDF>
   ';
   return($str);
}

function rdfs_get_str_uris(){
//RETORNA UMA STRING COM O XML DE TODAS AS URIS CADASTRADAS DE UM DETERMINADO
DB
   $rs = sql("
       select * from uri where DB_ID = ".$_SESSION["DB_ID"]
   );
   while($r = mysql_fetch_assoc($rs)){
       $str .= 'xmlns:'.$r['URI_PREFIX'].'="'.$r['URI_URI'].'" ';
   }
   return($str);
}

function rdfs_get_str_classes(){
//RETORNA UM XML COM TODAS AS CLASSES DE UM DETERMINADO DB
   //BUSCA NO BANCO TODAS AS QUERYS
   $rs = sql("
       select * from query where DB_ID = ".$_SESSION["DB_ID"]
   );
   //PRA CADA QUERY, INCREMENTA A STRING
   while($r = mysql_fetch_assoc($rs)){
       //COLOCA O ID DA CLASSE
       $str .= '<rdfs:Class rdf:ID="'.$r['QRY_RDFS_NAME'].'"/>';
       $str .= "\n";
       //BUSCA TODAS AS COLUNAS DESTA CLASSE QUE SÃO, OU CAMPOS CHAVE
ESTRANGEIRA OU NÃO SÃO CAMPOS CHAVE PRIMARIA
       $rs2 = sql("
              select * from db_column where QRY_ID = ".$r["QRY_ID"]." and
```

```
(COL_PRIMARY_KEY = 0 OR COL_ID_FK <> 0)
      ");
      //PRA CADA CAMPO
      while($r2 = mysql_fetch_assoc($rs2)){
            //INCREMENTA A STRING COM A DESCRIÇÃO DA PROPRIEDADE
            $str .= '
                  <rdf:Property rdf:ID="'.(empty($r2["COL_RDFS_NAME"]) ?
$r2["COL_NAME"] : $r2["COL_RDFS_NAME"]).'">
                        <rdfs:domain
rdf:resource="#'.$r['QRY_RDFS_NAME'].'"/>
                        <rdfs:range rdf:resource="'.($r2["COL_ID_FK"]=="0" ?
"rdfs:Literal" : column_get_rdfs_class_name($r2["COL_ID_FK"])).'"/>
                  </rdf:Property>
            ';
      }
  }
  return($str);
}


function column_get_rdfs_class_name($col_id){
  $rs = sql("select QRY_RDFS_NAME from db_column C, query Q where C.COL_ID =
".$col_id." and C.QRY_ID = Q.QRY_ID");
  $r = mysql_fetch_assoc($rs);
  return("#".$r["QRY_RDFS_NAME"]);
}


function get_d2rmap(){
  $db_info = get_data_base_info();
  $str = '<?xml version="1.0"?>
<d2r:Map xmlns:d2r="http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RMap/0.1#"
d2r:versionInfo="$Id: specDB.Map.d2r, v 1.0 2003/01/20 19:44:09 Chris Exp $">
<d2r:Namespace d2r:prefix="'.$db_info["DB_RDFS_PREFIX"].'"
d2r:namespace="'.$db_info["DB_RDFS_URI"].'"/>
<d2r:Namespace d2r:prefix="rdf" d2r:namespace="http://www.w3.org/1999/02/22-
rdf-syntax-ns#"/>
'.d2rmap_get_str_classmaps($db_info).'
</d2r:Map>';
  return($str);
}

function d2rmap_get_str_classmaps($db_info){
  $rs = sql("
```

```
	select * from query where DB_ID = ".$_SESSION["DB_ID"]
   );
   while($r = mysql_fetch_assoc($rs)){
	$rs2 = sql("
		select * from db_column where QRY_ID = ".$r["QRY_ID"]."
	");
	$str_groupBy = "";
	$str_uriPattern = "";
	$str_dtpb = ""; //DatatypePropertyBridge
	$str_opb = ""; //ObjectPropertyBridge
	while($r2 = mysql_fetch_assoc($rs2)){
		$d2r_property =
$db_info["DB_RDFS_PREFIX"].':'.(empty($r2["COL_RDFS_NAME"]) ? $r2["COL_NAME"]
: $r2["COL_RDFS_NAME"]);
		if($r2["COL_PRIMARY_KEY"] != "0"){
			$str_groupBy .= ",".$r2["COL_NAME"];
			$str_uriPattern .= "-@@".$r2["COL_NAME"]."@@";
		} elseif ($r2["COL_ID_FK"] == "0"){
			$str_dtpb .= '<d2r:DatatypePropertyBridge
d2r:property="'.$d2r_property.'" d2r:column="'.$r2["COL_NAME"].'" />';
			$str_dtpb .= "\n";
		}
		if ($r2["COL_ID_FK"] != "0"){
			$rs3 = sql("
				select Q.QRY_ID, Q.QRY_RDFS_NAME from db_column C,
query Q where C.COL_ID = ".$r2["COL_ID_FK"]." and C.QRY_ID = Q.QRY_ID
			");
			$r3 = mysql_fetch_assoc($rs3);
			$referedClass =
$db_info["DB_RDFS_PREFIX"].":".$r3["QRY_RDFS_NAME"];

			$rs3 = sql("
				select * from db_column where QRY_ID =
".$r3["QRY_ID"]."
			");
			$str_referredGroupBy = "";
			while($r3 = mysql_fetch_assoc($rs3)){
				if($r3["COL_PRIMARY_KEY"] != "0"){
					$str_referredGroupBy .= ",".$r3["COL_NAME"];
				}
			}
			$str_referredGroupBy =
substr($str_referredGroupBy,1,strlen($str_referredGroupBy));
			$str_opb .= '<d2r:ObjectPropertyBridge
```

```
d2r:property="'.$d2r_property.'" d2r:referredClass="'.$referedClass.'"
d2r:referredGroupBy="'.$str_referredGroupBy.'"/>';
                    $str_opb .= "\n";
            }

      }
      $str_groupBy = substr($str_groupBy,1,strlen($str_groupBy));
      $str_uriPattern = substr($str_uriPattern,1,strlen($str_uriPattern));
      $str .= '
      <d2r:ClassMap
d2r:type="'.$db_info["DB_RDFS_PREFIX"].':'.$r["QRY_RDFS_NAME"].'"
d2r:sql="'.$r["QRY_SQL"].'" d2r:groupBy="'.$str_groupBy.'"
d2r:uriPattern="'.$db_info["DB_RDFS_PREFIX"].':'.$r["QRY_RDFS_NAME"].$str_uriP
attern.'">
            '.$str_dtpb.'
            '.$str_opb.'
      </d2r:ClassMap>
      ';
  }
  return($str);
}
```

III - Esquema RDF de Espécies Florestais

```
<?xml version="1.0"?>
<rdf:RDF
  xml:base="http://agcantarelli.dyndns.org/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:sem="http://www.sementesul.ufsc.br/sementes.rdf#" >

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

  <rdf:Property rdf:ID="temEspecie">
    <rdfs:domain rdf:resource="#NomePopular"/>
    <rdfs:range rdf:resource="#Especie"/>
  </rdf:Property>

  <rdf:Property rdf:ID="temDescricao">
    <rdfs:domain rdf:resource="#NomePopular"/>
    <rdfs:range rdf:resource="rdfs:Literal"/>
  </rdf:Property>

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

  <rdf:Property rdf:ID="temUtilidade">
  <rdfs:domain rdf:resource="#UtilidadeEspecie"/>
  <rdfs:range rdf:resource="#Utilidade"/>
  </rdf:Property>

  <rdf:Property rdf:ID="temDetalhes">
  <rdfs:domain rdf:resource="#UtilidadeEspecie"/>
  <rdfs:range rdf:resource="rdfs:Literal"/>
  </rdf:Property>

  <rdf:Property rdf:ID="temEspecie">
    <rdfs:domain rdf:resource="#UtilidadeEspecie"/>
    <rdfs:range rdf:resource="#Especie"/>
  </rdf:Property>

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

  <rdf:Property rdf:ID="temDescricao">
    <rdfs:domain rdf:resource="#Utilidade"/>
```

```
      <rdfs:range rdf:resource="rdfs:Literal"/>
    </rdf:Property>

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

    <rdf:Property rdf:ID="temGrupoEcologico">
      <rdfs:domain rdf:resource="#Especie"/>
      <rdfs:range rdf:resource="rdfs:Literal"/>
    </rdf:Property>

    <rdf:Property rdf:ID="temFamilia">
      <rdfs:domain rdf:resource="#Especie"/>
      <rdfs:range rdf:resource="rdfs:Literal"/>
    </rdf:Property>

    <rdf:Property rdf:ID="temTipoFruto">
      <rdfs:domain rdf:resource="#Especie"/>
      <rdfs:range rdf:resource="rdfs:Literal"/>
    </rdf:Property>

    <rdf:Property rdf:ID="temPesoDisponivel">
      <rdfs:domain rdf:resource="#Especie"/>
      <rdfs:range rdf:resource="rdfs:Literal"/>
    </rdf:Property>

    <rdf:Property rdf:ID="temNomeCientifico">
      <rdfs:domain rdf:resource="#Especie"/>
      <rdfs:range rdf:resource="rdfs:Literal"/>
    </rdf:Property>

  </rdf:RDF>
```

IV - Mapa de Migração de Espécies Florestais

```
<?xml version="1.0"?>
<d2r:Map
  xmlns:d2r="http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RMap/0.1#"
  d2r:versionInfo="$Id: spec.Map.d2r, v 1.0 2004/10/20 19:44:09 Cantarelli
$" >

  <d2r:ProcessorMessage d2r:outputFormat="RDF/XML-ABBREV"/>

    <d2r:DBConnection d2r:odbcDSN="forest_species" />

  <d2r:Namespace
    d2r:prefix="spec"
    d2r:namespace="http://agcantarelli.dyndns.org/forest_species#"
  />

  <d2r:Namespace
    d2r:prefix="rdf"
    d2r:namespace="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  />

  <d2r:ClassMap
    d2r:type="spec:NomePopular"
    d2r:sql="select * from nome_popular"
    d2r:groupBy="npo_id"
    d2r:uriPattern="spec:NomePopular@@npo_id@@" >

    <d2r:DatatypePropertyBridge
          d2r:property="spec:temDescricao"
          d2r:column="npo_descricao"
    />

    <d2r:ObjectPropertyBridge
          d2r:property="spec:temEspecie"
          d2r:referredClass="spec:Especie"
          d2r:referredGroupBy="esp_id"
    />
  </d2r:ClassMap>

  <d2r:ClassMap
    d2r:type="spec:UtilidadeEspecie"
    d2r:sql="select * from especie_utilidade"
```

```
      d2r:groupBy="uti_id,esp_id"
      d2r:uriPattern="spec:UtilidadeEspecie@@uti_id@@-@@esp_id@@">

      <d2r:DatatypePropertyBridge
            d2r:property="spec:temDetalhes"
            d2r:column="eut_descricao"
      />

      <d2r:ObjectPropertyBridge
            d2r:property="spec:temUtilidade"
            d2r:referredClass="spec:Utilidade"
            d2r:referredGroupBy="uti_id"
      />

      <d2r:ObjectPropertyBridge
            d2r:property="spec:temEspecie"
            d2r:referredClass="spec:Especie"
            d2r:referredGroupBy="esp_id"
      />
    </d2r:ClassMap>

    <d2r:ClassMap
      d2r:type="spec:Utilidade"
      d2r:sql="select * from utilidade"
      d2r:groupBy="uti_id"
      d2r:uriPattern="spec:Utilidade@@uti_id@@">

      <d2r:DatatypePropertyBridge
            d2r:property="spec:temDescricao"
            d2r:column="uti_descricao"
      />
    </d2r:ClassMap>

    <d2r:ClassMap
      d2r:type="spec:Especie"
      d2r:sql="
            SELECT
                  especie.esp_id,
                  esp_nome nome_cientifico,
                  sum(lot_peso_disponivel) peso_disponivel,
                  tpf_nome tipo_fruto,
                  fam_nome familia,
                  gec_nome grupo_ecologico
            FROM
```

```
                especie, lote, tipo_fruto, familia, grupo_ecologico
        WHERE
                especie.esp_id = lote.esp_id
                and tipo_fruto.tpf_id = especie.tpf_id
                and familia.fam_id = especie.fam_id
                and grupo_ecologico.gec_id = especie.gec_id
        group by
                especie.esp_id"
    d2r:groupBy="esp_id"
    d2r:uriPattern="spec:Especie@@esp_id@@">

    <d2r:DatatypePropertyBridge
        d2r:property="spec:temGrupoEcologico"
        d2r:column="grupo_ecologico"
    />

    <d2r:DatatypePropertyBridge
        d2r:property="spec:temFamilia"
        d2r:column="familia"
    />

    <d2r:DatatypePropertyBridge
        d2r:property="spec:temTipoFruto"
        d2r:column="tipo_fruto"
    />

    <d2r:DatatypePropertyBridge
        d2r:property="spec:temPesoDisponivel"
        d2r:column="peso_disponivel"
    />

    <d2r:DatatypePropertyBridge
        d2r:property="spec:temNomeCientifico"
        d2r:column="nome_cientifico"
    />
  </d2r:ClassMap>
</d2r:Map>
```

V - Exemplo de Instâncias RDF sobre Espécies Florestais

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF
  xml:base="http://agcantarelli.dyndns.org/#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:spec="http://agcantarelli.dyndns.org/#" >

  <spec:NomePopular rdf:ID="http://agcantarelli.dyndns.org/#NomePopular1">
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie1"/>
    <spec:temDescricao>goiaba-serrana</spec:temDescricao>
  </spec:NomePopular>
  <spec:NomePopular rdf:ID="http://agcantarelli.dyndns.org/#NomePopular2">
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie1"/>
    <spec:temDescricao>araçá-do-rio-grande</spec:temDescricao>
  </spec:NomePopular>
  <spec:NomePopular rdf:ID="http://agcantarelli.dyndns.org/#NomePopular3">
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie2"/>
    <spec:temDescricao>canela-sêbo</spec:temDescricao>
  </spec:NomePopular>
  <spec:NomePopular rdf:ID="http://agcantarelli.dyndns.org/#NomePopular4">
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie2"/>
    <spec:temDescricao>canela-anhoíba</spec:temDescricao>
  </spec:NomePopular>
  <spec:NomePopular rdf:ID="http://agcantarelli.dyndns.org/#NomePopular5">
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie3"/>
    <spec:temDescricao>tanheiro</spec:temDescricao>
  </spec:NomePopular>
  <spec:NomePopular rdf:ID="http://agcantarelli.dyndns.org/#NomePopular6">
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org#Especie3"/>
    <spec:temDescricao>boleiro</spec:temDescricao>
  </spec:NomePopular>
  <spec:NomePopular rdf:ID="http://agcantarelli.dyndns.org/#NomePopular7">
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie3"/>
    <spec:temDescricao>caixeta</spec:temDescricao>
  </spec:NomePopular>
  <spec:NomePopular rdf:ID="http://agcantarelli.dyndns.org/#NomePopular8">
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie3"/>
    <spec:temDescricao>taneiro</spec:temDescricao>
  </spec:NomePopular>

  <spec:UtilidadeEspecie
rdf:ID="http://agcantarelli.dyndns.org/#UtilidadeEspecie1-1">
```

```
    <spec:temDetalhes>os pássaros se alimentam dos frutos e das pétalas
carnosas das flores</spec:temDetalhes>
    <spec:temUtilidade
rdf:about="http://agcantarelli.dyndns.org/#Utilidade1"/>
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie1"/>
  </spec:UtilidadeEspecie>
  <spec:UtilidadeEspecie
rdf:ID="http://agcantarelli.dyndns.org/#UtilidadeEspecie2-1">
    <spec:temDetalhes></spec:temDetalhes>
    <spec:temUtilidade
rdf:about="http://agcantarelli.dyndns.org/#Utilidade2"/>
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie1"/>
  </spec:UtilidadeEspecie>
  <spec:UtilidadeEspecie
rdf:ID="http://agcantarelli.dyndns.org/#UtilidadeEspecie1-2">
    <spec:temDetalhes>os frutos são provavelmente consumidos por
pássaros</spec:temDetalhes>
    <spec:temUtilidade
rdf:about="http://agcantarelli.dyndns.org/#Utilidade1"/>
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie2"/>
  </spec:UtilidadeEspecie>
  <spec:UtilidadeEspecie
rdf:ID="http://agcantarelli.dyndns.org/#UtilidadeEspecie3-2">
    <spec:temDetalhes>vigas</spec:temDetalhes>
    <spec:temUtilidade
rdf:about="http://agcantarelli.dyndns.org/#Utilidade3"/>
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie2"/>
  </spec:UtilidadeEspecie>
  <spec:UtilidadeEspecie
rdf:ID="http://agcantarelli.dyndns.org/#UtilidadeEspecie1-3">
    <spec:temDetalhes>folhas e frutos servem de alimento para macacos e
avifauna</spec:temDetalhes>
    <spec:temUtilidade
rdf:about="http://agcantarelli.dyndns.org/#Utilidade1"/>
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie3"/>
  </spec:UtilidadeEspecie>
  <spec:UtilidadeEspecie
rdf:ID="http://agcantarelli.dyndns.org/#UtilidadeEspecie4-3">
    <spec:temDetalhes>saponina e alcalóides são extraídos da
casca</spec:temDetalhes>
    <spec:temUtilidade
rdf:about="http://agcantarelli.dyndns.org/#Utilidade4"/>
    <spec:temEspecie rdf:about="http://agcantarelli.dyndns.org/#Especie3"/>
  </spec:UtilidadeEspecie>
```

```
  <spec:Utilidade rdf:ID="http://agcantarelli.dyndns.org/#Utilidade1">
    <spec:temDescricao>Alimentação da Fauna Silvestre</spec:temDescricao>
  </spec:Utilidade>
  <spec:Utilidade rdf:ID="http://agcantarelli.dyndns.org/#Utilidade2">
    <spec:temDescricao>Carvão</spec:temDescricao>
  </spec:Utilidade>
  <spec:Utilidade rdf:ID="http://agcantarelli.dyndns.org/#Utilidade3">
    <spec:temDescricao>Construção Civil</spec:temDescricao>
  </spec:Utilidade>
  <spec:Utilidade rdf:ID="http://agcantarelli.dyndns.org/#Utilidade4">
    <spec:temDescricao>Produção de constituintes
químicos</spec:temDescricao>
  </spec:Utilidade>

  <spec:Especie rdf:ID="http://agcantarelli.dyndns.org/#Especie1">
    <spec:temGrupoEcologico>Pioneira</spec:temGrupoEcologico>
    <spec:temFamilia>Myrtaceae</spec:temFamilia>
    <spec:temTipoFruto>Carnoso</spec:temTipoFruto>
    <spec:temPesoDisponivel>3.540</spec:temPesoDisponivel>
    <spec:temNomeCientifico>Acca sellowiana</spec:temNomeCientifico>
  </spec:Especie>
  <spec:Especie rdf:ID="http://agcantarelli.dyndns.org/#Especie2">
    <spec:temGrupoEcologico>Pioneira</spec:temGrupoEcologico>
    <spec:temFamilia>Lauraceae</spec:temFamilia>
    <spec:temTipoFruto>Carnoso</spec:temTipoFruto>
    <spec:temPesoDisponivel>18.500</spec:temPesoDisponivel>
    <spec:temNomeCientifico>Aiouea saligna</spec:temNomeCientifico>
  </spec:Especie>
  <spec:Especie rdf:ID="http://agcantarelli.dyndns.org/#Especie3">
    <spec:temGrupoEcologico>Secundária Inicial</spec:temGrupoEcologico>
    <spec:temFamilia>Euphorbiaceae</spec:temFamilia>
    <spec:temTipoFruto>Seco Deiscente</spec:temTipoFruto>
    <spec:temPesoDisponivel>12.610</spec:temPesoDisponivel>
    <spec:temNomeCientifico>Alchornea triplinervia</spec:temNomeCientifico>
  </spec:Especie>
</rdf:RDF>
```