***Resource Description Framework* (RDF)**

Júnio César de Lima *Cedric Luiz de Carvalho*

Technical Report - RT-INF_003-05 - Relatório Técnico
June - 2005 - Junho

Instituto de Informática
Universidade Federal de Goiás
www.inf.ufg.br

Resource Description Framework (RDF)

Júnio César de Lima [*] junio@inf.ufg.br

Cedric L. de Carvalho [†] cedric@inf.ufg.br

Abstract. *The Resource Description Framework (RDF) is a language for representing information about resources in the Web. It is intended for situations in which information needs to be processed by applications, rather than being only displayed to people. RDF is based on the assumptions that resources are described in terms of statements and they have properties which have values. It represents statements as a graph of nodes and arcs. Another form to model the statements is through triples. In triples notation, each statement is written as a subject, predicate, and object, in that order. To represent RDF statements in a machine-processable way, RDF provides an XML syntax, called RDF/XML. Besides these characteristics, RDF allows the definiton of vocabularies to be used in the statements, called RDF Schema. All these features of RDF are discussed and illustrated in this text.*

Keywords: RDF, RDF *Schema*, XML, metadata and Semantic Web.

Resumo. *O* Resource Description Framework *(RDF) é uma linguagem de propósito geral para representação de recursos na* Web*. Ele foi projetado para situações onde as informações necessitam ser processadas por aplicações, em lugar de somente serem visualizadas por pessoas. O RDF é baseado na idéia de que os recursos são descritos através de declarações e possuem propriedades que têm valores. Ele modela as declarações como* ***nós*** *e* ***arcos*** *em um grafo. Uma outra forma para modelar as declarações em RDF é através das chamadas "triplas RDF". Uma tripla é dividida em três partes: sujeito, predicado e objeto. Para representar as declarações de forma que possam ser mais facilmente processadas por máquinas, RDF usa a* Extensible Markup Language *(XML). O RDF define uma linguagem de marcação XML específica, chamada de RDF/XML. Além destas características, RDF permite definir um vocabulário para ser usado nas declarações. Este vocabulário é chamado de RDF Schema. Ele é especificado como um conjunto de classes, propriedades e restrições entre seus relacionamentos. Todas estas características do RDF são discutidas e exemplificadas neste texto.*

Palavras-Chave: RDF, RDF *Schema*, XML, metadados e *Web* Semântica.

[*]Mestrando em Ciência da Computação - GEApIS/INF/UFG

[†]Orientador - GEApIS/INF/UFG

1 Introdução

O *Resource Description Framework* (RDF) constitui-se em uma arquitetura genérica de metadados que permite representar informações sobre recursos na *World Wide Web* (WWW ou *Web*), tais como título, autor e data de atualização de uma página *Web*, por exemplo. Além disso, RDF também pode ser usado para representar informações sobre coisas que podem ser identificadas na *Web*, mesmo que elas não possam ser recuperadas, como informações sobre itens acessíveis de um mercado *on-line* (por exemplo: preço e marca de um produto). As principais características do RDF são:

- É proposto para situações onde as informações necessitam ser processadas por aplicações, em lugar de somente serem visualizadas por pessoas.
- Provê uma estrutura comum para expressar informações que podem ser trocadas entre diferentes aplicações sem perda de significado [9].
- Baseia-se no princípio de identificação de objetos usando identificadores *Web*, também chamados de URIs (discutidos na Seção 2.1), e na descrição de recursos em termos de propriedades e valores de propriedade. Isto capacita o RDF a representar declarações simples sobre recursos como um grafo de nós e arcos, representando os recursos, suas propriedades e valores [9].

Além disso, o RDF também provê uma sintaxe baseada em XML (chamada de RDF/XML) para registrar e intercambiar estes grafos. Esta sintaxe é processável por máquina e, usando-se URIs, pode-se ligar pedaços de informações através da *Web*. URIs em RDF podem se referir a qualquer coisa identificável, incluindo objetos que não podem ser recuperdos diretamente na *Web*, como por exemplo a pessoa de `Mário Sabino`.

O restante deste texto é organizado como se segue: na Seção 2, são mostrados os conceitos básicos, o modelo RDF e a representação de declarações RDF usando grafos e triplas. Na Seção 3, são discutidas as principais características da sintaxe RDF/XML, entre elas a criação de grupos e coleções de recursos e reificações. Na Seção 4, se discute os "Esquemas RDF" que estendem o modelo RDF para criação de vocabulários. Finalmente, na Seção 5, são apresentadas as considerações finais.

2 Declarações sobre recursos

O RDF foi projetado para fornecer uma maneira simples de fazer declarações sobre recursos na *Web*. Todas as coisas em RDF são chamadas de **recursos**, mas nenhuma suposição é feita sobre a sua natureza, ou seja, os recursos podem ser: páginas *Web*, pessoas ou qualquer outra coisa. Um **recurso** é tratado como sinônimo de entidade, isto é, como um termo genérico para qualquer coisa em um determinado domínio [7]. Nas Subseções 2.1 a 2.4, a seguir, se descreve como fazer declarações sobre recursos em RDF.

2.1 Conceitos básicos

O RDF é baseado na idéia de que as coisas que estão sendo descritas possuem propriedades que têm valores e que recursos podem ser descritos através de declarações [9]. Ele usa uma terminologia particular para realizar a descrição das partes de uma declaração. Especificamente, a parte que identifica o objeto da declaração (uma página *Web*, por exemplo) é chamada

de **sujeito** (recurso). A parte que identifica uma propriedade ou uma característica (o criador de uma página *Web*, por exemplo) de um recurso é chamada de **predicado** (propriedade), e a parte que identifica o valor de uma propriedade é chamada de **objeto** (valor de propriedade). Por exemplo, a declaração em português: " **`http://www.exemplo.org/index.html`** tem um **criador** cujo valor é **Mário Sabino**" poderia ser representada por uma declaração RDF como:

- O **sujeito** é a URL `http://www.exemplo.org/index.html`
- O **predicado** é a palavra criador
- O **objeto** é Mário Sabino

Enquanto uma linguagem natural, como o português, é adequada para a comunicação entre seres humanos, RDF é adequado para expressar descrições (declarações) a respeito de recursos, de forma a facilitar o processamento automatizado por máquinas. Segundo Manola e Miller [9], este processamento automatizado requer:

- Um sistema automatizado que consiga identificar o sujeito, o predicado e/ou o objeto em uma declaração sem ambigüidade;
- Uma linguagem para representar estas descrições que facilite o intercâmbio de informações entre máquinas.

Felizmente, estes dois requisitos são atendidos pela arquitetura *Web* existente.

Como foi discutido na Seção 1, a *Web* provê uma forma de identificação, o *Uniform Resource Locator* (URL). Um URL é uma seqüência de caracteres que identifica a localização de uma página *Web*. Contudo, também é importanteque se possa identificar muitas coisas que, diferentemente de páginas *Web*, não possuem localização na rede.

A *Web* Semântica [2] provê uma forma mais geral de identificação, chamado de *Uniform Resource Identifier* (URI). Diferentes pessoas e organizações podem, independentemente, criá-los e usá-los para identificar objetos [9]. Os URIs podem ser criados para referir-se a qualquer coisa que precise ser referenciada em uma declaração, não somente a recursos com endereços na *Web*.

Por causa da sua generalidade, RDF usa URIs como base do seu mecanismo de identificação de sujeitos, predicados e objetos em declarações. Para ser mais preciso, RDF usa referências URIs. Uma referência URI é um URI juntamente com um identificador de fragmento opcional no final. Por exemplo, a referência URI `http://www.exemplo.org/index.html#capitulo1` consiste do URI `http://www.exemplo.org/index.html` e do identificador de fragmento `capitulo1`, separados pelo caracter "#". Além disso, referências URIs em RDF podem conter caracteres *Unicode* [8], permitindo que várias linguagens possam ser usadas em referências URIs.

Para representar as declarações RDF de forma que possam ser mais facilmente processadas por máquinas, se usa a *Extensible Markup Language* (XML) [4]. Na representação de informações RDF e para o intercâmbio entre máquinas, pode ser usada uma linguagem de marcação XML específica, chamada de RDF/XML. Na Subseção 2.2, a seguir, se discute como é possível fazer declarações sobre objetos usando-se URIs.

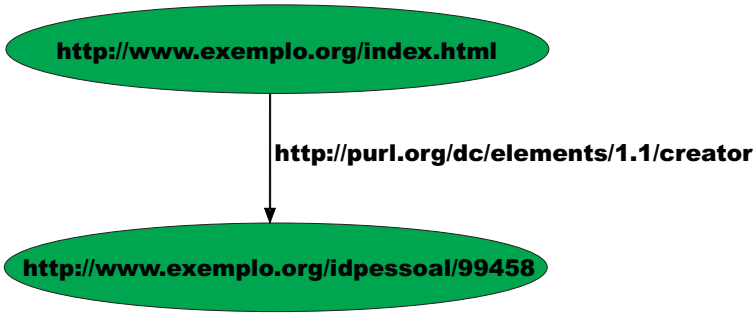

Figura 1: Uma declaração RDF simples .

2.2 O modelo RDF

Na Subseção 2.1, foram abordados os conceitos básicos de declarações RDF, o uso de referências URIs e a sintaxe RDF/XML. Nesta seção será descrito como RDF usa URIs para fazer declarações sobre recursos. Em RDF, a declaração em português "**`http://www.exemplo.org/index.html`** tem um **criador** cujo valor é **Mário Sabino**" poderia ser representada da seguinte maneira:

- O **sujeito**: `http://www.exemplo.org/index.html`
- O **predicado**: `http://purl.org/dc/elements/1.1/creator`
- O **objeto**: `http://www.exemplo.org/idpessoal/99458`

Ao invés de usar as palavras `criador` e `Mário Sabino` para identificar o predicado e o objeto, respectivamente, foram utilizadas referências URIs. O RDF modela as declarações como **nós** e **arcos** em um grafo. Nesta notação, uma declaração é representada por: um nó para o sujeito, um nó para o objeto e um arco para o predicado, dirigido do nó sujeito para o nó objeto [9]. A declaração acima poderia ser representada por um grafo como mostra a Figura 1.

Um conjunto de nomes é chamado de **vocabulário**. O vocabulário de um grafo é o conjunto de nomes que ocorrem como sujeito, predicado ou objeto de alguma tripla no grafo [7].

Grupos de declarações são representados por grupos correspondentes de nós e arcos. Por exemplo, sejam consideradas duas declarações em português sobre um mesmo recurso:

"**`http://www.exemplo.org/index.html`** tem uma **data de criação** cujo valor é **16 de Setembro de 2004**"

"**`http://www.exemplo.org/index.html`** tem uma **linguagem** cujo valor é **Português**"

Estas declarações podem ser representadas em um grafo RDF, como mostra a Figura 2, que ilustra como objetos em declarações RDF podem ser referências URI ou valores constantes (chamados de literais), representados por seqüências de caracteres. Literais não podem ser usados como sujeitos ou predicados em declarações RDF. No desenho de um grafo RDF, nós que são referências URI são mostrados como elipses, enquanto nós que são literais são mostrados como caixas.

Quando se deseja representar uma declaração RDF, o desenho de um grafo pode não ser o meio mais conveniente. Então, um caminho alternativo para escrever uma declaração RDF pode ser usado. Este caminho alternativo é chamado de **triplas RDF**. Na notação de triplas, cada declaração no grafo é escrita como uma tripla composta de sujeito, predicado e objeto, nesta

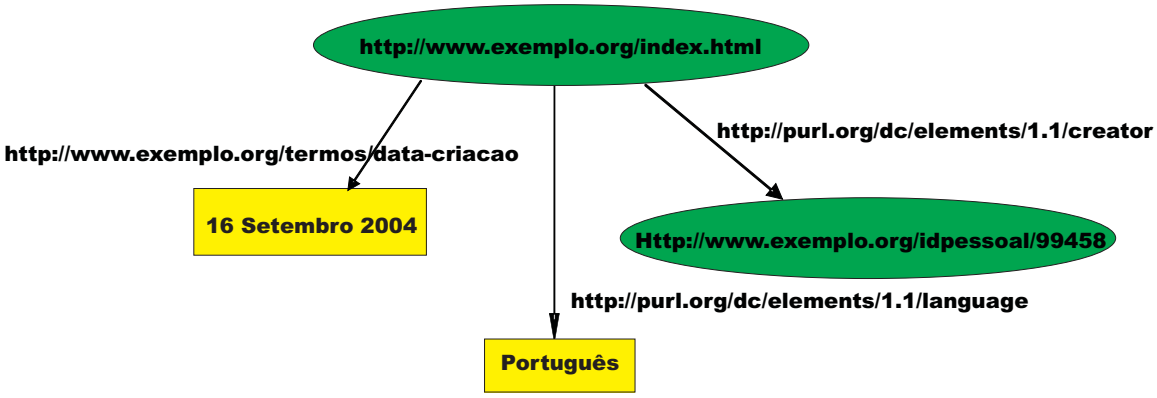

Figura 2: Várias declaração sobre o mesmo recurso.

ordem. Ou seja, a descrição de uma tripla é da forma: [`sujeito`, `predicado`, `objeto`]. Por exemplo, as três declarações mostradas na Figura 2 podem ser escritas na notação de tripla como mostra o Código 1.

Código 1 – Declarações RDF em forma de triplas.

```
1  <http://www.exemplo.org/index.html>
2  <http://purl.org/dc/elements/1.1/creator>
3  <http://www.exemplo.org/idpessoal/99458>.
4
5  <http://www.exemplo.org/index.html>
6  <http://www.exemplo.org/termos/data-criacao>
7  "16 Setembro 2004".
8
9  <http://www.exemplo.org/index.html>
10 <http://purl.org/dc/elements/1.1/language>
11 "portugues".
```

Cada tripla corresponde a um arco simples no grafo. Por exemplo, com base na Figura 2, a tripla (Código 1, linhas 1 a 3), corresponde ao arco do criador do recurso, `<http://purl.org/dc/elements/1.1/creator>`, que começa no nó `<http://www.exemplo.org/index.html>` e termina no nó `<http://www.exemplo.org/idpessoal/99458>`. As outras triplas seguem o mesmo raciocínio.

Como foi mostrado no exemplo do Código 1, a notação de triplas pode resultar em várias linhas longas em uma página. Por conveniência, pode-se usar taquigrafia[1] para se escrever triplas. Esta taquigrafia substitui uma referência URI completa por uma abreviação. Uma abreviação é formada por um nome qualificado XML (também chamado de *QName*) sem os sinais de < (menor) e > (maior). Um *QName* é formado por um prefixo associado a um *namespace*[2] URI [3], seguido por dois pontos **":"** e um nome

[1] Escrita abreviada e simplificada.

[2] *Namespaces* são usados para prevenir colisões de nomes, isto é, para identificar unicamente os elementos. Eles identificam uma parte da *Web* (espaço) que atua como um qualificador para um conjunto específico de nomes.

local. Por exemplo, se o prefixo de *QName* `carro` é associado ao *namespace* URI `http://exemplo.org/automoveis/`, então o *QName* `carro:corsa` é a taquigrafia para a referência URI `http://exemplo.org/automoveis/corsa`. A seguir são mostrados alguns prefixos de *QNames* muito utilizados e seus respectivos *namespaces* URI:

- Prefixo `rdf:` ⇒ *namespace* URI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- Prefixo `rdfs:` ⇒ *namespace* URI: `http://www.w3.org/2000/01/rdf-schema#`
- Prefixo `dc:` ⇒ *namespace* URI: `http://purl.org/dc/elements/1.1/` (refere-se ao vocabulário *Dublin Core*.)
- Prefixo `owl:` ⇒ *namespace* URI: `http://www.w3.org/2002/07/owl#`
- Prefixo `xsd:` ⇒ *namespace* URI: `http://www.w3.org/2001/XMLSchema#`

Considerando-se o exemplo da Figura 2, pode-se definir os seguintes prefixos:

- Prefixo `ex:` ⇒ *namespace* URI: `http://www.exemplo.org/` (refere-se a uma organização fictícia. Usada nos exemplos deste texto.)
- Prefixo `extermos:` ⇒ *namespace* URI: `http://www.exemplo.org/termos/` (termos usados pela organização.)
- Prefixo `expessoal:` ⇒ *namespace* URI: `http://www.exemplo.org/idpessoal/` (identificação dos funcionários da organização)

O conjunto de triplas do Código 1, usando-se os *Qnames* construídos a partir destes prefixos, pode ser reescrito como mostra o Código 2.

Código 2 – Declarações RDF em forma de triplas usando *Qnames*.

```
1 ex:index.html   dc:creator              expessoal:99458.
2 ex:index.html   extermos:data-criacao   "16 Setembro 2004.
3 ex:index.html   dc:language             "portugues".
```

Pode-se perceber claramente que, desta forma, a escrita das triplas é muito mais simples do que a forma usada no Código 1. Assim, este novo formato é preferível, já que ambas as representações têm o mesmo significado.

RDF usa referências URIs ao invés de palavras para nomear coisas nas declarações. Desta forma, um conjunto de referências URIs forma o seu vocabulário. Estas referências URIs são organizadas de forma que possam ser representadas como um conjunto de *QNames* que usam um prefixo comum [9], isto é, uma referência URI de um *namespace* comum será escolhida para todos os termos do vocabulário. A definição de um vocabulário pode ser feita da forma que melhor represente um domínio em questão.

Referências URIs de diferentes vocabulários podem ser livremente misturadas em um grafo RDF. Por exemplo, o grafo da Figura 2 usa referências URIs de três vocabulários: `extermos`, `expessoal` e `dc`. RDF também não impõe nenhuma restrição de como várias declarações que usem uma mesma referência URI como predicado, podem aparecer em um grafo para descrever o mesmo recurso. Por exemplo, o recurso `ex:index.html` pode

ter sido criado por um esforço cooperativo de vários membros da organização `exemplo.org`. Esta a declaração pode ser descrita como mostra o Código 3.

Código 3 – Declaração RDF em forma de triplas sobre os vários autores de um recurso.

```
1 ex:index.html    dc:creator    expessoal:99458.
2 ex:index.html    dc:creator    expessoal:14590.
3 ex:index.html    dc:creator    expessoal:35871.
```

Estes exemplos de declarações RDF ilustram algumas vantagens do uso de referências URIs para identificação de recursos. Uma das vantagens é que a identificação do sujeito da declaração pode ser mais precisa, isto é, o criador da página não é a seqüência de caracteres "Mário Sabino", nem uma pessoa qualquer chamada Mário Sabino, mas sim a pessoa chamada `Mário Sabino` que está associada à referência URI `http://www.exemplo.org/idpessoal/99458`.

O uso de referências URIs como sujeitos, predicados e objetos em declarações RDF permitem o desenvolvimento e uso de vocabulários compartilhados na *Web*, desde que pessoas possam descobrir e começar a usar vocabulários criados por outros, refletindo em um entendimento compartilhado destes conceitos. Este compartilhamento de vocabulários é um dos objetivos da *Web* Semântica [2].

Um predicado é uma referência URI e pode ser interpretado como um relacionamento entre dois nós ou como uma definição de um valor para um atributo de um nó sujeito [1], como mostra a tripla a seguir:

```
ex:index.html    dc:creator    expessoal:99458 .
```

Nesta tripla, o predicado `dc:creator`, quando expandido como uma referência URI, é uma referência não-ambígua para o atributo `creator` no conjunto de atributos de metadados do *Dublin Core*. O escritor desta tripla está efetivamente mostrando que o relacionamento entre a página *Web* (`http://www.exemplo.org/index.html`) e o criador da página (`http://www.exemplo.org/idpessoal/99458`), é exatamente o relacionamento identificado pelo conceito `http://purl.org/dc/elements/1.1/creator`.

Entretanto, o uso de referências URIs não resolvem todos os problemas de identificação. Por exemplo, pessoas podem usar diferentes referências URIs para referenciar as mesmas coisas. Por esta razão, é aconselhável tentar usar termos de vocabulários existentes quando possível, ao contrário de "reinventar a roda".

2.3 Propriedades estruturadas e nós em branco

Todos os exemplos de declarações em RDF mostrados nas seções anteriores possuem estruturas bastante simples. Entretanto, RDF permite fazer declarações mais complexas, como por exemplo, registrar o endereço de Mário Sabino em uma estrutura onde rua, cidade, estado e CEP estão em declarações separadas. Informações estruturadas, como um endereço, são representadas em RDF agregando-se os objetos a serem descritos como um recurso e fazendo-se declarações sobre o novo recurso, como ilustra a Figura 3.

A Figura 3 mostra um grafo RDF que representa um endereço. Ele usa um nó sem uma referência URI para representar o endereço de Mário Sabino. Este nó em branco serve para possibilitar a conectividade entre várias partes de um grafo [3] [9].

[3] Nós em branco são chamados de recursos anônimos.

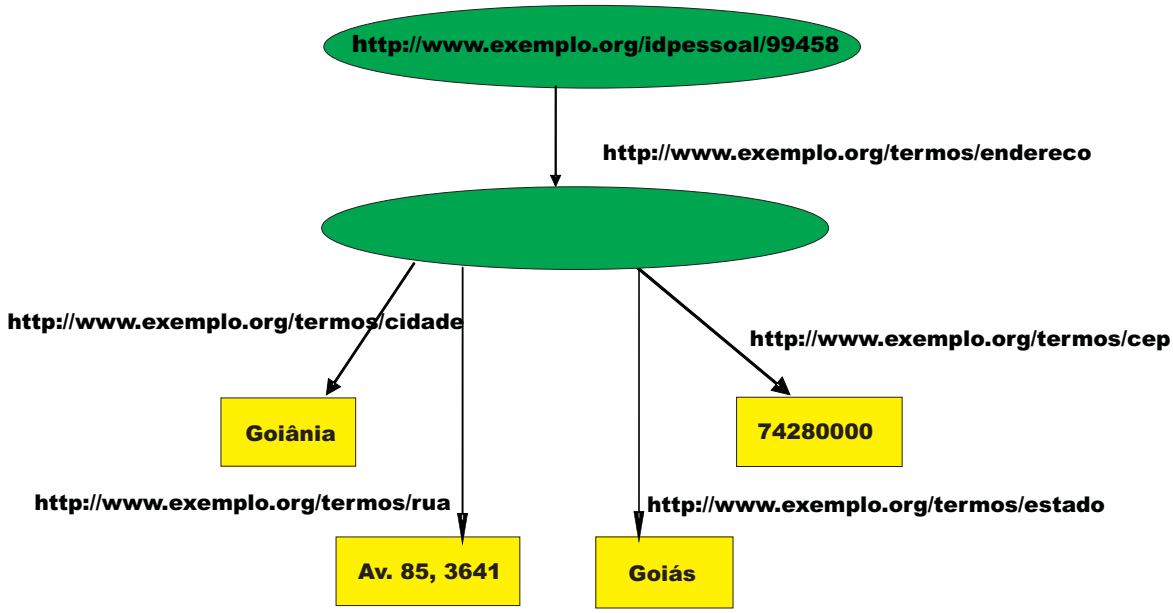

Figura 3: Uso de nó em branco.

Algumas vezes é necessário representar um grafo que contém um nó em branco na forma de triplas. As triplas correspondentes à Figura 3 podem ser representadas como mostra o Código 4.

Código 4 – Triplas representado um grafo com nó em branco.

```
1 expessoal:99451     extermos:endereço       ???.
2 ???                 extermos:rua            "Av. 85, 3641".
3 ???                 extermos:cidade         "Goiânia".
4 ???                 extermos:estado         "Goiás".
5 ???                 extermos:cep            "74280000".
```

O sinal **???** indica a presença de um nó em branco. Contudo, como grafos mais complexos podem conter vários nós em branco, é necessário uma maneira para diferenciar os vários nós em branco em uma tripla. Assim, triplas devem usar identificadores de nós em branco. Segundo [9], estes identificadores têm a forma `_:nome`. No exemplo das triplas que representam o grafo da Figura 3, no lugar de **???**, o identificador de um nó em branco poderia ser `_:enderecoMario`.

2.4 Literais tipados

Um literal tipado RDF é formado por um par constituído de uma seqüência de caracteres e uma referência URI que identifica um tipo de dado particular [9]. Os dois elementos do par são separados pelo símbolo "^^". Por exemplo, usando-se um literal tipado, a idade de `Mário Sabino` poderia ser descrita como sendo o número inteiro "52", como mostra a Figura 4.

Nos exemplos das seções anteriores, todos os valores constantes que servem como objetos nas declarações RDF foram representados por literais não tipados. O uso de literais não tipados

pode causar confusão. Por exemplo, caso o valor de um objeto em uma declaração RDF seja "11", não se pode afirmar que "11" seja a idade de uma pessoa, uma seqüência de caracteres qualquer ou mesmo que seja um número no formato binário.

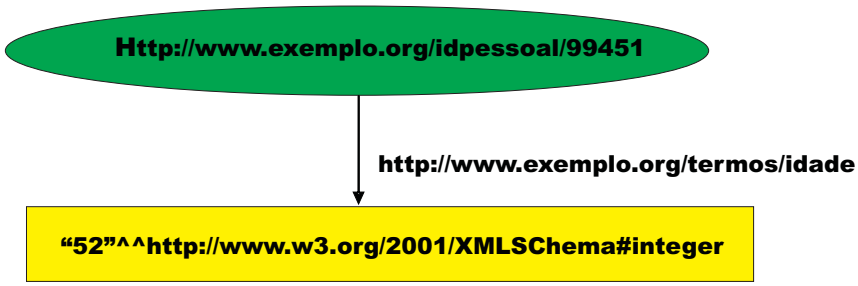

Figura 4: Um literal tipado para a idade de Mário Sabino.

A Figura 4 pode ser representada na forma de uma tripla, como é mostrado a seguir:

```
expessoal:99451     extermos:idade     "52"^^xsd:integer.
```

Diferentemente das linguagens de programação típicas, RDF não possui o seu próprio conjunto de tipos de dados. Em vez disso, os tipos de dados usados em literais tipados RDF são definidos externamente e identificados pelo URI do tipo de dado.

Um *software* chamado para processar dados RDF, que contenha referências para tipos de dados, deve ser capaz de encontrar erros no processamento, caso encontre coisas "estranhas", como acontece na tripla mostrada a seguir:

```
expessoal:99451     extermos:idade     "casa"^^xsd:integer.
```

Esta tripla é uma tripla RDF válida mas, obviamente, contém um erro de tipos, uma vez que "casa" não é do tipo `xsd:integer`. Logo, um *software* construído para processar estes dados RDF deve ser capaz de tratar este tipo de erro. Na Seção 3, a seguir, será discutida a sintaxe RDF/XML para representação de grafos e triplas RDF em uma linguagem.

3 Sintaxe RDF/XML

RDF provê uma sintaxe XML para o intercâmbio e escrita de grafos RDF, chamada de RDF/XML. Para ilustrar esta sintaxe serão usados alguns exemplos. Assim, para uma declaração em português: "**`http://www.exemplo.org/index.html`** tem uma **data de criação** cujo valor é **16 de Setembro de 2004**.", pode-se gerar o grafo RDF correspondente, conforme mostrado na Figura 5. O Código 5, mostra a sintaxe RDF/XML[4] correspondente ao grafo desta figura.

[4]Segundo Beckett [1], é recomendado que os arquivos RDF/XML tenham a extensão `.rdf` em todas as plataformas.

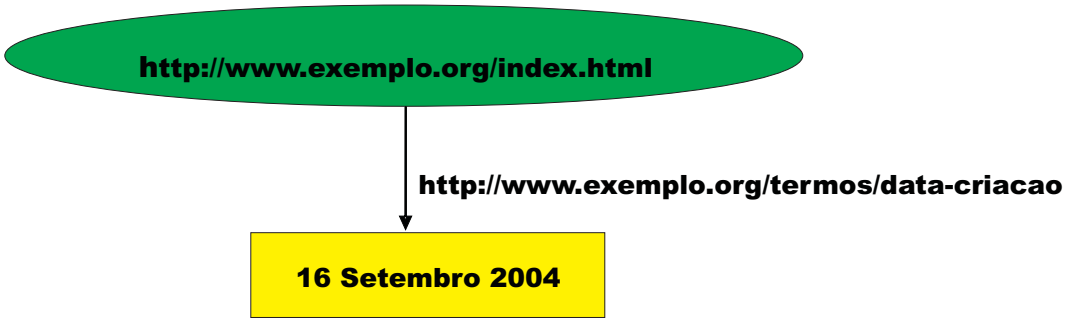

Figura 5: Descrevendo-se a data de criação de uma página *Web*.

Código 5 Sintaxe RDF/XML correspondente a um grafo RDF.

```
1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3          xmlns:extermos="http://www.exemplo.org/termos/">
4
5  <rdf:Description rdf:about="http://www.exemplo.org/index.html">
6     <extermos:data-criacao>16 Setembro 2004</extermos:data-criacao>
7  </rdf:Description>
8
9 </rdf:RDF>
```

A linha 1 é uma declaração XML indicando que este é um documento XML e qual a versão da XML que está sendo usada. A linha 2 se inicia com o elemento `rdf:RDF` indicando que o conteúdo XML vai representar um documento RDF. Depois do elemento `rdf:RDF`, aparece uma declaração de *namespace* XML, representando um atributo do elemento `rdf:RDF`. Esta declaração especifica que todas as *tags* com o prefixo `rdf:` são parte do *namespace* identificado pela referência URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#`. Esta referência URI é usada para os termos do vocabulário RDF.

A linha 3 especifica outra declaração de *namespace* XML com o prefixo "`extermos`". *Tags* com o prefixo `extermos` são usadas para os termos do vocabulário definido pelo exemplo da organização, `exemplo.org`. As linhas 1-3 são, geralmente, necessárias para indicar que o conteúdo é RDF/XML e para identificar os *namespaces* que serão usados.

As linhas 5-7 mostram, em RDF/XML, a declaração da Figura 5. O modo como RDF/XML representa uma declaração RDF é bastante simples: uma declaração RDF é uma descrição a respeito do sujeito da declaração [9]. Esta descrição é feita através de predicado(s) e objeto(s). A *tag* inicial `rdf:Description`, na linha 5, indica o início de uma descrição de um recurso e identifica o recurso que é declarado, o qual vai ser descrito usando-se o atributo `rdf:about`. A linha 6 mostra uma propriedade do elemento com a *tag QName* igual a `extermos:data-criacao`, para representar o predicado e o sujeito da declaração. O conteúdo deste elemento de propriedade é o objeto da declaração, `16 Setembro 2004`. O elemento de propriedade aninhado dentro do elemento `rdf:Description` indica que esta propriedade é aplicada ao recurso especificado no atributo `rdf:about` do elemento `rdf:Description`. A linha 7 indica o fim do elemento `rdf:Description`.

A linha 9 indica o fim de uma declaração RDF. Vale lembrar que as linhas em branco,

linha 4 e linha 8, são incluídas somente para facilitar a visualização do código.

O código anterior ilustra a idéia básica usada por RDF/XML para codificar um grafo RDF. Os nós e os predicados são representados por termos XML, ou seja, nomes de elementos, nomes de atributos, conteúdo de elementos e valores de atributos [1]. As referências URIs de predicados são escritas como *QNames* XML. As referências URIs de nós sujeitos são escritas como valores de atributos XML. Nós literais são representados por "caracteres de dados"[5] de elementos ou valores de atributos.

Um mesmo recurso pode ter várias propriedades e valores. Um caminho natural é codificar um elemento `rdf:Description` para cada propriedade. Entretanto, RDF/XML permite múltiplas propriedades de elementos aninhadas dentro de um mesmo elemento `rdf:Description`. Por exemplo, o Código 2, mostra algumas declarações sobre `http://www.exemplo.org/index.html`. Este grupo de declarações pode ser escrito usando-se a sintaxe RDF/XML, como mostra o Código 6.

Código 6 Sintaxe RDF/XML correspondente a triplas RDF.

```
1  <?xml version="1.0"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3           xmlns:dc="http://purl.org/dc/elements/1.1/"
4           xmlns:extermos="http://www.exemplo.org/termos/">
5
6   <rdf:Description rdf:about="http://www.exemplo.org/index.html">
7    <extermos:data-criacao>16 Setembro 2004</extermos:data-criacao>
8    <dc:language>portugues</dc:language>
9    <dc:creator rdf:resource="http://www.exemplo.org/idpessoal/99458"/>
10  </rdf:Description>
11
12 </rdf:RDF>
```

No Código 6, a propriedade de elemento `dc:creator` (linha 9) representa uma propriedade cujo valor é um outro recurso, identificado por sua referência URI. Além disso, três propriedades do sujeito estão representadas aninhadas dentro do elemento `rdf:Description`.

RDF/XML pode também representar grafos que contém nós em branco. A Figura 6 mostra um grafo que contém um nó em branco. Um nó em branco é representado em RDF/XML usando-se o atributo `rdf:nodeID`, com um identificador do nó em branco como seu valor [9]. Especificamente, uma declaração com um nó em branco como sujeito pode ser escrita em RDF/XML usando-se um elemento `rdf:Description` com um atributo `rdf:nodeID`, ao invés do atributo `rdf:about`. Similarmente, uma declaração com um nó em branco como seu objeto pode ser escrita usando-se uma propriedade de elemento com um atributo `rdf:nodeID`, ao invés de um atributo `rdf:resource`. O código em RDF/XML correspondente à Figura 6 é mostrado no Código 7.

[5]Os caracteres entre *tags* de início e fim, que representam dados.

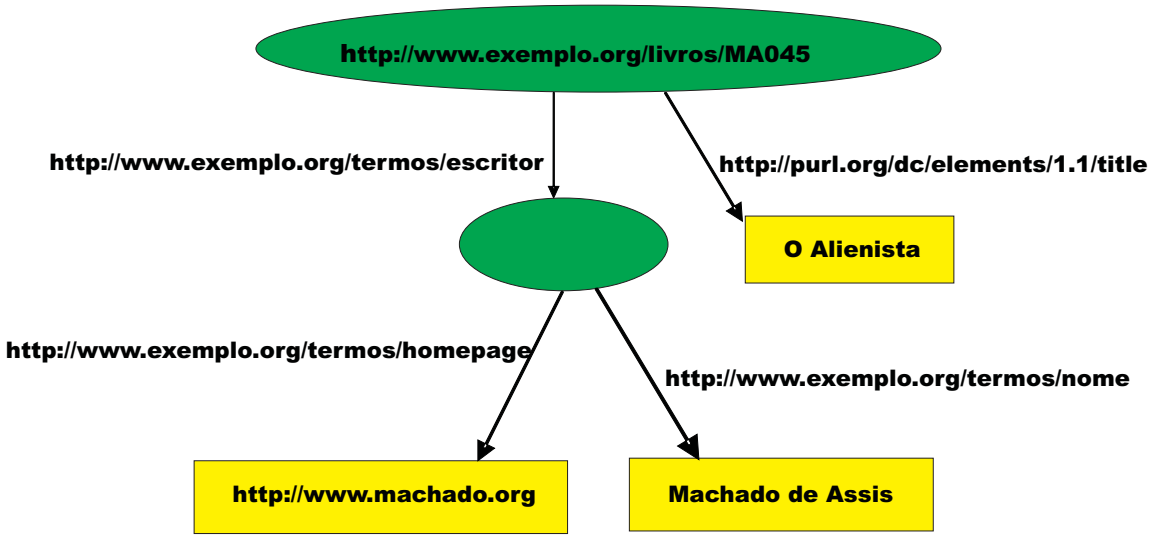

Figura 6: Um grafo contendo um nó em branco.

Código 7 Sintaxe RDF/XML correspondente a um grafo RDF com nó em branco.

```
1  <?xml version="1.0"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3           xmlns:dc="http://purl.org/dc/elements/1.1/"
4           xmlns:extermos="http://www.exemplo.org/pessoal/1.0/">
5
6    <rdf:Description rdf:about="http://www.exemplo.org/livros/MA045">
7      <dc:title>O Alienista</dc:title>
8      <extermos:escritor rdf:nodeID="ma"/>
9    </rdf:Description>
10
11   <rdf:Description rdf:about="ma">
12     <extermos:nome>Machado de Assis</extermos:nome>
13     <extermos:homePage rdf:resource="http://www.machado.org"/>
14   </rdf:Description>
15
16 </rdf:RDF>
```

Neste código, o identificador do nó em branco `ma` é usado na linha 11 para identificar o nó em branco como o sujeito de várias declarações, e é usado na linha 8, para indicar que o nó em branco é o valor da propriedade de recurso `extermos:escritor`.

A sintaxe RDF/XML também pode representar literais tipados. Um literal tipado é representado em RDF/XML adicionando-se o atributo `rdf:datatype`. Por exemplo, a declaração a seguir:

`ex:index.html extermos:data-criacao` “24-04-2004” `^^xsd:date .`

pode ser representada, na sintaxe RDF/XML, como mostrado no Código 8.

Código 8 Sintaxe RDF/XML correspondente a triplas RDF com um literal tipado.

```
1  <?xml version="1.0"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3           xmlns:extermos="http://www.exemplo.org/termos/">
4
5   <rdf:Description rdf:about="http://www.exemplo.org/index.html">
6     <extermos:data-criacao
7       rdf:datatype="http://www.w3.org/2001/xmlschema#date">24-04-2004
8     </extermos:data-criacao>
9   </rdf:Description>
10
11 </rdf:RDF>
```

Na Subseção 3.1, a seguir, apresenta-se uma discussão sobre os contêiners que permitem descrever grupos de recursos.

3.1 Contêiner RDF

RDF oferece mecanismos que possibilitam a descrição de grupos de recursos ou valores, chamados de **contêiners** [9]. Um contêiner é um recurso que contém objetos, sendo estes chamados de membros. Os membros de um contêiner podem ser recursos ou literais. RDF define três tipos de contêiners: *Bag*, *Sequence* ou *Seq* e *Alternative* ou *Alt* [9].

Uma *Bag* (tipo `rdf:Bag`) representa um grupo de recursos ou de valores não ordenados, utilizados para declarar que uma propriedade pode ser composta de múltiplos valores, independentes da ordem de atribuição [10]. Valores duplicados são permitidos.

Uma *Sequence* (tipo `rdf:Seq`) representa um grupo de recursos ou de valores ordenados, utilizados para declarar que uma propriedade pode ser composta de múltiplos valores que obedecem à uma determinada ordem [10]. Valores duplicados são permitidos.

Uma *Alternative* (tipo `rdf:Alt`) representa um grupo de recursos ou de valores que apresentam valores possíveis para uma propriedade, ou seja, apresentam várias alternativas para uma propriedade [10].

Para descrever um recurso como sendo um tipo de contêiner, o recurso recebe uma propriedade `rdf:type` cujo valor é um dos tipos predefinidos, ou seja, `Bag`, `Seq` ou `Alt`. Os membros de um contêiner podem ser descritos pela definição de uma propriedade de associação ao contêiner para cada membro, onde o contêiner é o sujeito e os membros são os objetos [9]. Estas propriedades de associação possuem nomes, que seguem a forma `rdf:_n`, onde "n" é um inteiro decimal maior do que zero.

Um uso típico de um contêiner é para indicar que o valor de uma propriedade é um grupo de coisas. Por exemplo, para representar a declaração "A fábrica 2.15 tem os diretores Alberto, Tatiane e Jeová", **diretores** poderia ser descrito pela propriedade `d:diretores` cujo valor é um contêiner do tipo `rdf:Bag`, contendo um grupos de pessoas, como mostra o grafo da Figura 7. A sintaxe RDF/XML oferece uma sintaxe especial e abreviações para descrever contêiners. O Código 9, descreve a Figura 7 usando esta sintaxe.

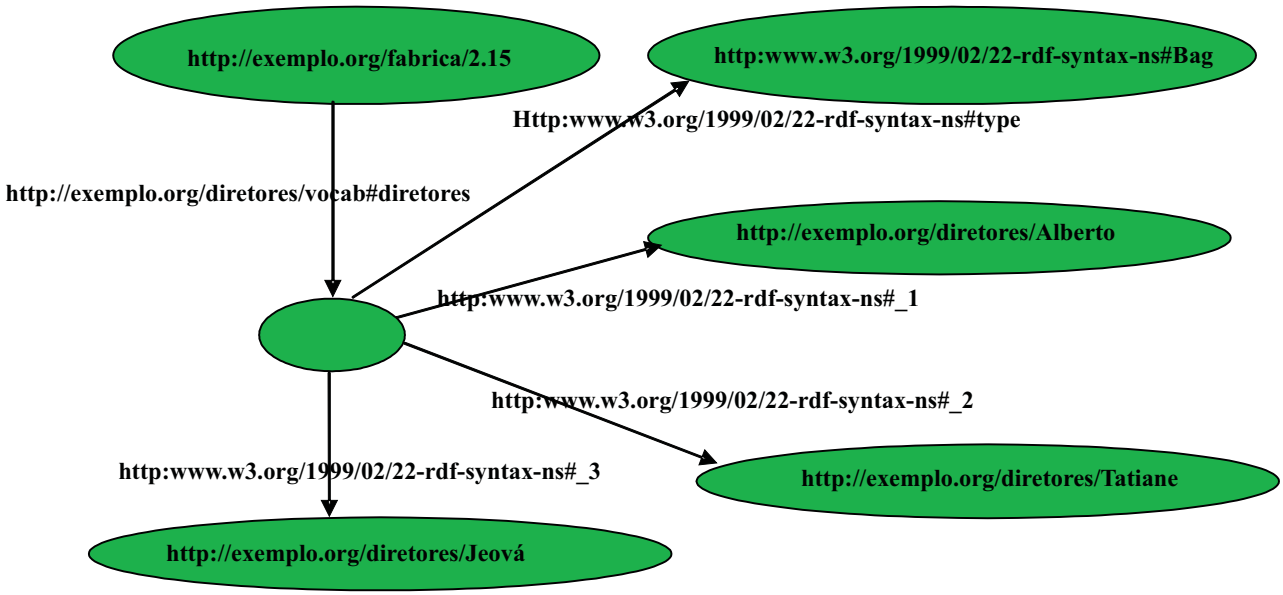

Figura 7: Descrição de um contêiner *Bag*.

Código 9 Sintaxe RDF/XML correspondente à descrição de um contêiner *Bag*.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:d="http://www.exemplo.org/diretores/vocab#">

  <rdf:Description rdf:about="http://exemplo.org/fabrica/2.15">
    <d:diretores>
      <rdf:Bag>
        <rdf:li rdf:resource="http://exemplo.org/diretores/Alberto"/>
        <rdf:li rdf:resource="http://exemplo.org/diretores/Tatiane"/>
        <rdf:li rdf:resource="http://exemplo.org/diretores/Jeova"/>
      <rdf:Bag>
    </d:diretores>
  </rdf:Description>

</rdf:RDF>
```

O Código 9 inclui o elemento `rdf:li` para evitar o uso de números para cada propriedade associada. Ele é equivalente a `rdf:_1`, `rdf:_2` e `rdf:_3`. O elemento `<rdf:Bag>` é aninhado dentro do elemento `<d:diretores>`.

A estrutura de um grafo para um contêiner `rdf:Seq` e a sintaxe RDF/XML correspondente é similar à `rdf:Bag`. Um contêiner `Alt` possui sintaxe RDF/XML similar à `rdf:Bag`, contudo, um contêiner `Alt` deve ter pelo menos um membro, identificado pela propriedade `rdf:_1`. Este membro deve ser considerado como valor padrão. Na Subseção 3.2 a seguir, se discute outra forma de agrupar recursos em RDF, as coleções.

3.2 Coleções RDF

Uma limitação do uso de contêiners é que não há uma forma para dizer coisas como "estes são todos os diretores da fábrica x", por exemplo. Um contêiner somente diz que certos recursos são membros de determinado recurso, mas não diz se existem outros membros. RDF permite descrever grupos contendo somente membros especificados, chamados de **coleções** RDF.

Uma coleção RDF é um grupo de objetos representado como uma lista estruturada em um grafo RDF [9]. Esta lista estruturada é construída usando-se um vocabulário predefinido. Coleções permitem estruturas ramificadas e têm uma terminação de elementos explícita, ou seja, permitem que as aplicações determinem o conjunto exato de itens na coleção [7]. Este vocabulário consiste do tipo predefinido `rdf:List`, das propriedades predefinidas `rdf:First` e `rdf:rest` e do recurso predefinido `rdf:nil` [9]. Por exemplo, a declaração "A fábrica 2.15 tem os diretores Alberto, Tatiane e Jeová e mais ninguém", poderia ser representada usando-se o grafo mostrado na Figura 8.

Figura 8: Descrição de uma coleção RDF.

No grafo da Figura 8, cada membro da coleção é um objeto de uma propriedade `rdf:first`, cujo sujeito (um nó em branco neste exemplo) é um recurso que representa uma lista. Esta lista de recursos é ligada ao resto da lista pela propriedade `rdf:rest`. O fim da lista é indicado pela propriedade `rdf:rest` tendo como seu objeto o recurso `rdf:nil`. O recurso `rdf:nil` representa uma lista vazia e é definido como sendo do tipo `rdf:List`.

Na sintaxe RDF/XML, uma coleção pode ser descrita por uma propriedade de elemento que tem o atributo `rdf:parseType:"Collection"` e que contém um grupo de elementos aninhados que representam os membros da coleção [9]. O atributo `rdf:parseType` indica

que o conteúdo de um elemento será interpretado de um modo especial. No caso de coleção, o atributo `rdf:parseType:"Collection"` indica que os elementos são usados para criar uma lista estruturada no grafo RDF. O Código 10, mostra o código RDF/XML correspondente à Figura 8.

Código 10 Sintaxe RDF/XML correspondente à descrição de uma coleção RDF.

```
1  <?xml version="1.0"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3           xmlns:d="http://www.exemplo.org/diretores/vocab#">
4
5   <rdf:Description rdf:about="http://exemplo.org/fabrica/2.15">
6     <d:diretores rdf:parseType="Collection">
7       <rdf:Description
8              rdf:resource="http://exemplo.org/diretores/Alberto"/>
9       <rdf:Description
10             rdf:resource="http://exemplo.org/diretores/Tatiane"/>
11      <rdf:Description
12             rdf:resource="http://exemplo.org/diretores/Jeova"/>
13    </d:diretores>
14  </rdf:Description>
15
16 </rdf:RDF>
```

Na Subseção 3.3 a seguir, é apresentado o mecanismo de reificação que permite fazer declarações sobre declarações.

3.3 Declarações sobre declarações

O modelo RDF permite descrever suas próprias declarações através do mecanismo de **reificação**. Segundo Manola e Miller [9], uma reificação em RDF permite que um enunciado (*statement*) possa ser tratado como um recurso com quatro propriedades:

- sujeito (*subject*): define o recurso que está sendo descrito pela declaração modelada, isto é, o valor é o recurso sobre o qual a declaração original é feita;
- predicado (*predicate*): identifica a propriedade original da declaração modelada;
- objeto (*object*): identifica o valor da propriedade em uma declaração modelada;
- tipo (*type*): descreve o tipo do novo recurso. Todas as declarações reificadas são instâncias do `rdf:statement`.

O Código 11, mostra uma reificação de uma declaração sobre o `preço` de um `carro`. Esta declaração é nomeada com referência URI `exprodutos:tripla09` (linha 1). Nas linhas seguintes, são feitas declarações a respeito desta declaração. Assim, a linha 2 especifica seu sujeito; a linha 3 o seu predicado; e a linha 4 o objeto do predicado.

A Figura 9 mostra a declaração representada no Código 11. No Código 12, a mesma representação é feita na forma de RDF/XML.

Código 11 Triplas representado uma reificação.

```
1 exprodutos:tripla09     rdf:type         rdf:Statement.
2 exprodutos:tripla09     rdf:subject      exprodutos:item159.
3 exprodutos:tripla09     rdf:predicate    extermos:preco.
4 exprodutos:tripla09     rdf:object       "20.000"^^xsd:decimal.
```

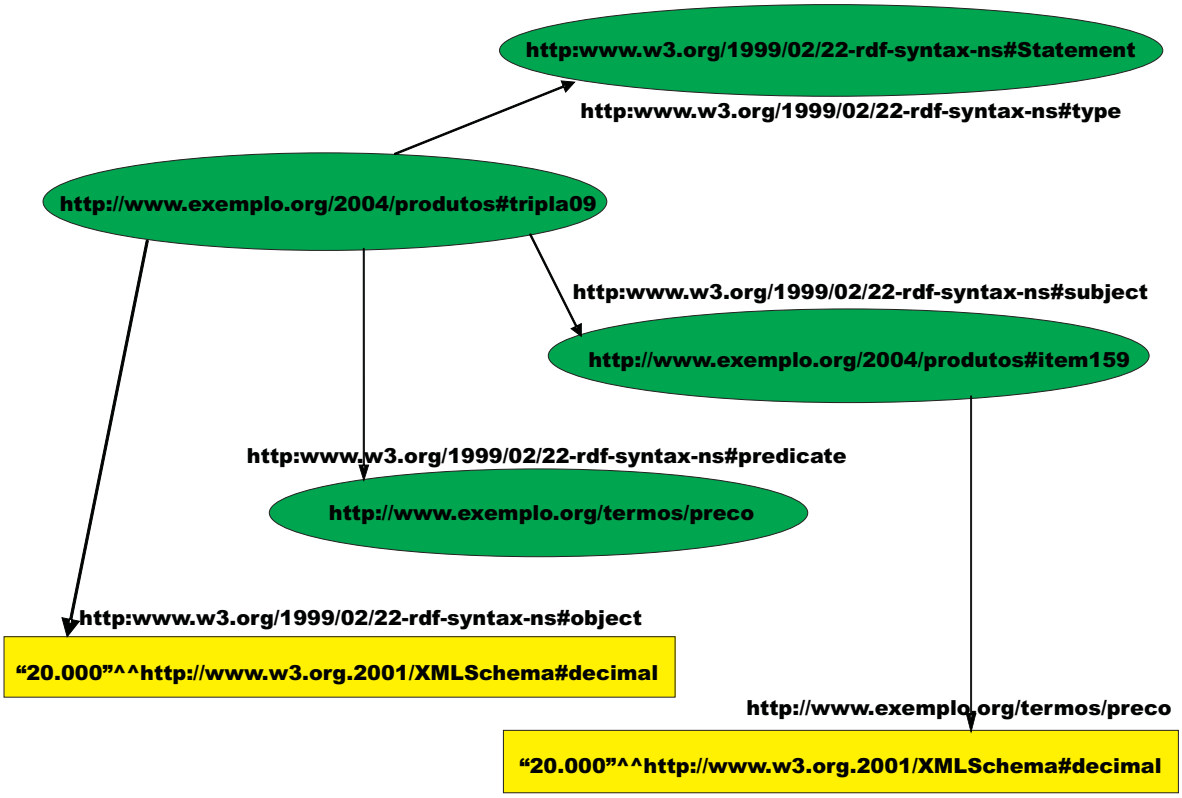

Figura 9: Uma declaração, sua reificação e sua atribuição.

Código 12 Sintaxe RDF/XML para uma reificação.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:extermos="http://www.exemplo.org/termos/"
         xml:base="http://www.exemplo.org/2004/produtos/">

 <rdf:Description rdf:ID="item159">
   <extermos:preco rdf:datatype="&xsd;decimal>20.000</extermos:preco>
 </rdf:Description>

 <rdf:Statement rdf:about="#tripla09">
   <rdf:subject
      rdf:resource="http://www.exemplo.org/2004/produtos#item159"/>
   <rdf:predicate
      rdf:resource="http://www.exemplo.org/termos/preco"/>
   <rdf:object rdf:datatype=
      "http://www.w3.org/2001/XMLSchema#decimal">20.000</rdf:object>
 </rdf:Statement>

</rdf:RDF>
```

Dois novos elementos sintáticos são empregadas no Código 12. O primeiro é a declaração `xml:base`. Ela especifica que a URI base para o conteúdo dentro do elemento `rdf:RDF` é `http://www.exemplo.org/2004/produtos`, e que todas as referências URI citadas dentro do conteúdo, sem um *QName*, serão interpretadas com relação à base [9]. O segundo é que o elemento `rdf:Description` possui o atributo `rdf:ID`, ao invés de atributo `rdf:about`. O atributo `rdf:ID` especifica um fragmento identificador, sendo uma abreviação para referência URI `http://www.exemplo.org/2004/produtos#item159`, onde `http://www.exemplo.org/2004/produtos` é a URI base. Na Seção 4, a seguir, será discutido o RDF *Schema* que permite a construção de um vocabulário para RDF.

4 Esquema RDF (RDF *Schema*)

Um Esquema RDF é construído sobre o modelo RDF básico, estendendo-o para incluir um vocabulário maior com restrições semânticas mais complexas [7]. Este mecanismo permite que grupos de descrições de recursos criem e compartilhem seus próprios vocabulários [6]. Um Esquema RDF é especificado como um conjunto de classes, propriedades e restrições entre seus relacionamentos [10].

Um Esquema RDF provê um sistema de tipos para RDF que é similar, em alguns aspectos, aos sistemas de tipos de linguagens de programação orientadas a objetos. Um Esquema RDF permite, por exemplo, que recursos possam ser instâncias de uma ou mais classes (indicado pela propriedade `rdf:type`). As classes são abstrações para agrupar recursos com características similares e podem ser organizadas de forma hierárquica. Por exemplo, uma classe `ex:MiniVan` poderia ser considerada como uma subclasse da classe `ex:Van` que é uma subclasse da classe `ex:Veiculo`. Isto significa que qualquer recurso do tipo `rdf:type ex:MiniVan` é também do tipo `rdf:type ex:Veiculo`. Um Esquema RDF descreve a propriedade `rdfs:subClassOf` para especificar tais relacionamentos entre classes.

Um Esquema RDF possui um vocabulário RDF que é um conjunto predefinido de recursos RDF com seu próprio significado especial [9]. Estes recursos têm uma referência URI com o prefixo `http://ww.w3.org/2000/02/rdf-schema#` (associado com o *QName* `rdfs:`). Na Subseção 4.1, a seguir, é apresentada a noção de classes em Esquemas RDF.

4.1 Classes

Segundo Hayes [7], classes em Esquemas RDF podem ser consideradas como algo a mais do que simples conjuntos. Elas correspondem ao conceito genérico de um **Tipo** ou **Categoria**, como a noção de classes em linguagens orientadas a objetos[9].

Com a utilização de classes, o modelo RDF torna-se extensível, pois as definições de esquemas existentes podem ser herdadas, especializando metadados de um determinado domínio, promovendo o reuso e o compartilhamento de esquemas [6]. O reuso e o compartilhamento de esquemas são fatores fundamentais para a consolidação da *Web* Semântica.

Classes RDF podem ser usadas para representar quase todas as coisas, tais como páginas *Web*, pessoas ou conceitos abstratos. As classes são geralmente identificadas por referências URIs. Classes são descritas usando-se `rdfs:Class` e `rfds:Resource` e as propriedades são descritas usando-se `rdf:type` e `rdfs:subClassOf`. Os recursos que pertencem a uma classe são chamados de suas **instâncias**. As classes são recursos por si mesmas [5]. As instâncias são indicadas através da propriedade `rdf:type`.

Em um Esquema RDF, uma classe é algum recurso que tem uma propriedade `rdf:type` cujo valor é o recurso `rdfs:Class`. O Código 13, mostra uma tripla em um Esquema RDF que descreve uma classe de veículo a motor de uma organização, por exemplo, `exemplo.org`, que deseja criar um esquema em RDF para representar diferentes tipos de veículos a motor.

Código 13 Uma tripla representado uma classe.

```
1 sve:VeiculoMotor        rdf:type        rdfs:Class.
```

O prefixo `sve:` refere-se a `http://www.exemplo.org/schemas/veiculos`. O Código 14 mostra o código RDF/XML para descrever a declaração da classe `sve:VeiculoMotor`, representada em forma de tripla no Código 13.

Código 14 Sintaxe RDF/XML para declaração de uma classe.

```
1 <rdf:Description rdf:ID="eiculoMotor">
2  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
3 </rdf:Description">
```

RDF/XML permite o uso de abreviações para descrever recursos que possuem a propriedade `rdf:type`. Uma vez que classes RDF *Schema* são recursos RDF, estas podem ser aplicadas para descrever classes. Assim, pode-se dizer que veículo a motor é uma classe, usando-se uma outra notação RDF/XML:

```
<rdfs:Class rdf:ID="VeiculoMotor">
```

O uso de `rdf:ID` é útil porque abrevia a referência URI e permite uma checagem adicional de que o valor do atributo `rdf:ID` seja único na base do URI corrente [9]. Então, tendo-se descrito veículo a motor como uma classe, pode-se usar a propriedade `rdf:type` para indicar que o recurso `excarro:onibusXY` é uma instância da classe `sve:VeiculoMotor`:

```
excarro:onibusXY        rdf:type        sve:VeiculoMotor.
```

Segundo [5], RDF distingue uma classe e o conjunto de suas instâncias. Associado à cada classe está o conjunto de suas instâncias, chamado de **extensão de classe**. Duas classes podem ter o mesmo conjunto de instâncias, mesmo sendo classes diferentes.

A declaração anterior usa a convenção, não obrigatória em RDF *Schema*, de que os nomes de classes são escritos com a primeira letra maiúscula, enquanto os nomes de propriedades e instâncias são escritos com a primeira letra minúscula.

Este Esquema RDF permite relacionamentos entre classes. O relacionamento de especialização entre duas classes é descrito usando-se a propriedade predefinida `rdfs:subClassOf`. O Código 15, mostra que `sve:Van` é uma classe e é uma especialização da classe `sve:VeiculoMotor`. O significado do relacionamento `rdfs:subClassOf` é que qualquer instância da classe `sve:Van` é uma instância da classe `sve:VeiculoMotor` [9]. A propriedade `rdfs:subClassOf` é transitiva [5], como mostra o Código 16.

Código 15 Uma tripla representado uma subclasse.

```
1 sve:Van     rdf:type            rdfs:Class.
2 sve:Van     rdfs:subClassOf     sve:VeiculoMotor.
```

Código 16 Uma tripla mostando que `rdfs:subClassOf` é transitiva.

```
1 sve:Van          rdfs:subClassOf     rdfs:VeiculoMotor.
2 sve:MiniVan      rdfs:subClassOf     sve:Van.
```

Um Esquema RDF define que `sve:MiniVan` também é uma subclasse de `sve:VeiculoMotor`. Com isso, os recursos que são instâncias `sve:MiniVan` também são instâncias da classe `sve:VeiculoMotor`. Uma classe pode ser subclasse de uma ou mais classes [9]. Todas as classes, em RDF *Schema*, são subclasses da classe `rdfs:Resource` [9]. A Figura 10 mostra uma hierarquia de classes que cria um esquema sobre o domínio de veículos.

A sintaxe RDF/XML que representa o grafo da Figura 10 é mostrado no Código 17. Na Subseção 4.2, a seguir, se discute sobre as propriedades que representam as relações entre classes e instâncias das classes em RDF `Schema`.

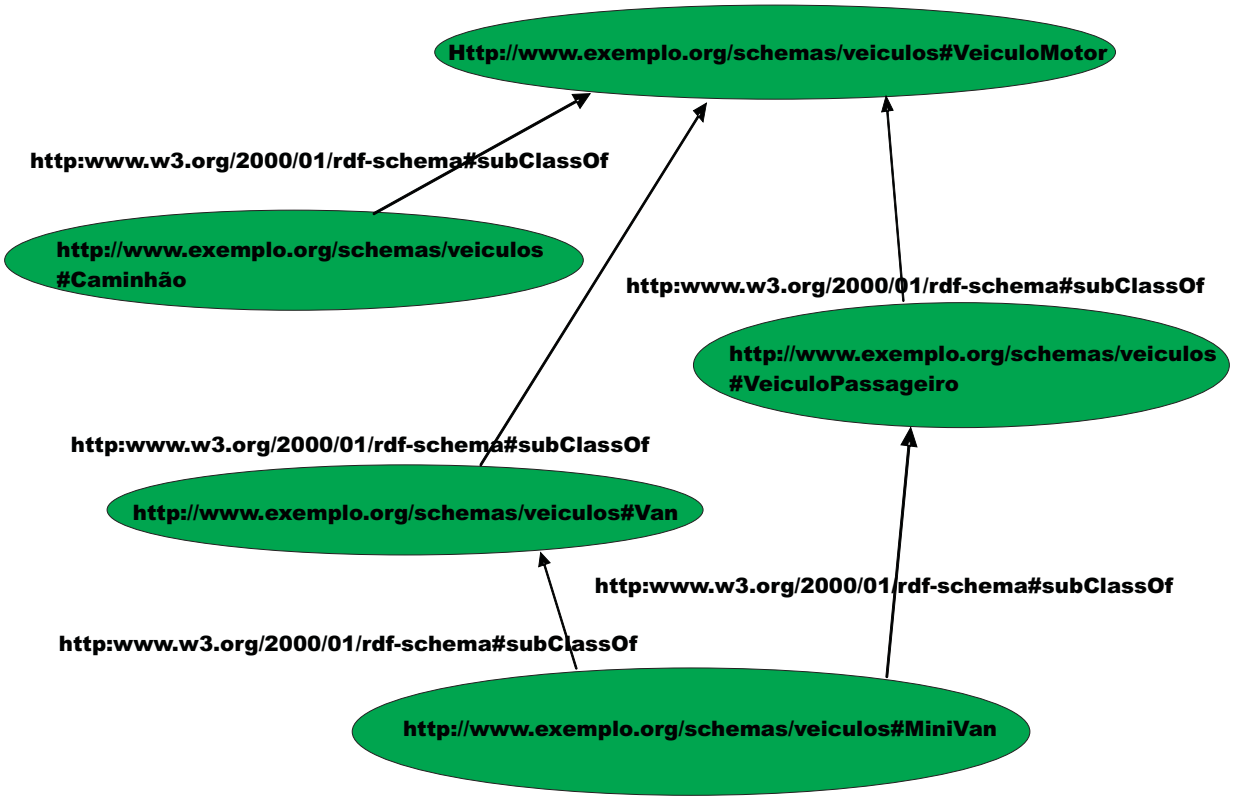

Figura 10: Hierarquia de classes.

Código 17 A sintaxe RDF/XML representando uma hierarquia de classes.

```
1  <?xml version="1.0"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4           xml:base="http://www.exemplo.org/schemas/veiculos/">
5
6  <rdfs:Class rdf:ID="VeiculoMotor"/>
7
8  <rdfs:Class rdf:ID="VeiculoPassageiro">
9      <rdfs:subClassOf rdf:resource="#VeiculoMotor"/>
10 </rdfs:Class>
11
12 <rdfs:Class rdf:ID="Caminhão">
13     <rdfs:subClassOf rdf:resource="#VeiculoMotor"/>
14 </rdfs:Class>
15
16 <rdfs:Class rdf:ID="Van">
17     <rdfs:subClassOf rdf:resource="#VeiculoMotor"/>
18 </rdfs:Class>
19
20 <rdfs:Class rdf:ID="MiniVan">
21     <rdfs:subClassOf rdf:resource="#Van"/>
22     <rdfs:subClassOf rdf:resource="#VeiculoPassageiro"/>
23 </rdfs:Class>
24
25 </rdf:RDF>
```

4.2 Propriedades

Propriedades permitem expressar relações entre classes e suas instâncias ou superclasses [9]. Em um Esquema RDF, as propriedades são descritas usando-se a classe RDF `rdf:Property` e as propriedades RDF `rdfs:range` e `rdfs:domain` e de um Esquema RDF `rdfs:subPropertyOf`.

Todas as propriedades em RDF são descritas usando-se a classe `rdf:Property`[6] e a propriedade de um Esquema RDF `rdfs:subPropertyOf` [7]. Logo, uma nova propriedade, como `extermos:marca`, por exemplo, é descrita nomeando-a com um URI e descrevendo-a com uma propriedade `rdf:type` cujo valor é o recurso `rdf:Property`, como é mostrado a seguir:

```
extermos:marca        rdf:type        rdf:Property .
```

Além disso, um Esquema RDF permite estabelecer restrições sobre as propriedades de um recurso. São elas: `rdfs:range` e `rdfs:domain`. A restrição `rdfs:range` é usada para indicar que os valores de uma determinada propriedade são instâncias de uma ou mais classes [9]. Em outras palavras, a restrição `rdfs:range` limita os valores que podem ser aplicados a uma determinada propriedade. O Código 18 mostra uma declaração em triplas RDF de um cenário onde a organização `exemplo.org` quer indicar que a propriedade `sve:proprietario` possui valores que são instâncias da classe `sve:VeiculoMotor`,

Código 18 Uso da propriedade `range`.

```
1 sve:VeiculoMotor        rdf:type        rdfs:Class.
2 sve:proprietario        rdf:type        rdf:Property.
3 sve:proprietario        rdfs:range      sve:VeiculoMotor.
```

A declaração do Código 18, indica que `sve:VeiculoMotor` é uma classe, que `sve:proprietario` é uma propriedade e que a declaração RDF, usando a propriedade `sve:proprietario`, possui instâncias da classe `sve:VeiculoMotor` como objetos. A restrição `rdfs:range` pode ser também aplicada para indicar que o valor de uma propriedade é um literal tipado. Por exemplo, a `exemplo.org` poderia querer indicar que a propriedade `sve:preco` tem valores do tipo de dados do XML *Schema* `xsd:integer`.

Uma propriedade pode ter zero, uma ou mais de uma propriedades com a restrição `range` [9]. Quando uma propriedade (`ex:selvagem`, por exemplo) não tem uma propriedade com a restrição `range`, então, nada é dito sobre seus valores. Se a propriedade `ex:selvagem` tem como restrição `range` a classe `ex:Carnívoro`, então, se diz que os valores da propriedade `ex:selvagem` são instâncias da classe `ex:Carnivoro`. Se a propriedade `ex:selvagem` tem mais de uma restrição `range`, ou seja, se a propriedade `ex:selvagem` tem como restrições `range` a classe `ex:Carnivoro` e a classe `ex:Predador`, então, se diz que os valores da propriedade `ex:selvagem` são instâncias das classes `ex:Carnivoro` e `ex:Predador`.

A restrição `rdfs:domain` é usada para especificar a classe na qual determinada propriedade pode ser aplicada [9]. Em outras palavras, a restrição `rdfs:domain` limita as instâncias de classes que podem utilizar uma determinada propriedade. O Código 19, mostra um cenário onde a organização `exemplo.org` quer indicar que a propriedade `sve:marca` é aplicada a todas as instâncias da classe `sve:VeiculoMotor`.

[6] Similar à noção de atributo em orientação a objetos.

[7] Denota a relação de especialização entre duas propriedades.

Código 19 Uso da propriedade `domain`.

```
1 sve:VeiculoMotor     rdf:type         rdfs:Class.
2 sve:marca            rdf:type         rdf:Property.
3 sve:marca            rdfs:domain      sve:VeiculoMotor.
```

A declaração do Código 19, indica que `sve:VeiculoMotor` é uma classe, que `sve:marca` é uma propriedade e que as declarações RDF que usem esta propriedade devem possuir instâncias da classe `sve:VeiculoMotor` como sujeito.

Uma propriedade pode ter zero, uma ou mais de uma restrição `domain` [9]. Quando uma propriedade, como `extermos:porta`, não tem uma restrição `domain`, então, nada é dito sobre quais instâncias de classes podem usar a propriedade `extermos:porta`, ou seja, qualquer instância de qualquer classe pode ter uma propriedade `extermos:porta`. Se a propriedade `extermos:porta` tem uma restrição `domain`, especificando `ex:Carro` como domínio, então é dito que a propriedade `extermos:porta` pode ser aplicada às instâncias da classe `ex:Carro`. Se a propriedade `extermos:porta` tem mais de uma restrição `domain`, ou seja, uma especificando `ex:Carro` como domínio e a outra especificando `ex:Geladeira` como domínio, então é dito que qualquer instância de uma classe que tem uma propriedade `extermos:porta`, é uma instância da classe `ex:Carro` e também da classe `ex:Geladeira`. Com estes exemplos, percebe-se que a especificação de restrições `range` e `domain` devem serem usadas com muito cuidado.

Um Esquema RDF permite a especialização de propriedades. Este relacionamento de especialização entre duas propriedades é descrito usando-se a propriedade predefinida `rdfs:subPropertyOf` [9]. Por exemplo, se `sve:motoristaProfissional` e `sve:motorista` são propriedades, poderia ser declarado que a propriedade `sve:motoristaProfissional` é uma especialização da propriedade `sve:motorista`. Esta declaração é mostrada no Código 20.

Código 20 Especialização de propriedades.

```
1 sve:motorista               rdf:type            rdfs:Property.
2 sve:motoristaProfissional   rdf:type            rdfs:Property.
3 sve:motoristaProfissional   rdfs:subPropertyOf  sve:motorista.
```

O significado do relacionamento `rdfs:subPropertyOf` pode ser visualizado da seguinte maneira: se a instância `expessoa:Pedro` é o valor da propriedade `sve:motoristaProfissional` da instância `sve:transTur`, então, um Esquema RDF define que a `expessoa:Pedro` é, também, o valor da propriedade `sve:motorista` da instância `sve:transTur`. Uma propriedade pode ser uma sub-propriedade de zero, uma ou mais propriedades. Todas as restrições `rdfs:range` e `rdfs:domain` que se aplicam a uma propriedade RDF também se aplicam a cada uma das suas sub-propriedades.

O Código 21 mostra um esquema de veículo completo, ilustrando as características de um Esquema RDF discutidas nos exemplos anteriores. Este exemplo mostra como descrever classes e propriedades usando-se um Esquema RDF. Logo, resta mostrar como as instâncias usam estas classes e propriedades. O Código 22 descreve uma instância da classe `sve:VeiculoPassageiro` juntamente com valores para suas propriedades.

Código 21 Um esquema completo de veículos em um Esquema RDF.

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

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

<rdfs:Class rdf:ID="VeiculoPassageiro">
    <rdfs:subClassOf rdf:resource="#VeiculoMotor"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Caminhão">
    <rdfs:subClassOf rdf:resource="#VeiculoMotor"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Van">
    <rdfs:subClassOf rdf:resource="#VeiculoMotor"/>
</rdfs:Class>

<rdfs:Class rdf:ID="MiniVan">
    <rdfs:subClassOf rdf:resource="#Van"/>
    <rdfs:subClassOf rdf:resource="#VeiculoPassageiro"/>
</rdfs:Class>

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

<rdfs:Datatype rdf:about=
               "http://www.w3.org/2001/XMLSchema#integer"/>

<rdf:Property rdf:ID="registradoPara">
    <rdfs:domain rdf:resource="#VeiculoMotor"/>
    <rdfs:range rdf:resource="#Pessoa"/>
</rdf:Property>

<rdf:Property rdf:ID="quantidadePassageiros">
    <rdfs:domain rdf:resource="#VeiculoMotor"/>
    <rdfs:range rdf:resource=
                "http://www.w3.org/2001/XMLSchema#integer"/>
</rdf:Property>

<rdf:Property rdf:ID="motorista">
    <rdfs:domain rdf:resource="#VeiculoMotor"/>
</rdf:Property>

<rdf:Property rdf:ID="motoristaProfissional">
    <rdfs:subPropertyOf rdf:resource="#motorista"/>
</rdf:Property>

</rdf:RDF>
```

Código 22 Descrição de uma instânciua da classe `sve:VeiculoPassageiro`.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xml:sve="http://www.exemplo.org/schemas/veiculos#"
         xml:base="http://www.exemplo.org/companhiaAviacao/">

  <sve:VeiculoPassageiro rdf:ID="TransTur">
    <sve:registradoPara
            rdf:resource="http://www.exemplo.org/idpessoal/54716"/>
    <sve:quantidadePassageiros rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#integer">
                 46  </sve:quantidadePassageiros>
    <sve:motoristaProfissional
               rdf:resource="http://www.exemplo.org/idpessoal/54716"/>
  </sve:VeiculoPassageiro>
</rdf:RDF>
```

Na próxima Subseção são apresentadas algumas outras propriedades de um Esquema RDF.

4.3 Outras propriedades

Um Esquema RDF tem outras propriedades [5] que podem ser usadas para documentação e outras informações sobre o próprio esquema ou sobre suas instâncias [9]. As principais são:

- `rdfs:comment` : pode ser usada para descrever um recurso em formato adequado para ser lido por seres humanos;
- `rdfs:label` : pode ser usada para indicar a versão do nome de um recurso em formato adequado para ser lido por seres humanos;
- `rdfs:seeAlso`: pode ser usada para indicar um recurso que contém informações adicionais a respeito do recurso em questão;
- `rdfs:isDefinedBy` : é uma sub-propriedade da propriedade `rdfs:seeAlso` que indica o recurso que define o recurso em questão.

5 Considerações Finais

O objetivo da *Web* Semântica é tentar resolver os problemas da *Web* atual, ou seja, tornar as informações mais fáceis de serem manipuladas por computadores. Segundo Berners-Lee [2], para a *Web* Semântica funcionar de forma efetiva, computadores têm que ter acesso à coleções estruturadas de informações e conjuntos de regras de inferência para que seja administrado o raciocínio automatizado, ou seja, a representação do conhecimento.

Existe um consenso de que esta representação só é possível a partir do uso de metadados (dados que fazem referência a outros dados e são destinados ao consumo das máquinas), constituindo no fator básico para promover integração e intercâmbio de informação entre fontes heterogêneas e distribuídas [10]. Contudo, para que o conhecimento possa ser representado

a partir de diferentes padrões de metadados e ainda assim ser interoperável, são necessárias arquiteturas de alto nível, capazes de acomodar essa diversidade de padrões.

O principal objetivo de uma arquitetura de metadados é representar e dar suporte ao transporte de uma grande variedade de esquemas de metadados num ambiente distribuído, provendo interoperabilidade. RDF é uma arquitetura genérica de metadados que permite descrever recursos no contexto da *Web* através de padrões de metadados, visando o processamento por máquina.

RDF permite a definição de vocabulários através de um Esquema RDF, que é uma linguagem para descrever um vocabulário RDF, permitindo hierarquias de classes e propriedades. Um Esquema RDF define a semântica para um domínio particular, embora com pouca expressividade. Portanto, RDF eleva as perspectivas de um padrão amplamente aceito para representação de informações na *Web*.

Por estas razões, se pode observar que o RDF/Esquema RDF é uma das tecnologias chave para consolidação da *Web* Semântica. Além disso, RDF utiliza a sintaxe da linguagem XML para expressar o significado dos recursos. Esta linguagem é considerada, atualmente, a linguagem mais importante para representação e troca de dados na *Web* [11].

6 Agradecimento

Ao Prof. Dr. João Carlos da Silva pela avaliação do presente texto e pelas sugestões feitas, as quais muito contribuiram para a melhoria do texto original.

Referências

[1] BECKETT, D. **RDF/XML Syntax Specification**. `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`, acessado em Junho de 2005, Fevereiro 2004.

[2] BERNERS-LEE, T; HENDLER, J; LASSILA, O. **The Semantic Web**. Scientific American, May 2001.

[3] BRAY, T; HOLLANDER, D; LAYMAN, A; TOBIN, R. **Namespaces in XML 1.1**. `http://www.w3.org/TR/2004/REC-xml-names11-20040204/`, acessado em Junho de 2005, Fevereiro 2004.

[4] BRAY, T; PAOLI, J; SPERBERG-MCQUEEN, C. M; YERGEAU, F. **Extensible Markup Language (XML) 1.0**. `http://www.w3.org/TR/2004/REC-xml-20040204/`, acessado em Junho de 2005, Fevereiro 2004.

[5] BRICKLEY, D; GUHA, R. **RDF Vocabulary Description Language 1.0: RDF Schema**. `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`, acessado em Junho de 2005, Fevereiro 2004.

[6] CARNEIRO, M. **Geração Automática de Metadados**. Projeto Final de Curso. Instituto de Informática - UFG, 2003.

[7] HAYES, P. **RDF Semantics**. `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`, acessado em Junho de 2005, Fevereiro 2004.

[8] INC., U. **Unicode Home Page**. http://www.unicode.org/, acessado em Setembro de 2005.

[9] MANOLA, F; MILLER, E. **RDF Primer**. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, acessado em Junho de 2005, Fevereiro 2004.

[10] MOURA, A. **A Web Semântica: Fundamentos e Tecnologias**. http://www.ipanema.ime.eb.br/~anamoura/publicacoes.html, acessado em Junho de 2005.

[11] W3C. **Home Page**. http://www.w3.org/, acessado em Junho de 2005.