Edgard Luiz Marx

Babel

Um framework extensível para a publicação de RDF de várias fontes de dados utilizando templates

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof.ª Karin Breitman

Rio de Janeiro, Fevereiro de 2012

Edgard Luiz Marx

Babel

Um framework extensível para a publicação de RDF de várias fontes de dados utilizando templates

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof.ª Karin Breitman
Orientador
PUC-RIO

Prof. José Viterbo Filho
UFRJ

Prof. Marco Antônio Casanova
PUC-RIO

Prof.ª Vânia Maria Ponte Vidal
UFC

Prof. José Eugênio Leal
Coordenador(a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 17 de Fevereiro de 2012

Edgard Luiz Marx

Edgard Luiz Marx é formado em administração pela escola de negócios do SEBRAE. Graduou-se em Informática pela PUC-RIO em 2006 despertando interesse pelo tema Governo Eletrônico. Desde 2005 trabalha para o laboratório de pesquisa em Computação Gráfica (Tecgraf) no desenvolvimento de Sistemas de Informações Geográficas destacando a participação no desenvolvimento cooperativo de bibliotecas de código aberto como Geotools. Em 2010 juntou-se ao grupo de pesquisas em Publicação de Dados Abertos ministrando cursos e tutorias junto ao W3C Brasil.

Ficha Catalográfica

Edgard Luiz Marx

descrição

informação técnica

natureza

Incluí referências bibliográficas.

palavra-chave

Este trabalho é dedicado a todas as pessoas que contribuíram de alguma forma para a minha formação, a minha família, aos meus amigos e mestres pela sorte de ter tido a oportunidade de aprender com vocês.

Agradecimentos

Agradeço

Resumo

Edgard Luiz Marx. **Babel – Um framework extensível para a publicação de RDF de várias fontes de dados utilizando templates.** Rio de Janeiro, 2012. xxxp. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A grande maioria dos dados que se encontram hoje na Web não estão preparados para a Web Semântica. Para facilitar e promover a conversão de dados, armazenados em bancos de dados relacionais e planilhas em particular, propomos o Babel. Diferentemente das abordagens existentes, nomeadamente RDB2RDF, Babel promove a conversão de dados em uma ampla variedade de formatos, que incluem OWL, RDFa, RSS e (X)HTML, além de RDF. A principal contribuição de Babel, no entanto, é sua facilidade de uso. Babel suaviza a curva de aprendizado, eliminando a necessidade de se familiarizar com técnicas de mapeamento complexas, que são substituídas pelo uso de templates.

Palavras-chave

RDF, Banco de dados relacionais, Plugin, Arquitetura Extensível, Template, Conversor, Linked Data, XML, RDF, Planilhas

Sumário

1
Introdução

Nos últimos anos, a Web Semântica vem se destacando como padrão para publicação e troca de informações na internet, uma vez que, oferece um framework comum que permite o compartilhamento e reutilização de dados entre usuários. Além disso, oferece tecnologias que possibilitam descrever, modelar e consultar informações. Com a adoção destes padrões, provedores de conteúdo podem publicar informações utilizando-se vocabulários específicos de domínio e *interfaces* de consulta, facilitando o acesso aos dados.

A World Wide Web Consortium (W3C) recomenda a utilização do padrão Linked Data [35] na representação de dados abertos. Este padrão baseia-se na representação de dados na forma de um conjunto de triplas RDF. Neste cenário a conversão de conjuntos de informações (esquemas de banco de dados e suas instâncias) para conjuntos de dados RDF é um passo importante.

A quantidade de dados convertidos para RDF cresce significativamente a cada dia [61][60][1]. No entanto, embora seja possível observar um crescimento exponencial no tamanho da Web como um todo, ainda há uma diferença significativa entre o crescimento da Web e o crescimento da Web Semântica [1]. Acredita-se que esta diferença seja devida à falta de ferramentas para gerar dados RDF, a partir da conversão de dados em diferentes formatos.

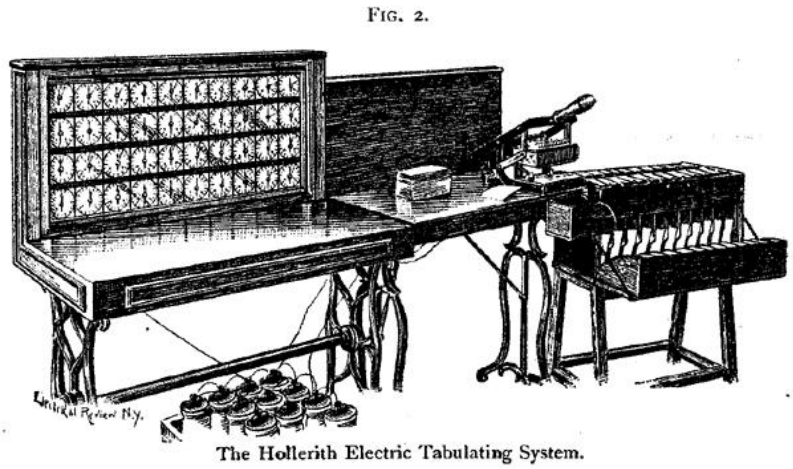

Figura 1. Sistema eletrônico de tabulação desenvolvido por Hollerith em 1989, a fim de facilitar o censo Norte Americano.

É interessante observar que a Web Semântica vem crescendo e sendo impulsionada pelo mesmo setor que apoiou o surgimento dos primeiros computadores, o setor governamental. A máquina de Hollerith, ilustrada na Figura 1, foi utilizada para auxiliar o censo Norte Americano em 1890, reduzindo o tempo de processamento de dados de sete anos, do censo anterior, para apenas dois anos e meio e ajudando o governo a economizar cinco milhões de dólares [83]. Curiosamente, os mesmos objetivos que levaram Hollerith à automatização do censo Americano, exemplificados pelo trecho abaixo, estão levando os governos à adoção de tal padrão. Novamente, cientistas estão somando forças no intuito de desenvolver mecanismos que melhorem o acesso e processamento da informação disponível.

"se o número de desempregados meses forem devidamente enumerados e compilados com referência à idade, ocupação, etc, muita informação de grande valor pode ser obtida para o estudioso de problemas econômicos que afetam nossos assalariados"[76] Hermann Hollerich, 1889

Neste trabalho, propomos Babel, um framework de apoio à conversão de dados armazenados em formatos tradicionais, e.g. bancos de dados relacionais e planilhas, para formatos semanticamente ricos, e.g., RDF-XML, RDF-Ntriples e RDFa. Mais do que um simples conversor, Babel facilita a criação de mapeamentos e a reutilização de vocabulários através de templates. Além disso,

fornece uma API única, que possibilita mesclar e publicar informações de várias fontes de dados em diferentes formatos, o que é fundamental para assegurar a adoção do padrão Linked Data.

1.1 Organização do trabalho

A dissertação foi estruturada da seguinte forma: no próximo capítulo, faremos uma revisão sobre os fundamentos básicos do framework proposto. Para aqueles familiarizados com a Web Semântica, é sugerido ir direto ao capítulo 3. No capítulo 3 motivamos a criação do framework Babel. No capítulo 4 discutimos os trabalhos relacionados. No capítulo 5 apresentamos uma nova linguagem de tempaltes, a TML. O framework é apresentado no capítulo 6. No capitulo 7, discutimos e avaliamos os resultados e, por fim, no capítulo 8, é apresentada a conclusão e trabalhos futuros.

2
Conceitos Básicos

Nesta seção abordaremos alguns conceitos que são fundamentais para o entendimento do trabalho aqui proposto.

2.1 A Web Semântica

A *World Wide Web* surgiu nos anos 90 baseada em três componentes básicos: HTTP – *HyperText Transfer Protocol*, URLs – *Universal Resource Locators* – e HTML – *HyperText Markup Language*. Esses elementos se tornaram essenciais ao compartilhamento e acesso à informação, levando a WWW a um crescimento explosivo. Grande parte desse crescimento deve-se à simplicidade do HTML, que inicialmente servia apenas para mostrar informações. Entretanto, o HTML não era extensível, pelo contrário, continha marcações específicas que requeriam o entendimento dos desenvolvedores antes que mudanças pudessem ser feitas.

O advento do *eXtensible Markup Language*, ou simplesmente XML, representou uma grande mudança. Proposta em 1996 pelo *World Wide Web Consortium* - W3C, o XML oferecia uma maneira de manipular e estruturar informação similar ao HTML, mas que poderia ser organizado com diferentes marcações, o que simplificava o processo de definição e uso de meta-informação, fornecendo extensibilidade, hierarquia e formatação.

Entretanto, problemas como usabilidade ainda persistiam: humanos ainda encontravam dificuldade em acessar o conteúdo Web, o que se agravava em processos automatizados. Estes problemas eram gerados porque a Web foi concebida para ser utilizada por humanos, o HTML foi concebido para o layout, tamanho, cor e outros requisitos de apresentação apenas.

Além disso, é notório o crescimento no uso de imagens na apresentação de informação. Usuários humanos podem facilmente interpretar esta informação, mas não se pode dizer o mesmo de processos automatizados, realizados por meio de agentes e *crawlers*, sem falar em usuários detentores de restrições cognitivas.

O final dos anos 90 determinou o início da mudança na forma de publicar a informação na Web. Os esforços começaram a se concentrar na busca da sua compreensão, o que se tornou um desafio conhecido hoje como Web Semântica.

"The goal of Semantic Web research is to transform the Web from a linked document repository into a distributed knowledge base and application platform, thus allowing the vast range of available information and services to be more effectively exploited." [44]

Os principais responsáveis pela sua popularização: Tim Beners-Lee, Hendler e Ora Lassila, descreveram sua visão da Web Semântica em um artigo científico em 2001 [43], a qual reproduzimos a seguir:

"To date, the World Wide Web has developed most rapidly as a medium of documents for people rather than of information that can be manipulated automatically. By augmenting Web pages with data targeted at computers and by adding documents solely for computers, we will transform the Web into the Semantic Web. Computers will find the meaning of semantic data by following hyperlinks to definitions of key terms and rules for reasoning about them logically. The resulting infrastructure will spur the development of automated Web services such as highly functional agents. Ordinary users will compose Semantic Web pages and add new definitions and rules using off-the-shelf software that will assist with semantic markup."

Para a concretização dessa visão, foi necessário adicionar novas camadas de linguagens de marcação adequadas à arquitetura Web. Na Figura 2, reproduzimos o diagrama proposto por Beners-Lee, onde não só é possível visualizar as tecnologias, como também os desafios por trás da Web Semântica 2000 [45].

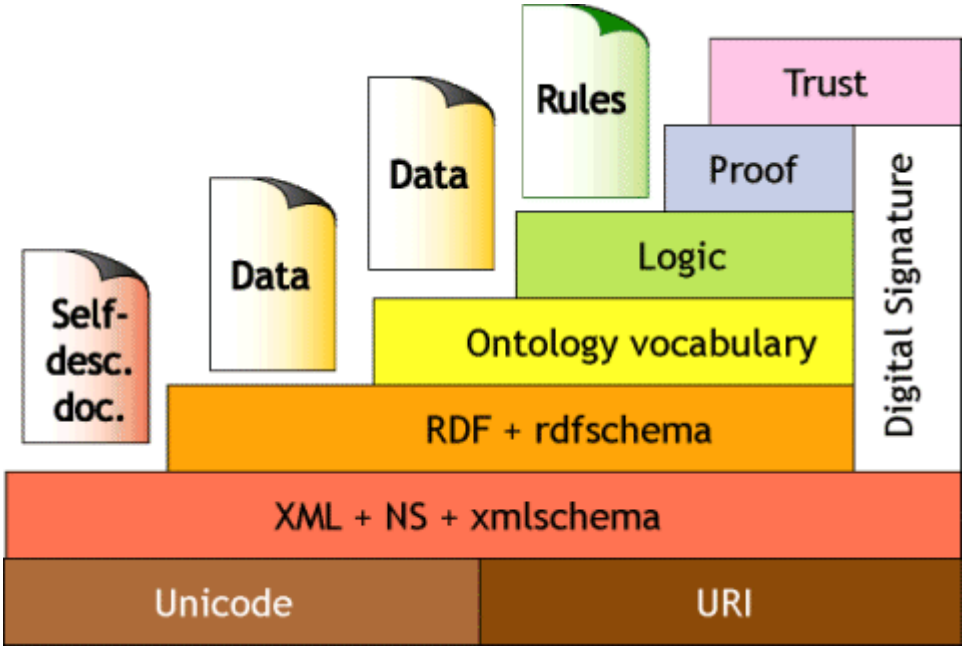

Figura 2. O "*bolo de aniversário*" da Web Semântica proposta por Berners-Lee

A camada base é formada pelo Unicode (um padrão utilizado pela indústria na representação digital da linguagem humana, símbolos e scripts) e a URI (*Uniform Resource Identifier)* que provê um meio de localizar e identificar recursos na Web. A seguir, na próxima camada, encontra-se o XML, *Name Space* e o XML *Schema*, mecanismos criados para capacitar a validação do documento XML. O uso de XML em uma arquitetura Web é de imensa importância, uma vez que ele esta presente em muitos padrões como, por exemplo, no SOA - *Service Oriented Archictecture*.

Em seqüência, a camada composta pelo RDF e o RDF Schema, que possibilita a representação semântica da informação. O RDF - Resource Description Framework - é uma notação baseada em lógica de descrição que descreve a informação por meio de meta-informação através de triplas, onde a informação é representada através do sujeitos, predicados e o objetos. O sujeito e o objeto funcionam como adjetivos, ou "coisas" que precisam ser descritas através de URIs. O predicado tem a função de um verbo, que descreve o relacionamento entre o sujeito e o objeto, e é geralmente expresso em sintaxes como *sameAs* ou *isPartof*. Em termos da Teoria dos Grafos, podemos representar um conjunto de triplas em RDF em um grafo dirigido, onde os sujeitos e objetos são nós e os

predicados são arestas. O RDF *Schema*[1] estende o RDF, adicionando maior semântica à informação como domínio, subclasses e subpropriedades.

À medida que camadas vão sendo adicionadas na arquitetura ilustrada na Figura 2, mais descrição e formalismo lógico são adicionados. A camada de Ontologia, por exemplo, provê mais "meta" informação, como transitividade, unicidade, ambigüidade, cardinalidade dentre outras. Finalmente, nas últimas camadas da arquitetura, a lógica é utilizada para mediar a heterogeneidade, e o difícil desafio de determinar a confiabilidade da informação.

2.2 RDF [46] [47] [48] [49]

Como dito anteriormente, a WWW foi originalmente concebida para a utilização humana. Embora toda informação contida nela possa ser lida por máquinas, esta informação não é interpretada automaticamente. Isso ocorre por que na Web existe uma infinidade de informações originadas de diferentes fontes, modeladas de diferentes formas, e publicadas com diferentes protocolos. O RDF, em particular, permite descrever esta informação através da adição de meta-informação, o que facilita a interoperabilidade entre as aplicações que realizam troca de informações e o processamento automatizado de dados e recursos, sendo empregado de diversas formas, como por exemplo: para encontrar recursos, melhorar as máquinas de busca; na catalogação, para descrever conteúdos, disponível em Sites ou páginas; por agentes de software, para facilitar o compartilhamento e troca de conhecimento; dentre outras.

O objetivo amplo do RDF é definir mecanismos para descrever recursos de maneira que não façam menção sobre um domínio em particular, nem que definam, em um primeiro momento, a semântica de uma aplicação. Em outras palavras, a definição do mecanismo deve ser neutra de domínio, de forma que o mecanismo seja adequado para descrever a informação de qualquer domínio.

A base do RDF consiste em um modelo para representação de propriedades e seus valores. O modelo RDF foi desenhado em princípios bem estabelecidos na representação da informação de várias comunidades. Uma propriedade RDF pode ser entendida como um atributo de um recurso, o que, neste contexto, corresponde

[1] http://www.w3.org/TR/rdf-schema/

ao tradicional par atributo-valor. Propriedades RDF também podem representar o relacionamento entre recursos, dessa forma, um modelo RDF pode se assemelhar a um diagrama de entidade-relacionamento.

O Framework de Descrição de Recursos – RDF – é uma forma neutra de sintaxe, utilizado para estimar a equivalência de significado entre expressões, o que determina que: duas expressões RDF só serão equivalentes se e somente se seus modelos de representação forem os mesmos. Esta definição de equivalência permite a ocorrência de variação de sintaxe sem que ocorra a alteração do significado. Dessa forma, RDF pode ser escrito de várias maneiras, a exemplo do RDF/XML, triplas e RDFa.

Intrinsecamente, uma declaração RDF representa um grafo rotulado direcionado. Assim sendo, podemos representar a sentença *"Existe uma pessoa cujo nome é Edgard"* na forma de grafo, como ilustrado na Figura 3.

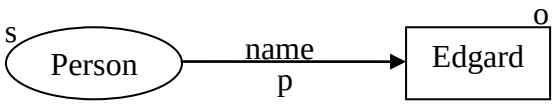

Figura 3. Representação simples de uma declaração RDF na forma de grafo.

Como podemos perceber, na Figura 3, o sujeito é representado pela classe *Person* e o predicado é um atributo da classe, e o objeto o valor (Edgard). Em RDF, o sujeito e o predicado são recursos; o objeto pode ser um literal ou outro recurso. No caso específico ilustrado pela Figura 3, o valor "Edgard" trata-se de um literal do tipo *String*.

A forma mais comum de se escrever RDF é através da *Extensible Markup Language,* ou simplesmente XML, que assim como o RDF, trata-se de uma recomendação da W3C para se escrever e estruturar a informação [47]. Além disso, XML é usado em vários protocolos de comunicação, como: SOAP [2]–*Simple Object Acess Protocol*; SPARQL [26]; ou na publicação da informação como em (X)HTML[3] [4]e RSS[5]. Como visto na seção anterior, XML consiste na camada base

[2] http://www.w3.org/TR/soap/

[3] http://www.w3.org/TR/xhtml1/

[4] http://www.w3.org/html/

da Web Semântica. Existem várias razões para se utilizar XML na publicação de informação, mas talvez a mais significativa, seja porque permite tornar a informação compreensível tanto pra homens quanto para máquinas. A seguir, na Listagem 1, apresentamos um RDF correspondente à declaração contida no grafo da Figura 3, serializado na sintaxe RDF/XML.

```
<Person namespace="http://xmlns.com/foaf/0.1/">
      <name>Edgard</name>
<Person/>
```

Listagem 1. Exemplo simples de RDF em XML utilizando o vocabulário FOAF[6].

Outra maneira de se representar RDF é através de triplas. Chamamos de triplas a forma de representar os nós na forma *{p, s, o}*, onde *p* é o predicado, *s* o sujeito e *o* o objeto. Dessa forma, a representação do grafo da Figura 3 no formato de triplas seria: *{name, Person, Edgard}*. Triplas são comumente encontradas nos chamados *triplestores*, também conhecidos como banco de dados nativos RDF, que nada mais são que banco de dados que armazenam RDF no formato de triplas: N3 visto anteriormente na forma sujeito, predicado e objeto; e no formato N4, que adiciona mais uma informação à tripla, o contexto [99].

Por fim, RDF também pode ser embutido em páginas (X)HTML através de RDFa. RDFa define um sintaxe de mapeamento RDF para um número de atributos (X)HTML, mas pode ser facilmente importada para outras linguagens baseadas em XML. A seguir, na Listagem 2, apresentamos um exemplo da declaração da Figura 3 escrita em RDFa

```
<div xmlns:foaf="http://xmlns.com/foaf/0.1/">
   <h2 property="foaf:name">Edgard</h2>
</div>
```

Listagem 2. Exemplo simples de RDFa embutido em uma página utilizando o vocabulário FOAF.

[5] http://feed2.w3.org/docs/rss2.html

[6] http://xmlns.com/foaf/spec/

2.3
R2RML [50] [51] [52]

R2RML é uma linguagem desenvolvida para criar mapeamentos customizados e foi proposta pelo grupo de trabalho RDB2RDF, como proposta de padronização da linguagem de mapeamento das ferramentas de conversão de banco de dados relacionais para RDF. Com R2RML, é possível materializar dados relacionais na forma RDF, estruturados e mapeados para vocabulários definidos pelo usuário. R2RML é escrito através da sintaxe Turtle [15] e, é por si só um grafo RDF, o que possibilita escrever uma sentença de diferentes formas.

Baseado no *Survey of Current Approaches for Mapping of Relational Databases to RDF*, preparado pelo grupo RDB2RDF [14], o escopo do R2RML é fundamentado nos seguintes princípios: definir o mapeamento de dados relacionais e esquemas relacionais para RDF e OWL; possuir uma sintaxe legível, bem como possuir representação em RDF e XML para possibilitar a geração e leitura por máquina; possibilitar o suporte a tipos específicos de dados SQL de diferentes fornecedores, e permitir o mapeamento de um mecanismo para criar identificadores de entidades de banco de dados.

2.3.1
Visão Geral [53]

Um documento de mapeamento R2RML consiste em uma ou mais estruturas chamadas *TriplesMaps*. Cada *TriplesMap* contém uma referência a uma tabela lógica do banco de dados relacional de entrada. A tabela lógica pode ser definida por uma tabela, uma *view*, ou simplesmente uma consulta.

Além disso, um *TriplesMap* contém as regras para o mapeamento das entradas da Tabela lógica para um conjunto de triplas RDF. Essas regras são constituídas por uma estrutura *SubjectMap* e uma ou mais estruturas *PredicateObjectMap* (s).

As triplas RDF geradas a partir de uma tupla compartilham o mesmo sujeito. A estrutura *SubjectMap* em um *TriplesMap* contém as regras para gerar o sujeito de uma tupla.

Cada estrutura *PredicateObjectMap* em um *TriplesMap* contém as regras para a geração de um par – objeto, predicado – a partir dos valores na linha da Tabela. Ele consiste de uma estrutura *PredicateMap* e uma estrutura *ObjectMap*.

Um *TriplesMap* é usado para gerar triplas RDF através das entradas da Tabela lógica de um banco de dados relacional, combinando o sujeito, gerado usando o *SubjectMap*, com o(s) par(es) – objeto, predicado(s) – gerado(s) usando o(s) *PredicateObjectMap*(s). A seguir, na Listagem 3, apresentamos um exemplo de mapeamento simples.

```
<#TriplesMap1>
    a rr:TriplesMapClass;
    rr:SQLQuery """
       Select "name"
         from person
       """;

    rr:subjectMap [ rr:class foaf:person;];

    rr:predicateObjectMap
    [
      rr:predicateMap [ rr:predicate foaf:name ];
      rr:objectMap    [ rr:column "name"; rr:datatype
xsd:String];
.
```

Listagem 3. Exemplo de um mapeamento do atributo *name* de uma Tabela ralacional *person* para o vocabulário FOAF utilizando R2RML.

2.4 Templates

O termo template, em Ciências da Computação, é usado para designar os documentos que têm uma estrutura pré-definida e são utilizados como ponto de partida para a criação de novos documentos, de modo que a estrutura não necessite ser recriada quando for reutilizada. Outra definição encontrada nos dicionários Cambridge[7] e Oxford[8] define um template como algo que serve como um modelo para produção de artefatos similares.

Templates vêm sendo usados há muito tempo, nas mais diversas áreas do conhecimento: Matemática, Biologia, Química, dentre outras. O uso de templates

[7] http://dictionary.cambridge.org

é notoriamente interessante por dois aspectos: primeiro, são genéricos e reutilizáveis. Templates não são uma linguagem específica e sim um tipo de estrutura, em geral um documento, com modelos e padrões que podem ser transcritos alterando-se apenas os valores, característica essa, especialmente útil na criação de novos documentos; segundo, templates possibilitam a divisão de expertise entre dois grupos: aqueles que criam o template, e aqueles que inserem os dados, uma vez que o conhecimento utilizado na criação do template é dispensável na hora de realizar o mapeamento dos dados e vice-versa, possibilitando tanto a sintetização do conhecimento utilizado na criação do documento quanto em seu preenchimento.

Infelizmente, a importância prática e teórica de se usar templates como vantagem competitiva, bem como evidências empíricas e sistemáticas de sua utilização são escassas na literatura, talvez pela ausência de métricas aceitas para validação do uso dos mesmo.

Em um estudo publicado em 2006, Gabriel Szulanski e Robert J. Jansen [42] avaliaram o uso de templates aplicado a rotinas organizacionais. Nelson e Winter [58] usaram o termo template para se referir aos exemplos de rotinas organizacionais, que em sua concepção, contêm aspectos críticos e não críticos da rotina, fornecendo os detalhes e nuanças do trabalho, em que seqüência, e de como vários componentes e sub-rotinas são interligadas. Segundo os autores, alavancar ativos de conhecimento através da replicação de rotinas da envolve recriar conhecimento produtivo do local de origem e facilita a transferência de conhecimento. O estudo realizado por Gabriel Szulanski e Robert J. Jansen foi aplicado em 15 países europeus, durante oito anos na Xerox Europa, e revelou que a adoção de templates leva a uma transferência eficaz de conhecimento.

2.5 XML [54]

XML foi desenvolvido por um Grupo de Trabalho XML (originalmente conhecido como o Conselho de Revisão Editorial SGML), formado sob os auspícios da *World Wide Web Consortium* (W3C) em 1996.

---

[8] http://oxforddictionaries.com

A eXtensible Markup Language (XML) está se tornando rapidamente o padrão de fato para troca de informações na Web, sua adoção está levando ao surgimento de um novo conjunto de requisitos à manipulação da informação, tais como a necessidade de armazenar e consultar documentos XML. XML é uma forma restrita de SGML ou Standard Generalized Markup Language [ISO8879], composta de unidades de armazenamento chamadas entidades - *Entities*. As entidades contêm ambos os dados, interpretáveis ou não. Dados interpretáveis são compostos de caracteres, alguns dos quais formam caracteres de informação, e alguns dos quais formam a estrutura como a marcação do formulário. A marcação codifica uma descrição da organização e a estrutura lógica de armazenamento do documento, que pode possuir restrições através de mecanismos (DTD, XMLSCHEMA).

O XML apresenta algumas vantagens. Por exemplo, é auto-descritivo – a marcação descreve a estrutura e nomes de tipo de dados, embora não a semântica, é portátil – Unicode, e pode descrever os dados em estruturas de árvore ou Gráfico. Ele também possui desvantagens, por exemplo, é prolixa e o acesso aos dados é lento devido à conversão e análise de texto. Dentre as principais interfaces de programação estão DOM [55] e SAX [56] que serão detalhadas a seguir.

2.5.1 APIs DOM e SAX [55, 56]

A definição de um conjunto de interfaces de programação independente da linguagem, que possibilita o acesso e a manipulação de documentos, tornaram a manipulação do XML mais fácil para os programadores. O XML não só explora a necessidade de uma codificação de informações e formato padrão de armazenamento, mas também permite aos programadores escolher uma forma de manipulá-la. Atualmente, exitem duas APIs – Application *Programming Interface* – principais que definem maneiras distintas de se manipular os documentos XML: SAX e DOM.

A especificação SAX – *Simple API for XML*, por outro lado, define uma abordagem baseada em eventos em que *parsers* navegam através das informações contidas no XML, chamando funções de manipulação sempre que determinadas

partes do documento - por exemplo, nós de texto ou instruções de processamento - são encontradas.

O sistema SAX – Simple API for XML – é baseado em eventos, onde o interpretador não cria uma representação interna do documento. Em vez disso, o interpretador chama funções de manipulação quando determinados eventos, definidos pela especificação SAX, ocorrererem. Estes eventos incluem a identificação do início e do final do documento; encontrar um nó de texto; encontrar elementos filhos ou um elemento mal formado, o que possibilita ao usuário personalizar ou criar seu próprio modelo de tratamento de eventos.

A especificação DOM – *Document Object Model* – define uma abordagem de navegação do documento XML baseada em árvore. Em outras palavras, um *parser* DOM processa dados XML e cria uma representação orientada a objetos de forma hierárquica, que pode ser percorrida em tempo de execução.

Diferentemente da API SAX, a API DOM cria uma árvore interna baseada na estrutura hierárquica dos dados contidos no arquivo XML, que permanece na memória até que seja liberada. DOM usa funções que retornam os nós, pai e filho, dando-lhes pleno acesso aos dados – estrutura, hirarquia, informações – contidos no XML. A manipulação do documento XML através de DOM é simples, e sua a API é de fácil entendimento.

3
Motivação

Nos últimos anos verificou-se um aumento de interesse em ferramentas de apoio ao processo de publicação de dados na Web, principalmente na esfera governamental[9][10]. Segundo Tim Berners-Lee, a busca na publicação de informações governamentais é motivada basicamente por três razões [74]:

- *Aumentar a consciência cidadã nas funções do governo para permitir o engajamento;*
- *Contribuir com informações valiosas sobre o mundo;*
- *Permitir que o governo, o país e o mundo funcionem com maior eficiência.*

No entanto, a dificuldade para utilização dos dados governamentais demonstrou que publicá-los na Web era muito mais que colocar informações em sites, para que pudessem ser acessadas publicamente. Embora o padrão Linked Open Data tenha demonstrado ser um caminho importante para as instituições governamentais, como veremos, ainda existem barreiras que impedem sua difusão.

Para Auer [1], a razão principal é a existência de ferramentas que se perdem na complexidade da geração de mapeamentos para banco de dados, por três razões:

- *Identificação de dados privados e públicos;*
- *Uso apropriado dos vocabulários existentes;*
- *Perda da descrição original do esquema do banco de dados.*

[9] www.data.gov.uk – site de publicação de conjunto de dados do governo britânico.

[10] www.data.gov – site de publicação de conjunto de dados do governo norte americano.

Butler [75] sintetiza os obstáculos para a adoção dos padrões da Web Semântica como sendo:

- *A complexidade do RDF/XML;*
- *Uso indevido de ferramentas complexas;*
- *Suporte a múltiplas versões de vocabulários;*
- *Suporte a múltiplos vocabulários;*
- *Simplificar a criação, validação e processamento da meta-informação da Web Semântica;*
- *Ocultar a complexidade da Web Semântica de usuários leigos;*
- *Prover técnicas de mapeamento entre múltiplos vocabulários e versões;*
- *Padronização de Vocabulários;*
- *Problemas de imprecisão ou inconsistência.*

Também podemos incluir outros obstáculos, tais como a maneira *bottom-up* de adoção de ontologias: a grande maioria das ontologias é criada por usuários que muitas vezes não utilizam vocabulários existente. Este problema seria minimizado utilizando um padrão de adoção *top-down*, no qual a criação e a padronização de ontologias fosse centralizada; um último problema é a falta de adoção do padrão pela indústria: muitas ferramentas foram projetadas para servirem de apoio à pesquisa, em detrimento a aplicações comerciais, o que leva ao surgimento de algumas barreiras que ainda restringem a sua utilização mais ampla.

As barreiras para difusão da Web Semântica se tornam ainda mais evidentes quando se comparamos a diferença existente entre o crescimento da Web propriamente dita e a Web Semântica, que pode ser constatada nos índices provenientes do Swoogle [11] [17] e do World Wide Web Size [16].

A World Wide Web Size é um site criado para estimar o tamanho da WWW baseado no número de páginas indexadas pelas máquinas de busca Google, Bing, Yahoo Search e Ask. O índice é uma estimativa, calculada a partir da subtração

[11]Swoogle - um mecanismo de busca para documentos semânticos – ontologias – na Web.

da sobra dos índices das quatro máquinas de busca obtidos em seqüência. São apresentados dois índices, dentre os possíveis: um partindo do Yahoo – YGBA – e o outro partindo do Google – GYBA.

O Gráfico 1 demonstra que, embora o crescimento da Web Semântica tenha sido grande nos últimos anos, o número de documentos indexados pela ferramenta de busca Swoogle ainda é tímido. A comparação do número de documentos encontrados pelo Swoogle com o número de documentos encontrado pelo World Wide Web Size, utilizando o índice GYBA durante os dois últimos anos, sugere que o crescimento da Web Semântica vem assumindo um comportamento linear em comparação ao crescimento da Web, que vem tendo um comportamento exponencial.

Em Janeiro de 2011, o número de documentos na Web, estimado pelo site World Wide Web Size, utilizando-se o menor índice encontrado em Janeiro de 2011, foi de aproximadamente 18.86 bilhões. Em contraste, o Swoogle indexou 1.32 milhões de documentos e 1.11 bilhões de triplas no mesmo período, uma média de 840 triplas por documento indexado. Analisando o estado atual dos *datasets* da Linked Open Data é possível encontrar um total de 35 bilhões de triplas [60]. Por analogia, podemos estimar que estas triplas estejam distribuídas em aproximadamente 42 milhões de documentos; mais ainda, supor que a maioria dos documentos publicados no formato RDF está concentrada em grandes *datasets*, a exemplo dos conjuntos de dados governamentais que detém 40% de toda a informação publicada na LOD [61].

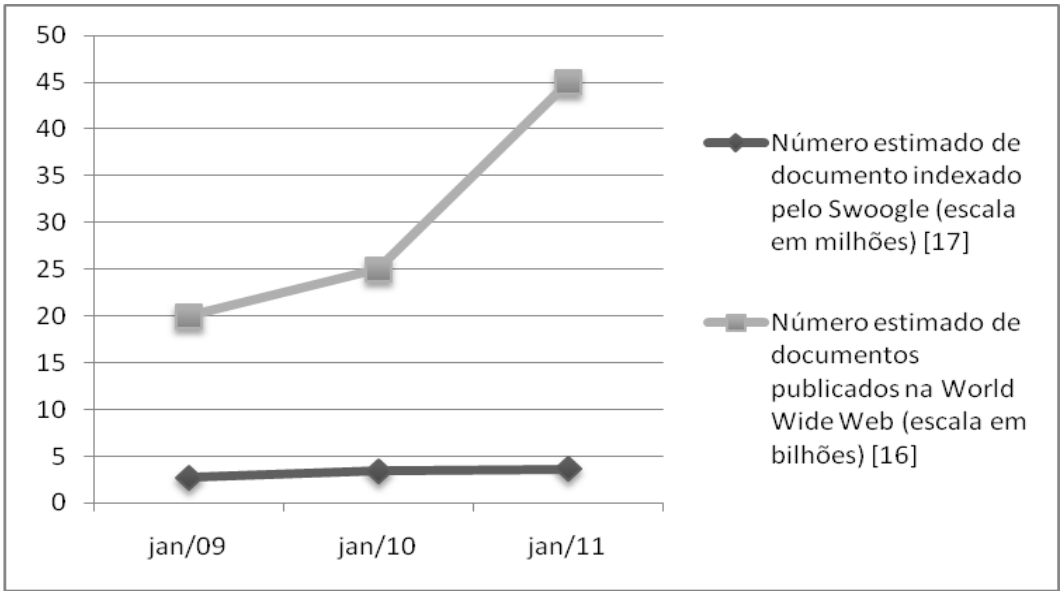

Gráfico 1. Crescimento estimado Web e da Web Semântica nos últimos dois anos, utilizando o índice GYBA do World Wide Web Size e o índice de documentos indexados pelo Swoogle.

Também é possível verificar, na Tabela 1, que uma pequena parcela da informação publicada na LOD está interligada, cerca de 11,5%. De todos os conjuntos de dados publicados, destaca-se o conjunto de Ciências da Vida, que possui um montante de 38% de dados interligados, mas, no entanto, grande parte deste conjunto está restrita a dados genéticos como grandes cadeias de DNA. Dessa forma, ainda não é possível afirmar qual será o futuro da Linked Data, os vários movimentos e técnicas de publicação de dados ainda não conseguiram um volume considerável de interligação, o que nos faz questionar a sua possibilidade real. De qualquer forma, mesmo que não seja possível a publicação total dos dados de uma forma interligada, os formatos da Web Semântica permitem que eles possam ser utilizados e interligados por terceiros.

Domínios	Número de Conjunto de Dados	Triplas	%	Links Externos	%	% Links Externos na LOD
Mídia	25	1.841.852.061	5,82	50.440.705	10,01	0,582582
Geografia	31	6.145.532.484	19,43	35.812.328	7,11	1,381473
Governo	49	13.150.009.400	42,09	19.343.519	3,84	1,616256
Publicações	87	2.950.720.693	9,33	139.925.218	27,76	2,590008
Conteúdo entre domínios	41	4.184.635.715	13,23	63.183.065	12,54	1,659042
Ciências da Vida	41	3.036.336.004	9,6	191.844.090	38,06	3,65376
Conteúdo gerado por usuário	20	134.127.413	0,42	3.449.143	0,68	0,002856
	295	31.634.213.770		503.998.829		11,485977

Tabela 1. Número de conjunto de dados distribuídos por domínio na LOD [61] em Setembro de 2011.

Outro dado nos mostra que a LOD não é tão grande como imaginamos. A DBPedia[12] contém 1 bilhão de triplas distribuídas em 22 *gigabytes*[13] no formato N-Triples e N-Quad. Sabendo que há duplicidade de informação, ou seja, cada declaração RDF no formato N-Triples está reescrita no formato N-Quad, podemos estimar que o tamanho real da DBpedia é de aproximadamente 11 GB, o que daria à LOD um tamanho total de 361 gigabytes. Segundo esta constatação, a LOD, poderia ser armazenada em qualquer computador pessoal, o que nos mostra certa fragilidade.

Além disso, Auer [1] constatou que a maioria dos termos indexados pelas ferramentas de busca como Swoogle, estão em RDF, RDFS, OWL e se utilizam de vocabulários RDF populares tais como FOAF, DC, RSS.

[12] www.dbpedia.org – Trata-se de um dos maiores e principais *datasets* da LOD que é um esforço comunitário para extrair os dados da Wikipedia e torná-los acessíveis.

[13] http://downloads.dbpedia.org/3.7/ - D*ataset* da DBpedia 3.7 disponível para download.

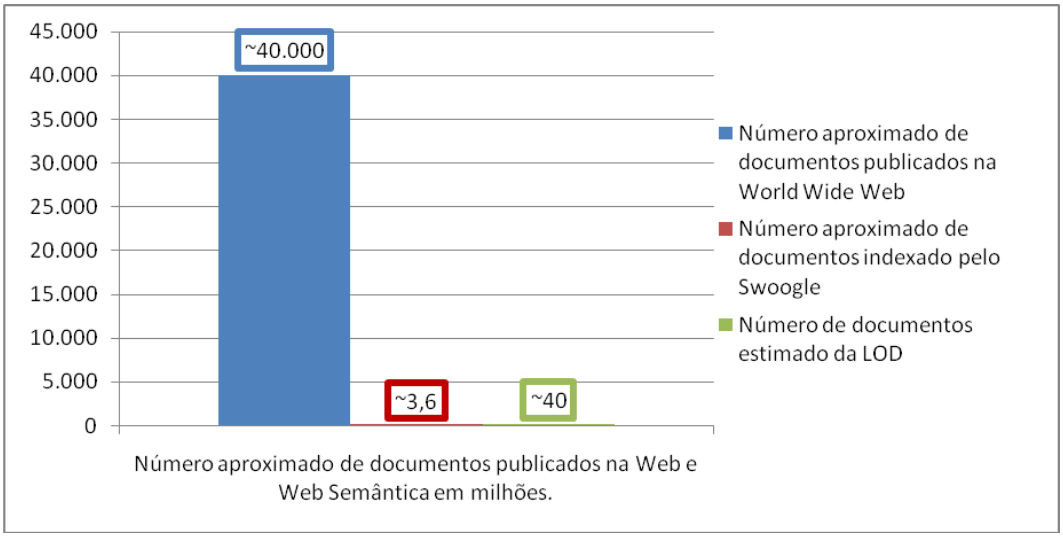

Gráfico 2. Comparação entre o tamanho da World Wide Web e a Web Semântica em 2011 utilizando os índices aproximados do World Wide Web Size e Swoogle encontrados em Janeiro de 2011, com o tamanho estimado de documentos na LOD (Uma estimativa utilizando a relação do número de triplas por documento extraído do Swoogle e aplicado à LOD).

Outro aspecto que merece nossa atenção é o desempenho das ferramentas de publicação de Linked Open Data, muito inferior a padrões e aplicações já estabelecidos como banco de dados relacionais. Há muitos trabalhos que recomendam a utilização de banco de dados relacionais para armazenar e consultar informações em RDF [78][79][80][81]. Embora haja muitos estudos envolvidos na transformação da álgebra SPARQL em SQL [78][79], ainda não há uma maneira definitiva e estabelecida para a realização desse processo. Dentre os fatos que impedem uma transformação eficiente da álgebra SPARQL para SQL podemos destacar a diferença de expressividade contida nas linguagens, a álgebra SPARQL é muito mais rica que a álgebra SQL [102]. Os resultados de um teste, para avaliar o desempenho das aplicações, ilustradas pelo Gráfico 3, sugeriu que conversores simples possuem um desempenho muito superior ao apresentado pelas ferramentas que manipulam o grafo RDF, o que leva a acreditar que, uma vez que as questões envolvendo a conversão de SPARQL para SQL sejam resolvidas, o desempenho destas aplicações deve melhorar. A seguir reproduzimos comentários de pesquisadores da área que servem para ilustrar este fato.

“Unfortunately, the join rule stated above does not fully reproduce SPARQL semantics"

Richard Cyganiak [78]

“a more sophisticated algorithm is required to express nested optional graph patterns"

Stephen Harris [79]

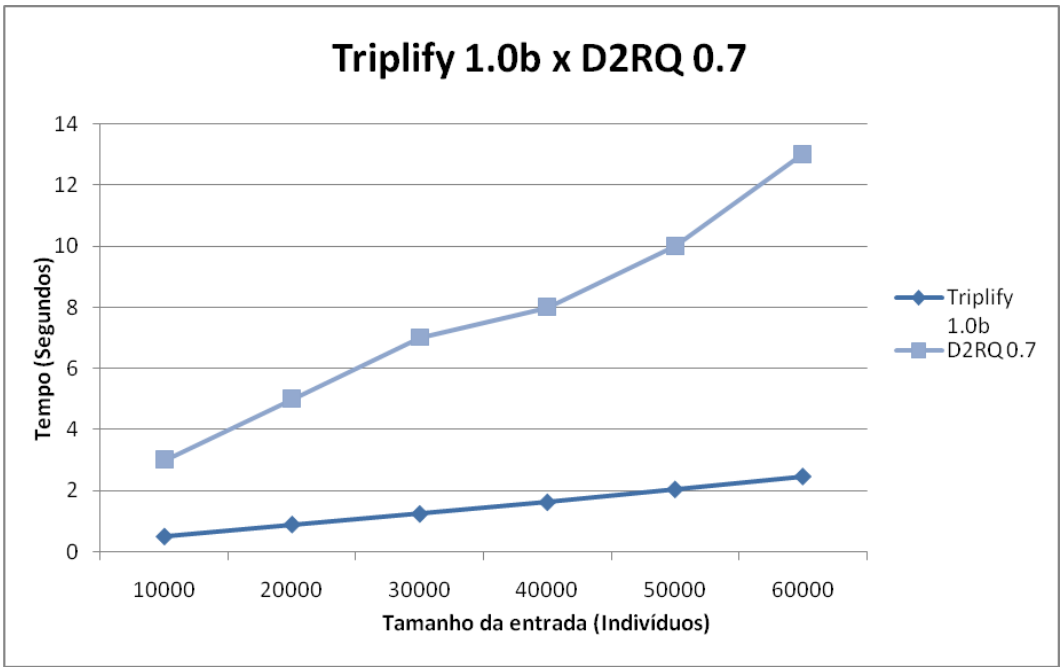

Gráfico 3. Comparação de desempenho entre Triplify 1.0b e D2RQ 0.7 na geração de indivíduos do tipo FOAF contendo apenas um atributo nome.

Por outro lado, as informações provenientes do Euro Semantic Web [11], demonstram que poucas destas ferramentas são, de fato, extensíveis. Os 73 conversores catalogados dão suporte a 49 tipos de formatos distintos. Em resumo, há uma ferramenta e meia para cada formato, o que é um número relativamente baixo. A ferramenta que é mais extensível apresentou suporte a cinco formatos, e a menos extensível apenas a um. Bergman [18] estima que a combinação de dados disponíveis na Deep Web e Surface Web seja superior a meio trilhão de documentos. Os números indicam a necessidade premente de ferramentas, métodos e técnicas para aprimorar e melhorar o acesso a tais informações, um processo também conhecido como *surface* [72].

Acredita-se que a falta de ferramentas que facilitam a conversão de dados armazenados em bancos de dados relacionais, planilhas e outras fontes, é um fator de impacto na publicação de conteúdo semanticamente enriquecido na Web.

Apesar do vasto número de soluções que suportam a conversão de diferentes tipos de dados em RDF – Triplify [1], D2RQ [2] e Virtuoso [5] [6] – ainda há problemas que inibem o desenvolvimento da Web Semântica. Algumas soluções [1] mostram uma mudança de atitude por parte dos pesquisadores, tornando o processo de conversão mais simples e acessível para os usuários, embora, em sua grande maioria, possam ser notadamente constatados problemas de extensibilidade, escalabilidade ou mesmo usabilidade. As principais ferramentas de conversão de dados RDB em RDF [1,2,5,6] não utilizam a mesma linguagem de mapeamento. É preciso simplificar e profissionalizar o processo. Algumas iniciativas vêm sendo tomadas para o estabelecimento de padrões, a exemplo do RDB2RDF [87], movimento que visa estabelecer um padrão para o mapeamento de dados no formato relacional para o RDF, no entanto, ainda há muito que ser feito.

Recentemente a W3C estabeleceu o padrão RDFa [19], que permite instanciar declarações RDF através de anotações de documentos HTML. Desde então, surgiram várias máquinas de busca [20] capazes de indexar páginas em RDFa. Apesar do aparecimento de várias ferramentas para conversão de dados em RDF, pouco ou nada tem sido feito para garantir o apoio para RDFa. Além disso, também não pode ser esquecida a existência de dados em outros formatos, como planilhas e formatos proprietários.

A necessidade de oferecer suporte a outros formatos além de RDB – *Relational Databases*, aliada à dificuldade de usar e estender as ferramentas existentes, causou o aparecimento de um número significativo de soluções voltadas, sobretudo, à solução de problemas específicos [11]. Além disso, essas soluções têm se mostrado um tanto ineficazes, ou até mesmo inacessíveis para a maioria dos usuários, pois apresentam muitas questões a serem resolvidas, tais como:

- Extensibilidade: muitas ferramentas são criadas para atender a uma funcionalidade específica, demonstrando ser uma verdadeira caixa preta aos usuários que desejam customizá-las ou estendê-las. Em alguns casos, o usuário é obrigado a modificar o código fonte enfrentando problemas de linguagem, suporte e documentação;

- Escalabilidade: algumas ferramentas não garantem robustez na hora de operar sob uma grande quantidade de informações, outras ferramentas não apresentam um desempenho adequado para aplicações profissionais;
- Facilidade de uso: um grande obstáculo para os usuários de maneira geral é a dificuldade de conFigurar ou operar essas ferramentas. Muitas vezes é preciso aprender linguagens ou, utilizar ferramentas adicionais para atingir o resultado esperado.

É preciso difundir a Web Semântica e fomentar o surgimento de ferramentas mais amigáveis e extensíveis, voltadas para não-especialistas, possibilitando a imersão desses profissionais nesse novo universo para promover um crescimento considerável dos documentos publicados nesse formato.

4
Trabalhos Relacionados

Existem várias ferramentas de conversão de diferentes fontes de dados em RDF [11]. As principais se concentram na conversão de informações de banco de dados relacionais. Outras ferramentas permitem a manipulação do grafo RDF, o armazenamento e consulta de dados. Nesta seção faremos uma revisão das principais tecnologias para conversão de dados para RDF e XML e na utilização de templates. A fim de facilitar o entendimento, elas serão agrupadas em quatro grupos: ferramentas de conversão; frameworks de armazenamento, publicação e manipulação; ferramentas de desenvolvimento Web baseadas em template e outras tecnologias.

4.1
Ferramentas de conversão RDF

4.1.1
Triplify

Triplify [1] foi desenvolvido para tornar simples a conversão e publicação de informações contidas em bancos de dados relacionais – RDB – em RDF. Para realizar a conversão, é preciso definir um mapeamento que envolve consultas SQL onde as Tabelas são mapeadas para classes e as colunas para atributos. Os dados mapeados podem ser acessados através de requisições HTTP. Triplify possui um bom desempenho e simplicidade de uso em comparação às demais ferramentas e por isso é largamente utilizada na geração de conteúdo em RDF e Linked Data.

4.2
Wrapper Babel

Babel [82] é um Wrapper que faz parte da suíte de aplicações SMILE do MIT, que tem o mesmo nome do framework aqui descrito. Babel oferece suporte a

conversão de diversos formatos (Excel, Exhibit JSON, Exhibit-embeddin Web Page, JPEG, N3, RDF/XML e informações delimitadas por tab) em RDF, JSON e RSS.

O Wrapper Babel foi originalmente concebido para facilitar a geração de conteúdo para o framework Exhibit[14] e, assim como todos os Wrappers e conversores que não possuem mapeamento, realiza uma simples traduções das estruturas e valores dos arquivos fontes, através de um mapeamento direto. Isso não permite uma flexibilidade na definição do esquema a ser utilizado e, conseqüentemente, prejudica a reutilização de vocabulários.

4.2.1 RDF123

RDF123 [95] é uma ferramenta que permite a serialização de planilhas em RDF através de um mapeamento baseado em template. RDF123 possui uma sintaxe relativamente simples, se comparada às demais ferramentas de mapeamento, que permitem a utilização de cláusulas condicionas, vide Listagem 4. No entanto, só oferece suporte ao mapeamento das colunas da planilha que poderão ser associadas a diferentes esquemas RDF.

RDF123 também possui duas interfaces. A primeira é uma interface gráfica que permite a criação de mapeamentos e a segunda, um Webservice que permite publicar a informação mapeada.

[14] http://www.simile-widgets.org/exhibit/ - Exhibit é um framework que permite a criação de páginas Web com funcionalidades avançadas de busca e filtro de texto, mapas interativos, linhas de tempo e outros tipos de visualizações.

```
<rdf:RDF
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:foo="http://www.foo.org/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <foo:Club rdf:about="#UMBC CS Friday Afternoon Research
Club"/>
      <rdf:Description rdf:about="Ex:$1">
      <rdf:type
      rdf:resource="Ex:foo+@If($4='Yes';'Professor';'Student')"
      />
      <foaf:mbox>Ex:'mailto:'+$2</foaf:mbox>
      <foaf:name>Ex:$1</foaf:name>
      <foo:officeNumber>Ex:$3</foo:officeNumber>
      foo:hasCoffeeDue>Ex:$5^^decimal</foo:hasCoffeeDue>
      <foo:advisor>
      <foo:Professor rdf:about="Ex:$6"/>
...
```

Listagem 4. Exemplo de mapeamento RDF123 aplicando a sintaxe "$n", onde '$' é o marcador e 'n' o número da coluna.

4.3 Frameworks de Armazenamento, Publicação e Manipulação de RDF

Nesta seção estão agrupadas as ferramentas que tem complexidade e escopo que vão além da simples conversão dos dados. As principais ferramentas do mercado estão nesse segmento, e são importantes por possibilitar não apenas a publicação, mas também a manipulação do grafo RDF o que facilita o surgimento de novas aplicações para a Web Semântica.

4.3.1 Jena

Jena [69] é um framework open source para construção de aplicações baseadas em Web Semântica. Ele provê uma API para manipulação do grafo RDF – onde é possível ler e escrever RDF na forma XML, N3 ou N-Triples – e OWL, em memória ou persistida. Além disso, Jena também oferece uma *engine* SPARQL.

Há dois subsistemas para persistência de RDF e OWL em JENA: SDB e TDB. O SDB[15] consiste em uma camada de persistência que utiliza bancos SQL[16]

[15] Veja mais em http://openjena.org/SDB/

e suporta transações do tipo *full ACID (Atomicidade, Consistência, Isolamento, Durabilidade*). TDB[17] é um banco de dados nativo, não transacional, e por este motivo é mais rápido. Também é possível realizar a conversão de informações armazenadas na forma relacional para RDF utilizando a ferramenta SquirrelRDF[18].

SquirrelRDF permite mapear esquemas de bancos de dados relacionais que implementam a API JDBC[19] utilizando a linguagem RDB Mapper baseada em Turtle. Assim, as informações armazenadas nestes bancos podem ser consultadas através de SPARQL. SquirrelRDF acompanha uma QueryEngine ARQ - para acesso Java, uma ferramenta de linha de comando e um servlet para acesso HTTP via SPARQL.

4.3.2 Plataforma D2RQ

D2RQ [65] é uma plataforma que permite tratar banco de dados relacionais não nativos – não-RDF – como um grafo RDF virtual. Através de uma linguagem declarativa, o D2R Mapping Language[20] [66], é possível definir mapeamentos que possibilitam a criação de RDF-views, que permitem que estes bancos possam ser acessados como RDF e Linked Data através da Web. A plataforma D2RQ acompanha um plugin, D2RQ Engine, que reescreve as APIs Jena e Sesame de consulta SQL, possibilitando que banco de dados relacionais possam ser acessados e convertidos para RDF. Além disso, a plataforma ainda oferece um servidor HTTP – D2R Server[21] – que provê um SPARQL enpoint para realizar consultas.

---

[16] PostgreSQL, MySQL, SQL Server, Oracle, IBM DB2, dentre outros.

[17] Veja mais em http://openjena.org/wiki/TDB

[18] Veja mais em http://jena.sourceforge.net/SquirrelRDF/

[19] Veja mais em http://developers.sun.com/product/jdbc/drivers

[20] Veja mais em http://www4.wiwiss.fu-berlin.de/bizer/d2rmap/D2Rmap.htm

[21] Veja mais em http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/

4.3.3 Virtuoso

A plataforma Virtuoso [67] se trata de um servidor multimodal, desenvolvida para o gerenciamento, acesso e integração de conteúdo. Através de uma arquitetura híbrida, Virtuoso possibilita uma distinta lista de funcionalidades como: gerenciamento de conteúdo relacional, RDF, XML e texto.

Com o Virtuoso é possível definir visões – Virtuoso RDF View[22] – que permitem publicar informações de bancos de dados não RDF, tornando-as acessíveis na forma de RDF para que sejam consultadas através de SPARQL ou atualizadas, o SPARUL[23].

Na plataforma Virtuoso a conexão às informações de fontes de dados não-RDF é feita pelo *middleware* Sponger[24], que extrai e converte as informação para RDF através do método apelidado de RDFizer. A extração das informações é possível através do desenvolvimento de conversores apelidados de cartuchos (*cartridge)* que são acoplados ao *middleware*. Um cartucho é constituído por um extrator de meta-informação e um mapeador de ontologias. O *middleware* Sponger faz do Virtuoso uma plataforma extensível, já que é possível o desenvolvimento e adição de novos cartuchos ao *middleware*.

4.3.4 Sesame

Sesame [68] é uma ferramenta de armazenamento e consulta de informações e esquemas RDF. Sua arquitetura se assemelha muito ao Virtuoso, pois permite tratar a informação independente da sua origem. Atualmente é possível acessar o banco de dados relacionais RDF, banco de dados RDF ou repositórios remotos na Web. Também é possível publicar, consultar ou até mesmo criar o seu próprio repositório de RDF na Web. Sesame representou um importante avanço quando foi concebido, em 2001, por ter sido a primeira ferramenta pública disponível a

---

[22] Veja mais em http://virtuoso.openlinksw.com/whitepapers/relational%20rdf %20views%20mapping.html

[23] SPARQL/Update - trata-se de uma extensão da linguagem de consulta SPARQL que provê suporte à adição, atualização e deleção de dados na forma RDF.

[24] Veja mais em http://virtuoso.openlinksw.com/whitepapers/Virtuoso%20Sponger.pdf

realizar consultas em informações e esquemas RDF utilizando a linguagem RQL. Há uma variedade de *plugins* e extensões que foram desenvolvidas para o Sesame ao longo dos anos. Dentre elas podemos destacar:

- Virtuoso Sesame Provider: através do Virtuoso Sesame Provider é possível acessar os dados armazenados no banco de dados do Virtuoso utilizando a API do Sesame;
- Adaptador Sesame para o Banco Oracle que permite a integrar a API do Sesame com o suporte das Tecnologias de Semântica da Oracle.

4.3.5 StrixDB

StrixDB é mais uma abordagem que segue a linha do Sesame, oferecendo uma plataforma que permite o armazenamento de informações na forma RDF através de um banco de dados RDF nativo, StrixStore, e a manipulação do grafo RDF. Além do banco de dados StrixStore, StrixDB também acompanha uma ferramenta gráfica para exploração de grafos RDF, o Visual Graph Requestor. Também é possível tornar as informações acessíveis na Web através do servidor Apache que o acompanha, permitindo realizar consultas e atualizações ao banco de dados nativo, utilizando os protocolos SPARQL e SPARUL ou até mesmo gerar páginas dinâmicas utilizando Lua.

4.3.6 STDTrip

O reaproveitamento de vocabulários é um aspecto importante para a concretização da Linked Data, um padrão da Web Semântica para a publicação de Dados Abertos. Bizer, Cyganiak e Heath [63] enfatizam que para tornar o processamento de informação fácil para aplicações cliente, quando possível, deve-se utilizar termos e vocabulários conhecidos e apenas definir novos termos quando não encontrá-los nos vocabulários existentes. Ainda, segundo os autores, o mesmo é descrito por Casanova [62] quanto à especificação do projeto de banco de dados:

"ao especificar banco de dados que precisam interagir com outros, o projetista deverá selecionar primeiramente um padrão apropriado, se ele existir,

para guiar o design do banco de dados resultante. Se não existir, o projetista deverá publicar uma proposta de um esquema comum, abrangendo o domínio da aplicação" [62]

Em [39] é introduzida STDTrip, uma ferramenta com uma proposta de facilitar o processo de decisão da representação de

"esquemas de bancos de dados relacionais em classes e propriedades do vocabulário RDF, para serem usados como base para geração de triplas RDF" [62]

Breitman [64] entende que

"a definição do vocabulário é um passo importante uma vez que quanto mais se utiliza padrões existentes, mais fácil será interligar os resultados com conjunto de dados existentes." [64]

4.4 Frameworks de desenvolvimento Web baseados em templates

De fato há vários frameworks de publicação de informações na Web baseados em templates. Em particular os *frameworks* Tapestry[25], JavaServer Pages[26] e mais recentemente JavaServer Faces[27] que possibilitam o desenvolvimento de interfaces Web, embutindo informações extraídas de classes Java por meio da definição de marcações específicas. Além da camada de visualização, também fazem parte do escopo a camada de modelo e controle que formam a conhecida arquitetura MVC – *Model View Controll*.

[25] Veja mais em http://tapestry.apache.org/

[26] Veja mais em http://www.oracle.com/technetwork/java/javaee/jsp/

[27] Veja mais em http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

4.4.1
JavaServer Pages

JavaServer Pages, JSP, é uma tecnologia baseada na linguagem Java que permite a geração de conteúdo dinâmico em páginas Web através da adição de código Java com marcações especiais. O código adicionado à página será compilado e executado, a cada *request*, e o resultado incorporado à mesma em tempo de execução. A seguir, na Listagem 5, exemplificamos um template Web utilizando a sintaxe JSP.

```
<html>
      <body>
        <h1>Welcome to JSP</h1>
        <%
                Person p = new Person();
                System.out.println(p.getName());
            %>
      </body>
</html>
```

Listagem 5. Exemplo de template Web utilizando JSP.

4.4.2
JavaServer Faces

O JavaServer Faces, JSF, foi desenvolvido como padrão pelo comitê JCP (*Java Community Process*) com o *Java Specification Request* (JSR – 314) e disponibilizado pela Sun. Nas aplicações JSF as páginas JSP representam a interface. Cada componente JSF pode conter uma ou mais páginas JSP, representando os componentes como formulários, caixas de texto, e botões. Além de possibilitar a extensão e reutilização dos componentes, com JSF também é possível tratar eventos disparados por eles. Atualmente há várias versões de JSF como ICEFaces[28], Jboss RichFaces[29], PrimeFaces[30] e Rich Client Faces[31]. A seguir, na lsitagem 6, um exemplo de template em JSF.

[28] Veja mais em http://www.icefaces.org

[29] Veja mais em http://labs.jboss.com/portal/jbossrichfaces

[30] Veja mais em http://www.primefaces.org

[31] Veja mais em http://www.rcfaces.org

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<f:view>
      <h:outputText value="Welcome to JSF" />
      <h:outputText value="#{person.name}" />
</f:view>
```

Listagem 6. Exemplo de tempalte JSF.

4.4.3 Tapestry

O framework Tapestry é muito similar ao JSF. Tapestry utiliza um modelo de componentes em que cada página HTML corresponde a uma classe. O que diferencia Tapestry dos outros frameworks – JSP e JSF – é a utilização de arquivos de configuração XML, exemplificado na Listagem 8, para descrever a informação que será extraída das classes e inserida nos templates das páginas HTML, exemplificado na Listagem 7.

```
</html>
      <body>
            <h1>Welcome to Tapestry</h1>
            <span jwcid="@Insert" value="Edgard">
                     Message here
            </span>
       </body>
</html>
```

Listagem 7. Exemplo de template Tapestry.

```
<beans>
     <bean id="springappController" class="Person">
          <property name="productManager">
                <ref bean="prodMan"/>
          </property>
     </bean>
</beans>
```

Listagem 8. Exemplo de arquivo de configuração Tapestry.

4.4.4 Less

Less [78] é uma ferramenta de distribuição e utilização de conteúdo no formato Linked Data baseada em templates. Com o Less é possível gerar conteúdo texto não apenas em HTML, mas também diagramas, RSS, ou mesmo mashups de dados, sem qualquer envolvimento de programação. Para tanto, Less combina informações no formato Linked Data ou extraídas de consultas SPARQL em um template que segue o mapeamento LeTL (Less Template Language), uma extensão da linguagem de template Smarty[32], exemplificado na Figura 4. Less permite com que modelos de tipos comuns de entidades possam ser combinados, permitindo uma ampla gama de aplicações, tais como na produção de conteúdo Web.

```
<h2>{{foaf:name}}</h2>
{{dc:description}} <br />
<a href="{{foaf:homepage}}">web</a>
<div id="members">
 {foreach {{foaf:member}} as $var}
  {template id="5" uri="{$var}"
    parameter_language="en"}
 {/foreach}
</div>
```

```
<div id="foaf-card-block">
 {if {{foaf:depiction}} !== ''}
  <img class='photo'
    src="{{foaf:depiction}}" />
 {/if}
 <div id="name">{{foaf:name}}</div>
 ...
 {if {{foaf:phone}} !== ''}
  <div id="tel">{{foaf:phone}}</div>
 {/if}
 <div id="email">
  <a href="{{foaf:mbox}}">
   <img src="email.png" />
   {if $language === 'en'}
     email {else} E-Mail
   {/if}
  </a>
 </div>
</div>
```

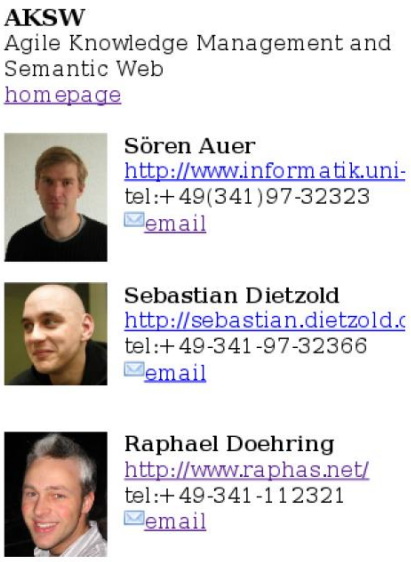

Figura 4. Exemplo de template utilizando LeTL para geração de conteúdo. [78]

O framework proposto apresenta uma abordagem semelhante ao Less, que tem algumas limititações. A geração de conteúdo do Less, ao contrário do Babel, baseia-se apenas em informações no formato Linked Data e na serialização do tipo XML. Além disso, apresenta uma linguagem complexa de mapeamento

[32] http://www.smarty.net/

(LeTL) que distancía a estrutura do template com a do conteúdo gerado, embora seja possível compartilhar os templates cirados através de um repositório e realizar operações mais complexas como a geração de sub-templates.

4.5 Outras Tecnologias

4.5.1 Suporte a estruturas XML do SQL Server

A Microsoft desenvolveu o suporte nativo a XML no SQL Server desde a versão 2000. Esse suporte é sustentado através de duas tecnologias: FOR XML, que garante a geração dinâmica de conteúdo XML a partir do modelo relacional, e XML Data Type que possibilita o armazenamento de instâncias XML nativamente.

O FOR XML, exemplificado na Listagem 9 e 10, suporta quatro tipos de transformação XML – RAW, AUTO, EXPLICIT, e PATH – que podem ser vistas com mais detalhes em Suporte a Integração Relacional/XML *Server-Side* [70]. Segundo a Microsoft [71] a natureza hierárquica do XML torna difícil sua reprodução em um modelo relacional quando a estrutura torna-se complexa como, por exemplo, o aumento da profundidade. Além disso, há um custo significativo envolvido na construção de um documento XML a partir de dados relacionais. No entanto, essas limitações podem ser superadas através da utilização de instâncias XML nativas – XML Data Types.

```
WITH XMLNAMESPACES ('uri' as ns1)
SELECT ProductID as 'ns1:ProductID',
       Name      as 'ns1:Name',
       Color     as 'ns1:Color'
FROM Production.Product
WHERE ProductID=316 or ProductID=317
FOR XML RAW ('ns1:Prod'), ELEMENTS
```

Listagem 9. Exemplo simples de uma consulta utilizando FOR XML com RAW.

```
<ns1:Prod xmlns:ns1="uri">
  <ns1:ProductID>316</ns1:ProductID>
  <ns1:Name>Blade</ns1:Name>
</ns1:Prod>
<ns1:Prod xmlns:ns1="uri">
  <ns1:ProductID>317</ns1:ProductID>
  <ns1:Name>LL Crankarm</ns1:Name>
  <ns1:Color>Black</ns1:Color>
</ns1:Prod>
```

Listagem 10. Resultado obtido pela consulta na Listagem 9.

4.5.2 Tecnologias de Web Semântica da Oracle

Algumas companhias de banco de dados já começaram a se engajar na construção de sistemas de apoio à Web Semântica. Como exemplo pode-se falar das Tecnologias de Web Semântica da Oracle [73] que possibilitam a utilização do banco de dados relacionais Oracle para armazenar e consultar triplas RDF, realizar a conversão de dados relacionais para RDF através de consultas, e utilizar inferências fornecidas ou definidas pelo usuário para expandir a capacidade da consulta sobre a informação armazenada.

O banco que utiliza a Tecnologia de Semântica da Oracle pode conter não apenas triplas RDF e ontologias (OWL), como também dados relacionais tradicionais. A capacidade da consulta pode ser expandida através da utilização de uma base de regras que possibilitam realizar inferências. As inferências permitem realizar deduções lógicas baseadas na informação e nas regras.

As triplas armazenadas são mantidas sob modelos específicos de dados semânticos criado pelo usuário. Um modelo criado pelo usuário tem um nome que refere-se às triplas armazenadas em uma coluna da tabela especificada. Todas as triplas interpretadas são armazenadas no sistema como entradas em tabelas no esquema MDSYS. A tripla (sujeito, predicado, objeto) é tratado como um objeto de banco de dados. Como resultado, um único documento contendo várias triplas resultará em vários objetos no banco.

5
TML (Template Model Language)

Antes de começarmos a apresentar a linguagem TML é preciso entender o contexto no qual o trabalho foi desenvolvido, a evolução, abordagens e problemas que surgiram ao longo do tempo. Quando iniciamos esse trabalho, o grupo RDB2RDF do W3C ainda não havia sido criado. Nossa preocupação era desenvolver uma linguagem que possibilitasse a usuários familiarizados com as estruturas de dados RDF criarem mapeamentos, utilizando-se modelos produzidos e publicados por outros usuários. De modo a oferecer apoio aos usuários durante o processo de geração dos mapeamentos, concebemos uma linguagem baseada em templates. A idéia surgiu durante a execução de um curso promovido e ministrado junto ao W3C Brasil. Nossos alunos realmente haviam entendido os conceitos por trás da Web Semântica, sabiam como criar, e até mesmo como obter estruturas similares – RDFs – que eles pretendiam utilizar. No entanto, não estavam familiarizados com as ferramentas que exigiam um conhecimento de linguagens de mapeamento. Estas linguagens são, em geral, complexas e podem demandar conhecimento de programação. O nosso desafio era então a definição de uma linguagem de mapeamento simples para ser utilizada como base para a conversão das várias fontes de dados.

Os usuários com conhecimentos avançados de Web Semântica não encontram a mesma dificuldade que os usuários com menor conhecimento. Para os primeiros, é difícil imaginar as dificuldades que um usuário comum pode ter ao realizar um determinado mapeamento. Sem sombra de dúvida, estava claro que para contribuir para o crescimento da Web Semântica e torná-la realidade era preciso que as ferramentas fossem fáceis o suficiente, de forma que usuários com pouco conhecimento pudessem publicar seu conteúdo. Neste processo não devemos nos esquecer da existência de vocabulários que abrangem com certa propriedade determinados domínios. O objetivo da Web Semântica não é apenas tornar a informação acessível, mas também interligá-la.

Os meses se passavam e muitas soluções surgiam, porém ainda sem propriedade suficiente para estimular o crescimento da Web de dados. Neste contexto estavam os *Wrappers*, dentre eles o mais conhecido, Babel[33], desenvolvido pelo MIT. A adesão aos *Wrappers* foi grande, por serem simples e de fácil uso. Nos *Wrappers* toda a conversão é realizada utilizando mapeamento direto (*Direct Mapping*), onde as propriedades, classes e valores são extraídos e criados a partir da fonte original ou de algum vocabulário próprio. É importante notar que o mapeamento das propriedades, valores e classes é fixo e os resultados poderão ser: dados que não estarão propriamente interligados com outras fontes e conseqüentemente não poderão ser reaproveitados; ou a publicação de informações indesejadas, privadas.

Com o objetivo de resolver o problema da reutilização de vocabulários, surgiram os mapeamentos (Triplify Script [1], R2RML[50], Direct Mapping[40], RDF123 MAP[95], D2R MAP [100], LeTL [78]). Os mapeamentos começaram a ser utilizados para mapear as propriedades e valores das fontes de dados com as respectivas propriedades e classes dos vocabulários existentes, contribuindo para a re-utilização de vocabulários e o desenvolvimento da Web Semântica. Muitas linguagens e ferramentas apareceram em um curto espaço de tempo, sobretudo ferramentas para banco de dados relacionais, o que contribuiu para a criação do grupo RDB2RDF incubado na W3C, objetivando o estabelecimento de uma linguagem comum a ser utilizada entre as várias ferramentas de conversão de Banco de Dados Relacionais para RDF.

Após este rápido intervalo, a preocupação se tornou a criação de interfaces e métodos simples para a criação dos mapeamentos. Como exemplo, tem-se o STDTrip que foca no mapeamento semi-automático de estruturas de Banco de dados Relacionais para ontologias, abstraindo do usuário a necessidade de se aprender especificamente algumas linguagens de mapeamento. No entanto, ainda há muito a ser feito para que essas ferramentas se tornem utilizáveis de fato, pois ainda necessitam de uma intervenção manual para se chegar a resultados satisfatórios.

Outro aspecto importante era a abrangência das linguagens de mapeamento. Apesar do enorme número, nenhuma era ampla o suficiente para expressar o

[33] http://simile.mit.edu/babel/

mapeamento; não apenas de Banco de Dados Relacionais, como formatos proprietários e de outras fontes encontradas em sites governamentais, como planilhas; tudo isso aliado à dificuldade que alguns usuários encontravam para realizar a manutenção dos mapeamentos gerados, uma vez que tanto o esquema quanto os vocabulários da linguagem de mapeamento e da fonte dos dados pode sofrer alterações ao longo do tempo. R2RML, a linguagem padrão para representar RDB, por exemplo, é constituída de 15 classes, várias propriedades, e passou por três versões em menos de um ano, o que leva muitos usuários a optarem pelo mapeamento direto.

Outra deficiência das linguagens é a inflexibilidade. Com R2RML, por exemplo, é possível realizar o mapeamento de fontes de dados relacionais para RDF/XML e N3; mas o que fazer com os demais formatos existentes na Web Semântica como RDFa e JSON?

A existência desse enorme vazio separando os criadores de conteúdo da Web com das ferramentas da Web Semântica nos motivou a propor uma nova linguagem, mais poderosa e abrangente que as existentes, baseada em Templates, a Template Model Language (TML).

Ao contrário das demais linguagens, a TML não define a estrutura da informação, como listas ou loops, nem tampouco a fonte, ou a forma de representação.

Novas formas de representação da informação podem surgir. Desta forma, separar a camada de modelo da linguagem nos possibilita uma maior flexibilidade. A adição de regras à linguagem a torna mais complexa, porém expressiva. Linguagens de templates já são suficientemente expressivas por possibilitarem a utilização de modelos, logo, a inclusão de regras deve ser fortemente ponderada.

A linguagem de definição de mapeamentos proposta é orientada a templates, i.e., fornece uma estrutura que orienta os usuários no processo de definição de mapeamentos de fontes de dados para RDF, e pode ser utilizada em conjunto com qualquer estrutura de informação como, por exemplo, formatos utilizados pela Web Semântica, e.g. XML, JSON, N3 e N4.

TML baseia-se nas estruturas, tipicamente encontradas em documentos, que dividimos em simples e compostas. As estruturas simples são aquelas que geralmente carregam um valor, um literal, já as estruturas compostas são aquelas

que encapsulam mais de uma estrutura simples, estruturas compostas ou ambas. Se analisarmos qualquer documento, certamente veremos que toda estrutura pode ser expressa através desses dois tipos de elementos. Esse mesmo conhecimento é utilizado, por exemplo, na definição das marcações de um documento XML. Dessa forma, um vetor contendo uma cadeia de caracteres, pode ser definido através de um elemento composto e vários elementos simples do tipo cadeias de caracteres (*Strings*), como exemplificado na Figura 5.

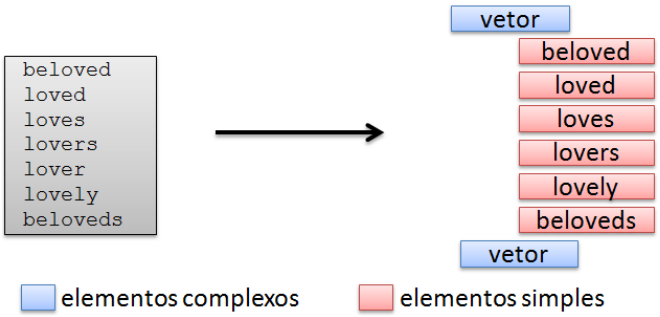

Figura 5. Representação de um vetor em uma estrutura de dados utilizando elementos simples e complexos.

Um relacionamento entre duas tabelas pode ser interpretado como uma composição de dois elementos compostos, onde o primeiro deles representa uma determinada propriedade do segundo, como exemplificado na Figura 6.

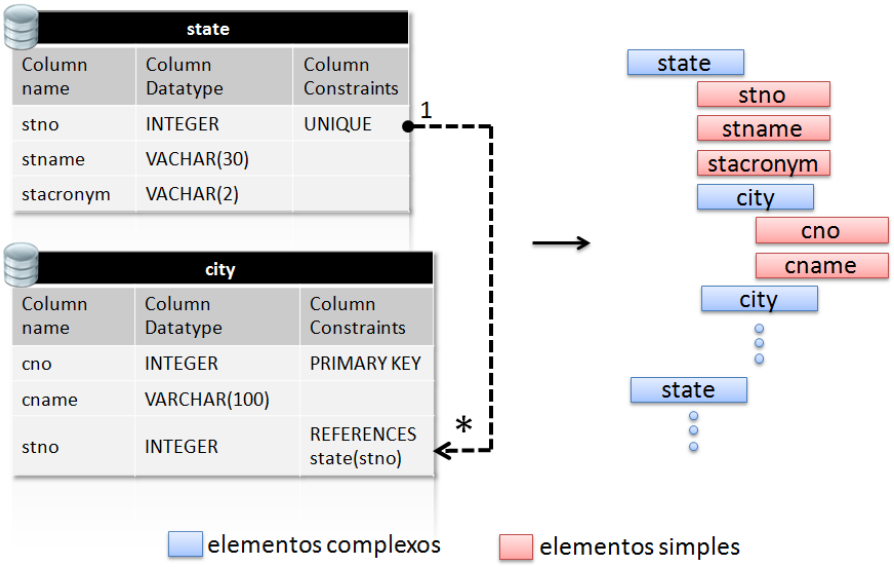

Figura 6. Representação de uma estrutura formada por dois elementos compostos, onde a Tabela cidade (*city*) representa uma propriedade composta da estrutura estado (*state*).

Já que estamos convencidos de que qualquer estrutura pode ser organizada dessa forma, definindo hierarquias, propriedades e tipos de objetos, é razoável utilizar essas hierarquias para definir a estrutura da informação, e não o inverso. Ou seja, o que tentamos com o TML é definir onde uma determinada informação deve ocorrer em certa estrutura e utilizar a estrutura da informação de origem para determinar a quantidade de ocorrências dessa informação.

A utilização do TML permite portar a informação de um contexto de origem para um novo contexto, definido pelo usuário. TML também não especifica o tipo de estrutura de destino, o que permite portar para diversas outras estruturas, embora, no contexto deste trabalho exploramos apenas a conversão para o XML e triplas.

TML trabalha com o mapeamento de coleções de elementos. Assim, como em uma Tabela de banco de dados, os elementos contidos na coleção possuem os mesmos atributos em comum. Uma vez que as coleções estão definidas e que a estrutura de destino é conhecida, o mapeamento deve ser realizado por meio das seguintes regras, exemplificadas na Figura 7:

1. As coleções de objetos deve seguir o padrão, “#{coleção.atributo}” ou “#{coleção}”, onde:

a. ‘#’ é um indicador (token) definido pela aplicação que determina o início de uma expressão interpretável pela máquina de template;
b. ‘{‘ e ‘}’ determinam respectivamente, o início e o fim da expressão;
c. ‘coleção’ representa o nome da coleção;
d. ‘atrubuto’ representa o atributo que se deseja mapear;

2. Caso um elemento tenha sido mapeado com um atributo da coleção, a estrutura resultante conterá um elemento idêntico ao mapeado, para cada indivíduo da coleção, na mesma ordem e com os mesmos valores.
3. Caso uma estrutura tenha sido mapeada com a mesma coleção em níveis diferentes, a replicação da estrutura se dará no elemento de nível mais alto.
4. A utilização apenas do nome da coleção determina que se deva atribuir o valor vazio à estrutura de destino.

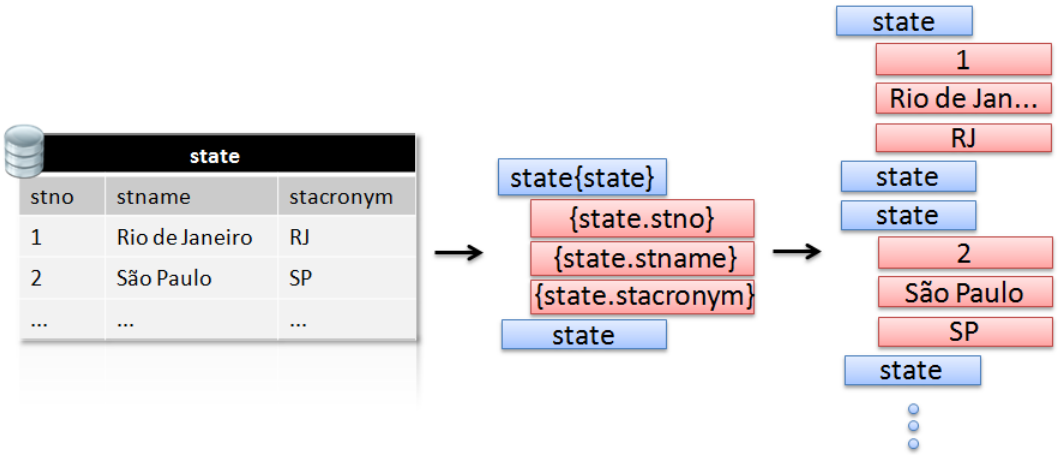

Figura 7. Representação dos dados a partir de um template genérico utilizando TML.

Na Figura 8, abaixo, descrevemos uma base de dados, definimos uma coleção e serializamos a informação extraída da coleção em três estruturas diferentes – JSON, N3 e XML – utilizando chaves como marcação.

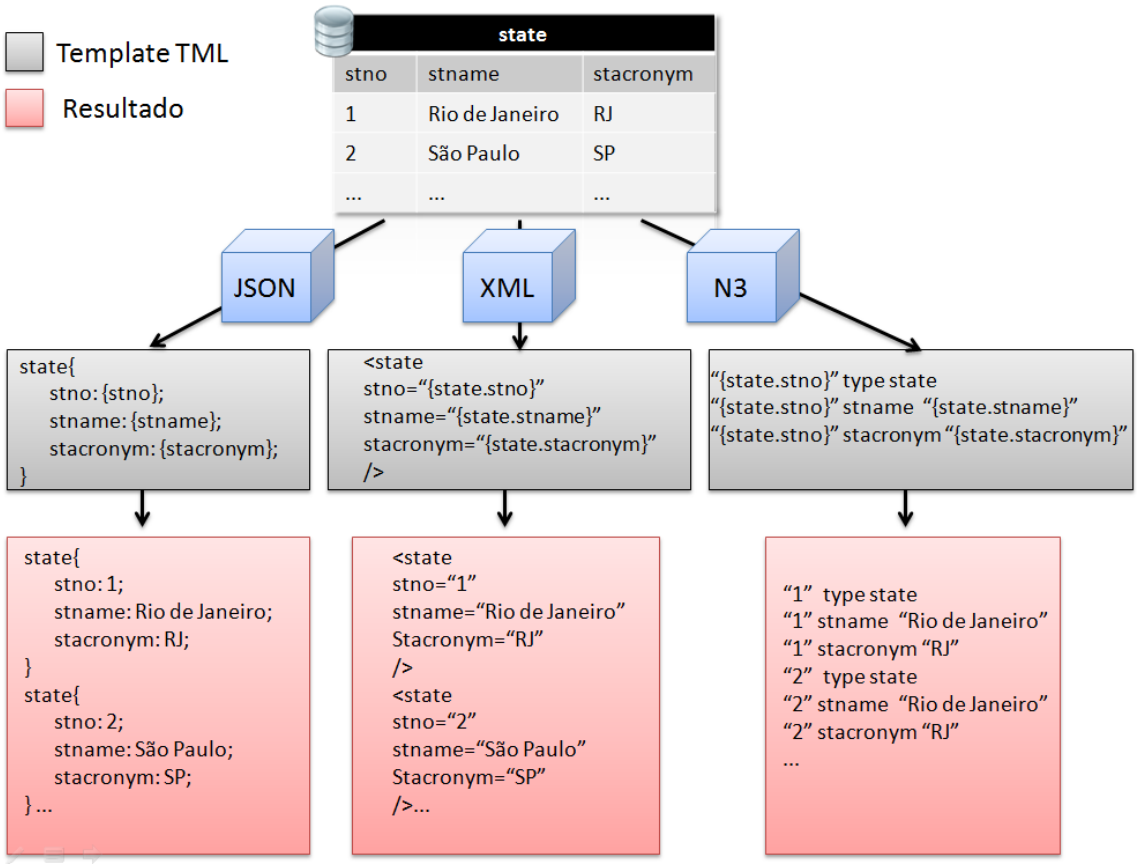

Figura 8. Resultado da conversão dos dados contidos em uma coleção *state* para diferentes formatos de serialização (JSON, XML e N3) utilizando TML.

5.1 Utilização de templates em estruturas sensíveis ao contexto

Nosso estudo estaria completo não fosse a existência de estruturas mais complexas que as demonstradas anteriormente, onde os templates apenas são uma reflexão das estruturas de origem. A existência dessas estruturas determinou a utilização do nome da coleção para indicar em qual nível a estrutura será replicada a cada novo elemento inserido.

Vejamos o caso, por exemplo, de um documento HTML, como ilustrado na Figura 9; O documento HTML é formado por estruturas de dados que são sensíveis ao contexto. A alteração de qualquer atributo ou ordem de qualquer elemento influenciará na maneira com que documento será interpretado pelos *browsers*. O mesmo não pode ser afirmado para as estruturas apresentadas anteriormente, onde a ordem de qualquer elemento ou atributo na estrutura, não causaria prejuízo na representação da informação.

Embora a marcação do nível da estrutura que será replicado possa ser feita utilizando um atributo da coleção, deve-se notar que sua utilização implica necessariamente na adição de um determinado valor à estrutura de destino, o que não ocorre utilizando apenas o nome da coleção, como exemplificado na Figura 9.

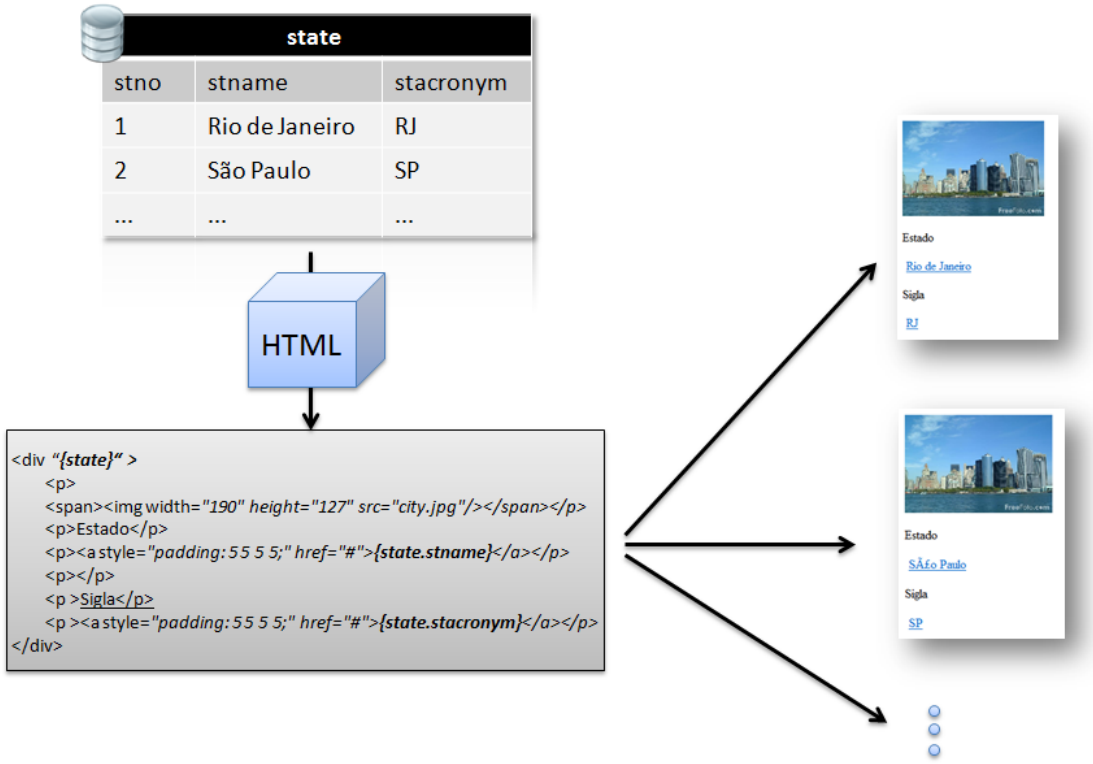

Figura 9. Utilizando templates em estruturas sensíveis ao contexto.

5.2 Escrevendo um compactador de texto simples utilizando TML

Tanto para promover, como para demonstrar a flexibilidade da linguagem proposta, optamos por explorar a construção de um compactador de texto baseado na TML.

Imagine um fragmento de texto extraído de qualquer documento, livro, jornal ou site. Existe um padrão entre todos eles, a utilização de um vocabulário, composto por N palavras, verbos, adjetivos, pronomes, substantivos, etc., além de uma série de N caracteres especiais de acentuação que se encontram ao longo de várias linhas. É possível perceber também que estes M caracteres e N palavras se repetem de uma a K vezes ao longo do texto (onde K é um número maior que

zero). Se isso é verdade, podemos afirmar que um texto é um conjunto de linhas formadas por um determinado conjunto de cadeia de caracteres que se repetem ao longo do documento, também podemos afirmar que é possível navegar pelas linhas e identificar as ocorrências dessas cadeias de caracteres.

A partir do momento que podemos identificar as cadeias de caracteres existentes ao longo do texto, para realizar a compactação, basta substituir a cadeia no texto por um marcador que mapeie a cadeia listada ao lugar onde ela aparece no texto. Supondo que o marcador contenha uma cadeia de caracteres menor que a palavra mapeada, é possível afirmar que: seja K o número de cadeias de caracteres substituídas pelo marcador e L a diferença entre o tamanho da palavra pelo tamanho do marcador, teremos uma compactação na ordem de (K)*L. Aplicando essa lógica recursivamente às M cadeias restantes teremos uma ordem total de M*(K)*L, , como exemplificado na Figura 10.

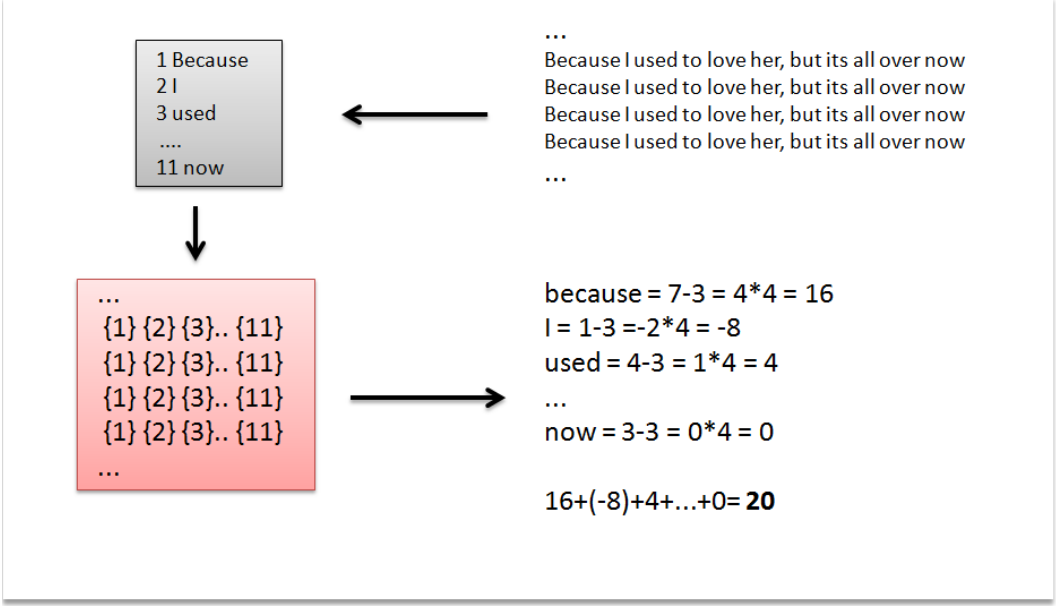

Figura 10. Exemplo de compactação de texto utilizando TML. No exemplo acima utilizamos a linguagem TML para realizar uma compactação de um trecho da música *It´s all over now* do grupo Rolling Stones. Ao final do processo é possível perceber que houve uma compactação de vinte caracteres do texto original.

5.3
Mapeamento direto utilizando TML

O termo mapeamento direto é comumente utilizado para descrever o processo que consiste na criação de instâncias RDF a partir de um esquema retirado de um banco de dados, no qual as Tabelas são mapeadas para classes e as colunas para propriedades. Embora esse processo seja vastamente debatido no plano de trabalho da W3C chamado "*A Direct Mapping of Relational Data to RDF*" [40], o termo é vastamente empregado em uma variedade de ferramentas, a exemplo dos *Wrappers*.

No mapeamento por templates, o esquema das coleções é utilizado para mapear as informações na estrutura de dados. No mapeamento direto, o esquema do modelo do banco de dados é utilizado na definição das classes e propriedades. Se é possível gerar o esquema automatizado a partir do esquema do modelo do banco de dados, também é possível gerá-lo a partir de coleções. Comparativamente podemos enxergar as coleções como tabelas, uma vez que estas possuem o mesmo conjunto de atributos, e realizar o mapeamento convertendo o esquema da coleção em classes e propriedades. Coleções podem não ter chave primária, no entanto a técnica de mapeamento direto especifica que não havendo chaves primárias, a posição da tupla deva ser tomada como identificador. Ao final, cada coleção será representada por uma classe como demonstrado na Figura 11 abaixo.

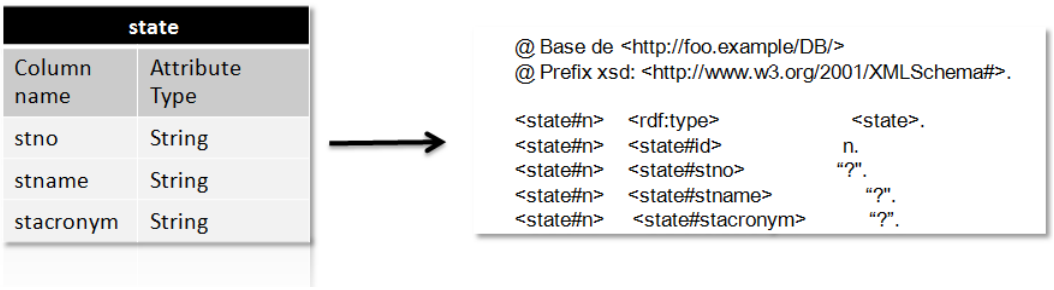

Figura 11. Demonstração do mapeamento da coleção *state* para triplas utilizando a estratégia de mapeamento direto onde *n* representa o número da ocorrência da tupla na coleção e o caracter '*?*' a informação contida na tripla do respectivo atributo mapeado.

Uma vez que demonstramos que a o mapeamento direto de coleções é possível, outro passo seria a geração automatizada do template, mas esse processo se torna trivial uma vez que o template também utiliza o esquema das coleções no mapeamento das informações. Dessa forma, para cada coleção existirá uma classe de mesmo nome e atributos conforme a Figura 12.

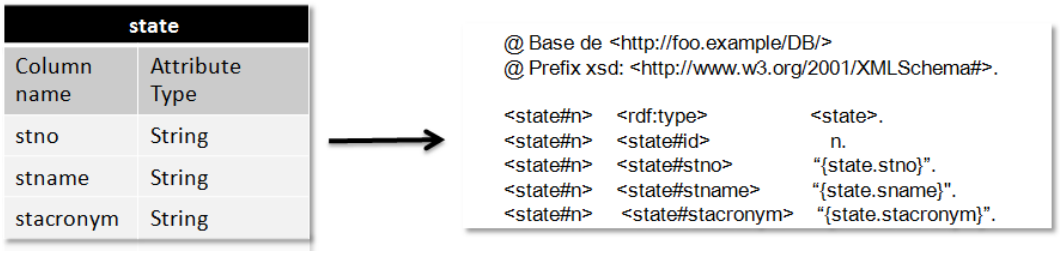

Figura 12. Template de triplas gerado a partir da coleção state, utilizando a técnica de mapeamento direto.

6
Babel

Babel representa uma abordagem inovadora para a conversão de dados tradicionais para os formatos da Web Semântica, e apresenta uma série de melhorias em relação às técnicas existentes. Em primeiro lugar, a maioria das ferramentas RDB para RDF são baseadas em alguma estratégia de mapeamento. Mapeamento consiste no processo de redefinição do conteúdo, conceitos e relações em termos de descrições semanticamente mais ricas, por exemplo, vocabulários RDF [84][85]. A fim de fazê-lo, os usuários são obrigados a aprender alguma técnica ou estratégia de mapeamento [86], um processo cognitivamente exigente, que requer muitas vezes familiaridade com alguma linguagem de mapeamento empregada.

Atualmente existem inúmeras estratégias de mapeamento de dados relacionais para RDF [87]. Sahoo [88] fornece uma análise compreensível das abordagens existentes, na qual são analisadas mais de 15 estratégias, classificadas em 3 grandes classes: Projetos e Ferramentas, Projetos Prova de Conceito e Aplicações de domínio específico. Mais importante do que o número de estratégias existentes, é o fato de que elas estão em constante evolução. Durante o período que este estudo estava sendo realizado, a W3C produziu três versões para seus padrões, tanto para o mapeamento direto, que define uma transformação direta em que o vocabulário RDF reflete diretamente os nomes de elementos do domínio [40], quanto para o R2RML, uma linguagem que permite realizar mapeamentos personalizados de bancos de dados relacionais em conjuntos de dados RDF [89] [90] [91].

Babel facilita o processo de aprendizagem, uma vez que tem um enfoque baseado em templates que guiam os usuários durante o processo de construção dos mapeamentos [92]. Templates permitem capturar e encapsular o conhecimento do domínio e orientar os usuários na realização das tarefas necessárias [93].

Além disso, Babel é mais flexível, pois abrange outros formatos da Web Semântica que não RDF/XML, incluindo OWL/XML, RDFa e N3. A possibilidade de serialização de informações em mais de um formato da Web Semântica é muito importante no cenário atual, por possibilitar a utilização da mesma informação por diversas aplicações e a criação de Mashups [97][39].

Babel foi projetado para facilitar e promover a conversão de dados de formatos tradicionais, em particular aqueles armazenados em bancos de dados relacionais e planilhas, para formatos compatíveis com a Web Semântica, mas permite extensões cutomizadas, o que possibilita aos usuários escrever extensões para habilitar o uso de outras fontes de dados, através de Plugins. Um modelo de desenvolvimento preferido pela comunidade de engenharia de software, por encapsular o conhecimento sobre como resolver problemas, operar em diversos ambientes de desenvolvimento, e promover a integração com os sistemas existentes, programas e aplicações [98]. Arquiteturas de Plugins são extensíveis, permitindo aos usuários adicionar novas funcionalidades, personalizar e adaptar a ferramenta às suas necessidades específicas. Elas são tipicamente menos complexas do que as ferramentas monolíticas, modular, e mais portável [94].

A principal contribuição desta abordagem em relação a ferramentas de conversão de dados existentes [11], no entanto, é o apoio que fornece aos usuários. Babel suaviza a curva de aprendizagem, eliminando a necessidade de aprender uma nova estratégia de mapeamento, que é substituída pelo uso de templates. Os templates são utilizados para guiar o usuário na definição dos mapeamentos, na escolha dos vocabulários adequados, e na definição do formato de saída dos dados.

6.1 Arquitetura e conceitos

Nesta seção apresentaremos a arquitetura do Babel. Listamos, a seguir, os componentes, que serão detalhados nas seções subseqüentes. A Figura 13 ilustra os principais componentes, sua organização, e como estão relacionados.

- DataStore: abstração de fonte de dados;
- Collection: abstração de conjunto de dados;
- TML: define o conjunto de regras para a criação dos Templates;

- Template: permite criar, salvar e compartilhar mapeamentos;
- Máquina de processamento de Templates (Template Engine): responsável pelo processamento dos dados proveniêntes das Collections e inseri-las nos templates.
- Arquivo de Configuração (Config File): permite criar, salvar e compartilhar conFigurações de execução do Framework;
- Servlet: permite publicar dados na Web através de uma interface HTTP.
- Interface Gráfica de Usuário (GUI): permite definir coleções, criar mapeamentos automatizados, salvar conFigurações, e converter informações de uma forma gráfica.
- Inteface de linha de comando: permite a execução da Template Engine através de comandos digitados.

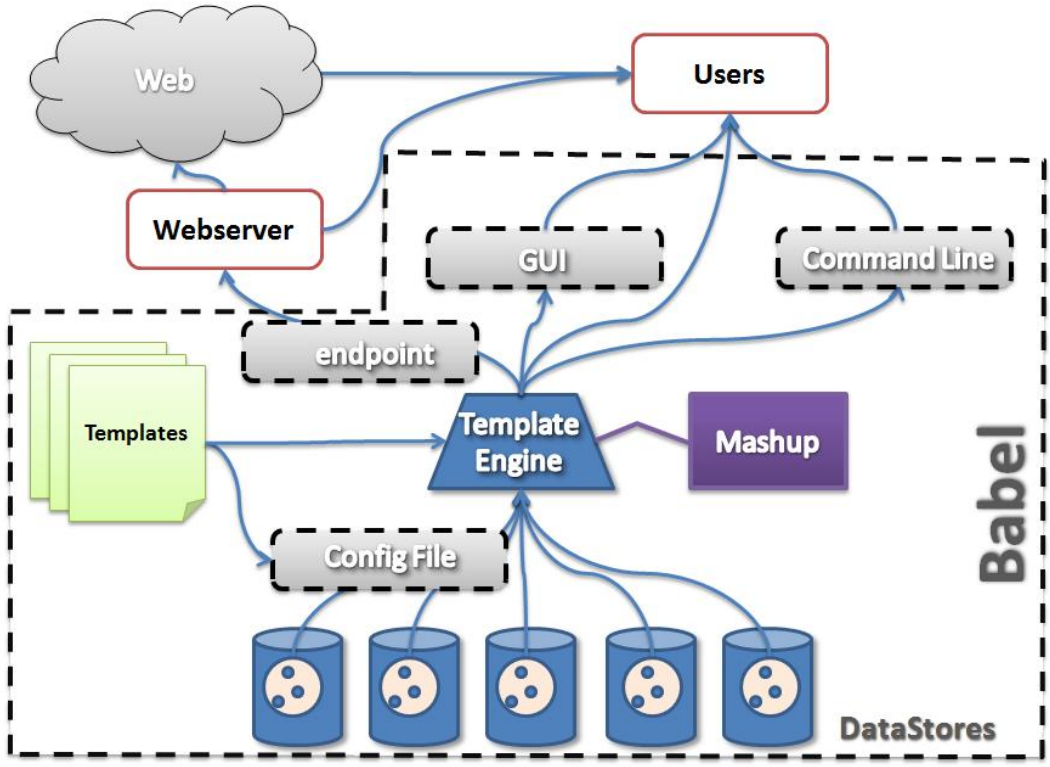

Figura 13. Diagrama que ilustra os componentes principais da arquitetura do Babel e suas relações.

6.1.1
DataStores e Collections

Babel foi concebido sob dois conceitos principais o DataStore e a Collection. Um DataStore representa uma fonte de informação. Como o processo de conversão pode utilizar várias fontes de informação (banco de dados, arquivos proprietários e planilhas) a arquitetura proposta permite com que DataStores possam ser facilmente adicionados. Cada DataStore poderá possuir uma ou mais visões para a informação que armazena, esta abstração é chamada de Collection.

Uma Collection, representa um conjunto de informações, com as mesmas propriedades, que pode ser obtida através da aplicação de um filtro sob um DataStore. A forma de se definir o filtros para diferentes Collections é ditada pela implementação do DataStore. A Collection, ilustrada na Figura 14, é importante para restringir o acesso a informações sensíveis e.g. senhas e dados pessoais.

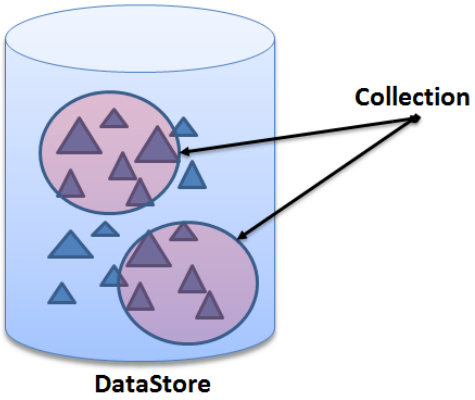

Figura 14. Ilustração de um DataStore contendo duas coleções de dados.

Babel já oferece suporte à leitura das principais fontes de dados, como planilhas, banco de dados relacionais e estruturas em memória que serão apresentados na seção 8.

6.1.2
O template

O template baseia-se na linguagem TML, apresentada no capítulo 5, e, portanto, é o mecanismo pelo qual a informação é mapeada. A utilização do template é importante porque fornece tanto a estrutura quanto o vocabulário que será utilizado. TML é flexível, pois não define uma estrutura fixa para a

informação, possibilitando a portabilidade da informação original para qualquer tipo de estrutura. Entretanto, Babel suporta apenas a serialização de estruturas de dados no formato XML, triplas e quádruplas (N4) [99].

A criação e o mapeamento do template utilizando TML são mais fáceis porque pessoas que são familiarizadas com estruturas da Web Semântica, por exemplo, RDF, se tornam aptas a editar e fazer o mapeamento manualmente, utilizar qualquer ferramenta da Web Semântica, que possibilite criar a estrutura da informação em um dos formatos possíveis ou utilizar estruturas e ontologias existentes. Como um dos objetivos da Web Semântica é a publicação e interligação de informações, é cada vez mais fácil encontrar estruturas que poderão ser utilizadas como templates. Elas podem ser encontradas utilizando ferramentas de busca de ontologias [17], ou acessadas através de *endpoints* SPARQL.

Após a definição da estrutura, o próximo passo passa a ser o mapeamento. Para fazê-lo o usuário deverá obedecer ao conjunto de regras estabelecidas pela TML, isto é, a informação deverá ser endereçada na estrutura do template através de marcações, onde o nome da coleção da estrutura de origem deverá ser precedido de tralha (“#”), seguido de um ponto (“.”) e o nome do atributo da coleção, eg: “#” + coleção + “.” + atributo, como demonstrado na seção 6.

No Babel, o endereçamento deverá ser feito utilizando o *id* da Collection. Este mecanismo permite maior flexibilidade uma vez que os detalhes referentes tanto a conexão quanto ao meio de obtenção da informação ficam reservados à aplicação.

6.1.3
O Arquivo de Configuração

O arquivo de configuração é uma interface que permite definir, salvar ou alterar parâmetros de configurações utilizados pela ferramenta, sem que seja necessário fazer alterações no código. O template, o conjunto de DataStores e as Collections são exemplos de configurações que podem ser inseridas no Arquivo de Configuração.

O formato do arquivo de configuração é XML. A criação do arquivo é regulamentada por um conjunto de regras definidas pelo esquema (*XML Schema* Definition) do Babel na Figura 15, que serão detalhadas a seguir.

O arquivo deve ser iniciado e ser terminado com a marcação *babel-config*. A marcação babel-*config* pode possuir de uma a várias tags do tipo DataStore, e uma tag do tipo template, onde serão colocados os templates que serão utilizados no processo de conversão. Atualmente a ferramenta suporta três tipos de formatos de templates: N3, N4 e XML.

A tag DataStore possui um Descriptor e um conjunto de Collections. Na tag Descriptor serão colocados os parâmetros de conexão correspondentes à implementação do DataStore que será utilizado.

O Descriptor possui tags do tipo Param, que servem para adicionar parâmetros específicos de cada implementação. A tag Param possui dois atributos: *name* e *value*. O atributo *name* corresponde ao nome do parâmetro e *value* ao valor. Essa tag é especialmente importante para garantir a versatilidade do modelo, pois possibilita que parâmetros possam ser adicionados, removidos ou alterados de acordo com a necessidade de cada implementação utilizada. Uma tag Param não contém restrições de tipo ou quantidade de tags que pode encapsular.

A Collection possui um identificador definido pelo atributo *id*. Esse identificador será útil durante o processo de conversão da informação, pois é utilizado no mapeamento das informações no template. A Collection também possui um atributo que permite localizar o identificador único da Collection, e um conjunto te tags do tipo Param. O indentificador único possibilita correlacionar informações de diferentes Collections e as tags do tipo Param e, dentre outras coisas, a restringir o conjunto de informações que farão parte da coleção.

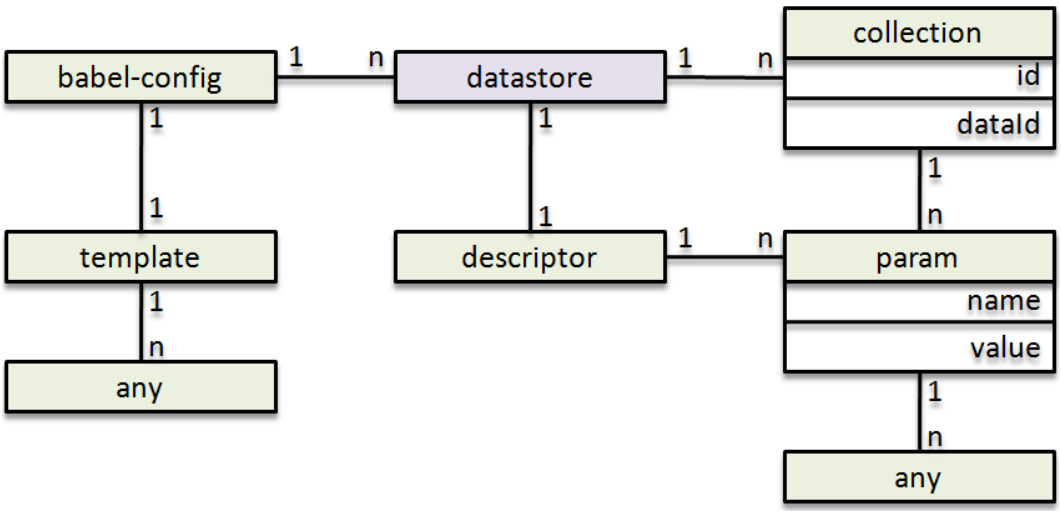

Figura 15. Diagrama de modelo do esquema do arquivo de configuração.

Tanto o elemento *template* quanto o elemento *param* podem encapsular elementos de qualquer tipo, e em qualquer quantidade. Essa característica garante a versatilidade do modelo uma vez que, caso haja necessidade, elementos de diferentes naturezas podem ser adicionados ao modelo.

6.1.3.1 Exemplo

Imaginemos agora que desejamos publicar informações provenientes de uma base de dados utilizando o vocabulário FOAF. Esta base de dados possui uma tabela cujo nome é *person*, contendo informações sobre pessoas como: nome, endereço, idade, email, CPF e RG.

Como podemos perceber esta tabela possui um conjunto de atributos sensíveis que, dependendo do objetivo e da plataforma de veiculação dos dados, não deverão ser publicados e um conjunto de atributos passível de publicação (nome, endereço, email, idade). Chamamos de dados sensíveis, aqueles que possibilitam a identificação do indivíduo ou que contém informações pessoais como senhas. Portanto iremos publicar apenas os dados pertinentes através da criação da Collection cujo nome é o mesmo da tabela, isto é, *person.*

Para publicar o conteúdo da tabela person é preciso, antes de tudo, definir o DataStore. Para definir o datastore é preciso criar um Descriptor que contenha os parâmetros de conexão. No nosso exemplo iremos nos conectar a um banco de dados hipotético cuja conexão é realizada por um DataStore que deverá receber

três parâmetros: URL_CONNECTION, que contém uma URL de conexão baseada no padrão JDBC (“jdbc:mysql://host:port/dbName/”); o User, que contém o nome do usuário que se conectará no banco de dados; e o PASSWORD, onde é inserida a senha do usuário. Ambos vazios no exemplo.

Após configurar os parâmetros do DataStore, é necessário configurar a Collection. Cada Collection contém um conjunto de parâmetros que serão interpretados por seu DataStore. No nosso exemplo o parâmetro que restringe o conjunto de dados da Collection é *query* e ele possui como valor uma cláusula *select* do banco de dados (“Select name from person”).

```
<babel-config>
  <datastore>
   <descriptor>
      <param name="URL_CONNECTION"
        value="jdbc:mysql://host:port/dbName/”>
     <param name="USER" value=""/>
     <param name="PASSWORD" value=""/>
   </descriptor>
   <collection id=”person”>
              <param name=”query” value=”Select name from person”>
   </collection>
  </datastore>
   <tenplate>
   <foaf:Person>
            <foaf:name>#{person.name}</foaf:name>
   </foaf:Person>
  </tenplate>
</babel-config>
```

Listagem 11. Exemplo hipotético de um arquivo de configuração do Babel.

Após definir o conjunto de DataStores e Collections, será preciso criar o template seguindo a orientação da linguagem TML no capítulo 6 que no nosso exemplo, na Listagem 11, é uma estrutura RDF/XML do tipo FOAF.

6.1.4
Máquina de processamento de Templates (Template Engine)

A máquina de processamento de templates é responsável por processar o template, e mapear a informação necessária extraídas das Collections. Cada template possui uma estrutura particular, e por isso é necessário uma máquina específica que permita seu processamento.

A máquina de processamento de templates do framework proposto é composta por duas máquinas de processamento de estruturas, que possibilitam o processamento de templates em dois formatos: XML e Triplas (N3 e N4). Estas máquinas de processamento permitem a geração de conteúdo nos seguintes formatos da Web Semântica: RDF, RDFa, OWL, triplas, bem como em outros formatos RSS e HTML.

A validação da estrutura do template fica a cargo do usuário, o que significa que, caso seja inserida um template com uma estrutura inválida, teremos um dos seguintes resultados:

- O template não processado;
- Uma exceção causada pela má-formação da estrutura do template;
- Geração de conteúdo em um formato inválido.

6.1.5 Interface por Comando Texto (Textual Command User Interface)

A interface por comando foi criada para usuários mais avançados, e permite o acesso os serviços básicos oferecidos pela Máquina de processamento, através do arquivo de configuração, que pode ser criado manualmente ou através da interface gráfica.

Esta interface permite com que as informações provenientes dos DataStores declarados no arquivo de configuração possam ser convertidas para a estrutura declarada como template.

Para utilizar a interface, o usuário deve utilizar o comando texto *java* seguido de *org.babel.Babel*, como ilustrado pelo fragmento de código a seguir:

"`java org.babel.Babel`"

O comando *java* é utilizado para acionar o interpretador e executar o programa escrito e compilado em Java. Em seguida o usuário poderá passar os seguintes parâmetros:

- *"-c configpath"*: o parâmetro c serve para indicar o caminho onde está o arquivo de configuração, a exemplo:

  - "-c myConfigFile.xml" – indica que o arquivo myConfigFile.xml que se encontra no diretório corrente deverá ser utilizado.

- *"-t templatepath"*: o parâmetro *t* serve para indicar o caminho onde está o template, se o arquivo não for especificado o programa utilizará o primeiro template encontrado no diretório corrente, se ainda assim o arquivo não existir a lista de comandos será exibida, a exemplo:
  - "-t c:\mytemplate.xml" – indica que o arquivo mytemplate.xml que se encontra na raiz do diretório 'c' deverá ser usando como template.
- *"-out outputfile"*: o parâmetro out serve para indicar onde o resultado será escrito, se o arquivo de saída não for especificado o programa escreverá o resultado no arquivo result.xml, a exemplo:
  - "-out c:\myrdf.rdf" – indica que o resultado da conversão deverá ser escrito no arquivo *myrdf.rdf* na raiz do diretório '*c'*.
- *-h*: o comando *h* solicita a exibição da lista de comandos.

6.1.6 Interface Web

A interface Web provê um mecanismo para publicar a informação na Web. Deve ser instalada em um Servidor de Aplicações Java (Tomcat[34], JBOSS[35]). As informações publicadas através da interface Web estarão acessíveis através de requisições HTTP, onde será possível, especificar o formato de serialização da informação. Assim como na Interface por Comando Texto, é necessária a especificação dos arquivos de configuração e do template, sem os quais não será possível realizar a serialização da informação no formato desejado.

A interface padrão não provê um mecanismo de filtrar a informação desejada nem implementa o protocolo SPARQL. Para cada conjunto de meta-informação que o usuário desejar publicar, será necessário a instalação de um novo *endpoint* no servidor.

Cada *endpoint* permite a publicação da informação em três formatos: triplas, RDF/XML e HTML, que deverão ser fornecidos como parâmetros na requisição

[34] http://tomcat.apache.org

[35] http://www.jboss.org

HTTP. Caso o parâmetro seja omitido a informação será serializada em HTML, que é o formato padrão para a serialização de dados na Web.

É possível declarar os DataStores e as Collections programaticamente. A interface atual também pode ser modificada conforme os requisitos do usuário. Nada impede, por exemplo, a adição de novos parâmetros, que possibilitem a definição do conjunto de meta-informações a ser visualizado.

A interface Web é baseada em servlets. Um servlet é um mecanismo que permite chamar dinamicamente uma classe em Java, que é acessada através de um modelo de requisições e resposta. Embora em teoria os servlets possam responder a qualquer tipo de requisição, eles são comumente utilizados em aplicações hospedados em servidores Web para gerar conteúdo HTML e XML. A Figura 24, a seguir ilustra o modelo de requisição e resposta HTTP utilizado pelo endpoint do framework.

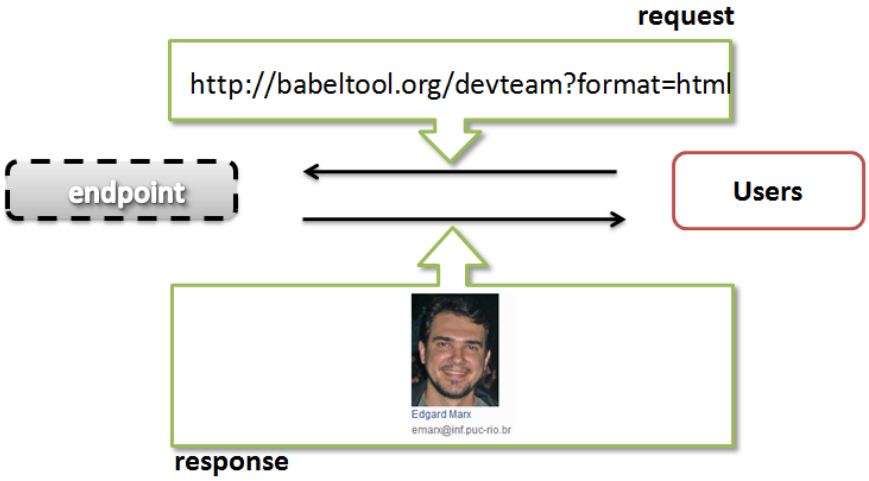

Figura 16. Ilustração do modelo requisição e resposta HTTP utilizado pelo *endpoint* do framework Babel.

6.1.7
Inteface gráfica

A fim de promover a adoção dos Padrões Linked Data e o uso da ferramenta por não-especialistas, desenvolvemos uma interface gráfica que permite definir DataStores, novos tempaltes e criar Collections. Após a definição destes parâmetros, o usuário poderá serializar essa configuração em um arquivo de

configuração do Babel, converter a informação em RDF (ou qualquer outro formato de serialização permitido), bem como publicá-la.

A interface gráfica do Babel, Figuras 17 e 18, também permite a geração automática de templates seguindo heurísticas de mapeamento direto, e a sincronização de um repositório RDF local ou remoto, e.g., repositório RDF Virtuoso, com a base de dados de origem - útil para aqueles que possuem repositórios de dados independentes e desejam mantê-los atualizados.

Apesar de possibilitar a conversão dos DataStores suportados pela Máquina de processamento de templates, a interface gráfica atual só permite manipular e selecionar graficamente coleções do tipo de banco de dados relacionais, que é particularmente interessante por permitir que usuários não familiarizados com SQL e banco de dados possam selecionar a informação ou criar o mapeamento.

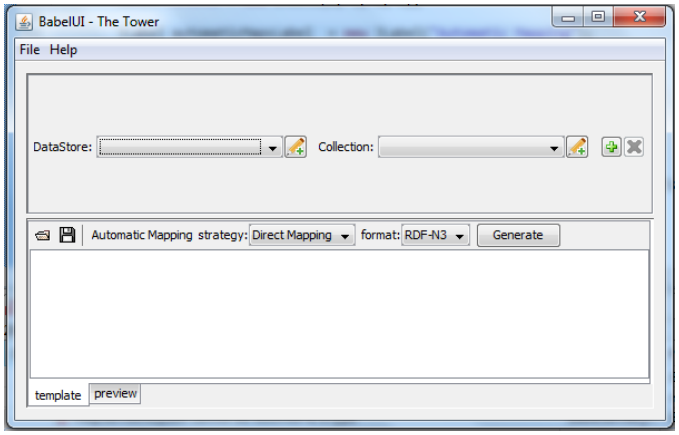

Figura 17. Interface gráfica do framework proposto.

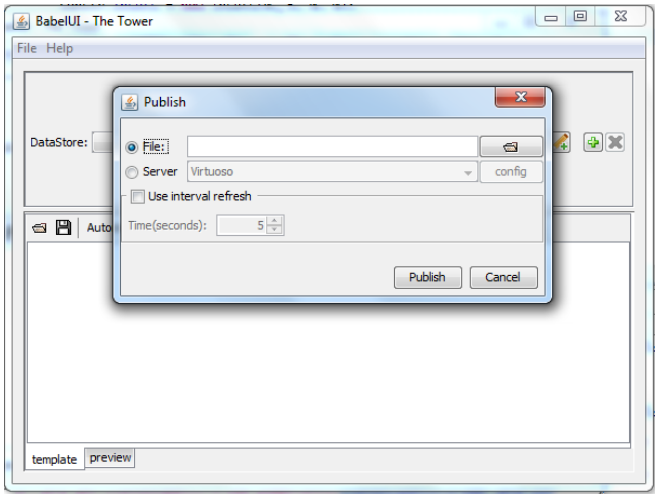

Figura 18. Publicando dados através da interface gráfica.

6.2 Implementação

Nesta seção, serão apresentadas as implementações da Máquina de Porcessamento de Tempaltes e dos DataStores que possibilitaram o desenvolvimento dos vários componentes abordados na seção anterior. Essas implementações podem ser acessadas e obtidas gratuitamente através do endereço do framework, www.babeltool.org.

O framework foi desenvolvido levando em consideração a natureza das necessidades dos usuários. Parte da solução requeria um componente capaz de extrair a informação de uma fonte de dados qualquer e realizar a conversão, com base em um template pré-estabelecido. Os requisitos para este componente estão bem entendidos e bastante estáveis, a parte restante, um componente flexível para lidar com a implementação de uma variedade de fonte de informações que são muito voláteis, como argumentado nas seções anteriores.

A solução encontrada foi separar a implementação em dois módulos distintos. O primeiro módulo converte informações para o formato base do template e é responsável por: (1) realizar a leitura do template contendo o mapeamento, (2) extrair as informações das Collections nos respectivos DataStores, e (3) inserir as informações extraídas das Collections no template e serializá-la. O segundo módulo trata-se de uma ponte, que implementa o suporte a uma grande variedade de fonte de dados e provê um mecanismo que permite a criação de coleções (seleção dos dados que serão inseridos no template) encapsuladas nas Collections. Esta arquitetura separa as duas principais preocupações da implementação proposta, mas ainda mais importante, promove o isolamento do componente responsável pelo acesso à informação que tem uma maior probabilidade de sofrer alteração. A intenção é oferecer uma maneira de atualizar ou possibilitar a adição de outras fontes de dados sem que haja necessidade de alteração no código principal.

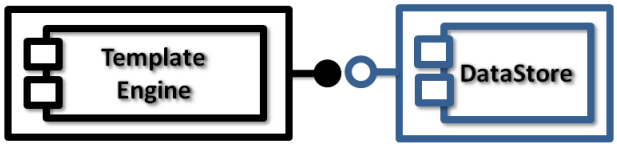

Figura 19. Módulos principais do framework.

Como as fontes de dados estão em constante evolução, qualquer aplicação que implemente o acesso deve ser flexível a ponto de permitir tais mudanças. Dessa forma, as implementações de acesso foram isoladas em um componente separado, acessível através de uma interface, o que permitiu separar as partes da aplicação que são mais estáveis, daquelas que tem mais chances de sofrer modificações. Ao fazermos isso, nós promovemos a reutilização e a evolução do framework.

A manutenção e a adição de novas fontes de acesso é possível através da interface DataStoreFactory que é construída em cima do Service Provider Interface (SPI). A SPI é uma API bem conhecida, amplamente adotada no desenvolvimento de componentes reutilizáveis em diversas tecnologias e.g.: Java Database Connectivity, Java Cryptography Extension, Java Naming and Directory Interface, Java Application Program Interface for XML Processing, e GIS (Geotools), entre outros. A descrição e especificações para SPI são encontrados em [13] [14], respectivamente. O uso de SPI torna a implementação do DataStore resceptivo à mudanças, pois permite a atualização e adição de novos algoritmos através da implementação de uma interface simples na qual os usuários podem substituir (ou modificar) o algoritmo de acesso, sem causar alterações na Máquina de processamento. A Figura 4 mostra a implementação proposta, com destaque para a interface DataStoreFactory.

A interface DataStoreFactory é responsável por selecionar a implementação adequada do DataStore com base nos parâmetros fornecidos pelo usuário, através de dois métodos:

- *canProcess* – Método que recebe como parâmetro as informações configuradas na tag do *DataStore* do arquivo de configuração e é responsável por determinar se o DataStore poderá processar o conjunto de parâmetros fornecidos, retornando *sim*, caso seja, e *não*, caso contrário.
- *createDataStore* – Esse método retorna a implementação adequada do DataStore, com base nos parâmetros fornecidos, que gerarão exceção caso sejam inválidos.

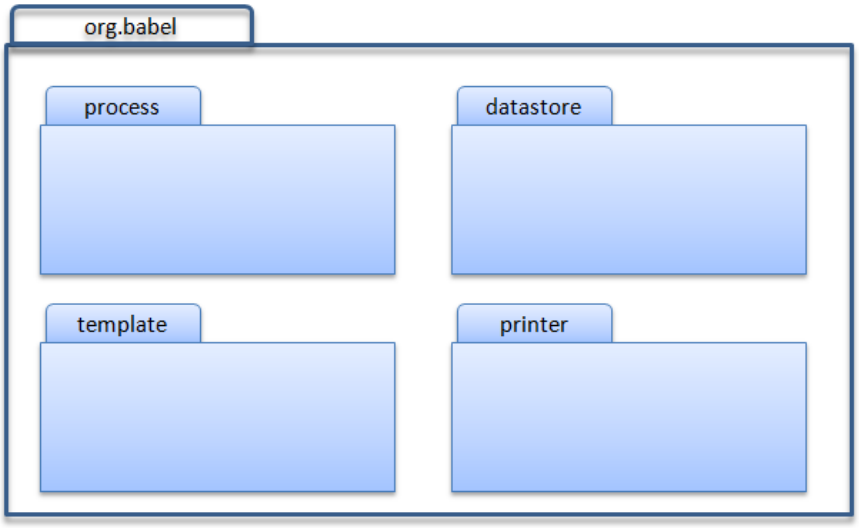

Diagrama 1. Diagrama de pacotes.

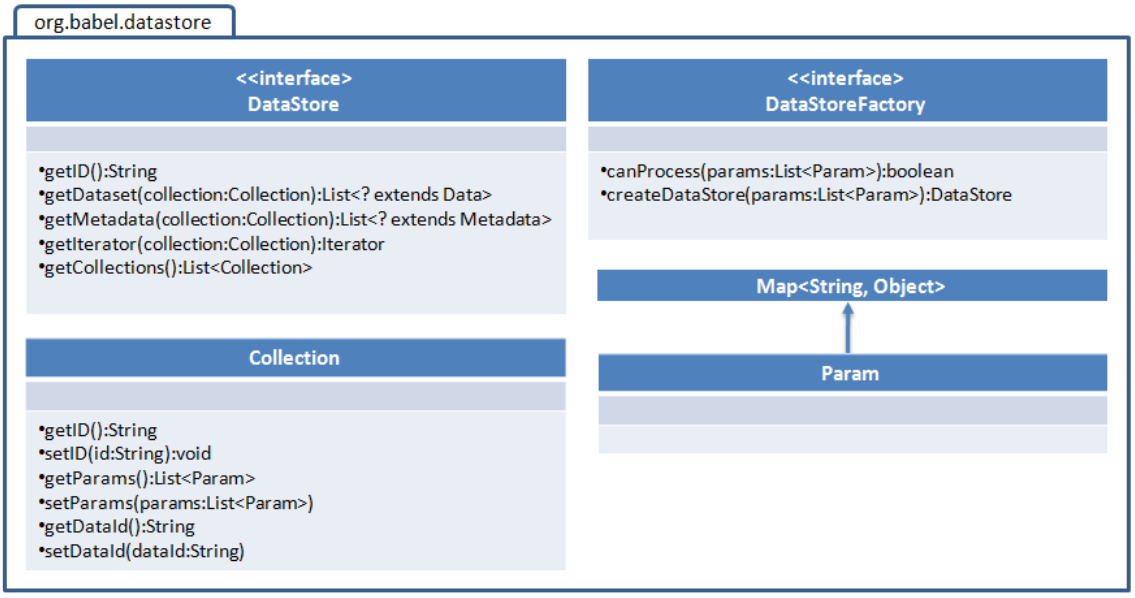

Diagrama 2. Diagrama de classes do pacote *datastore,* do Diagrama 1.

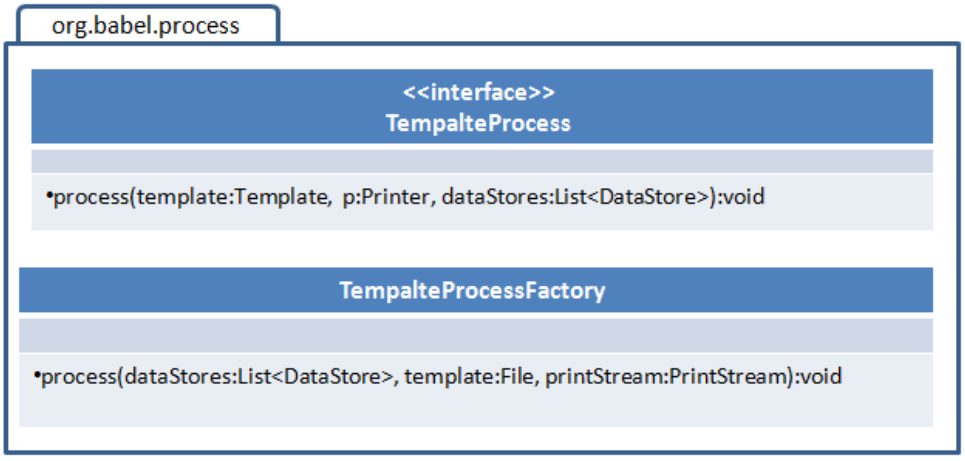

Diagrama 3. Diagrama de classes do pacote *process,* do Diagrama 1.

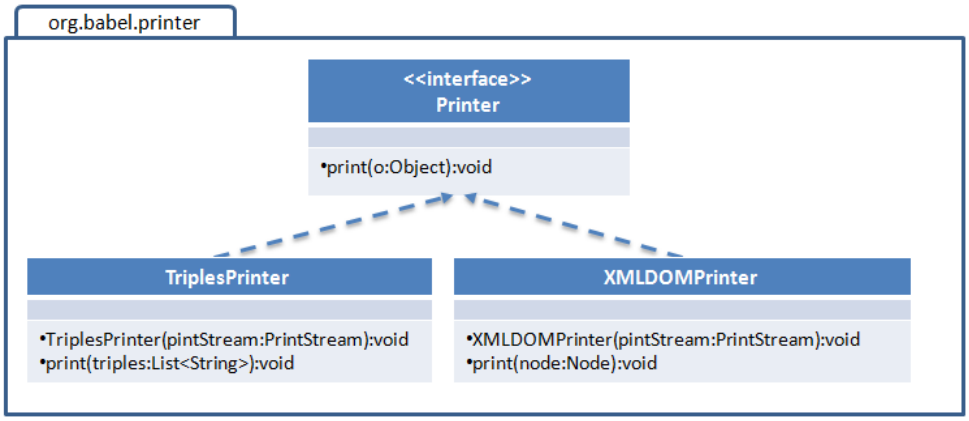

Diagrama 4. Diagrama de classes do pacote *printer,* do Diagrama 1.

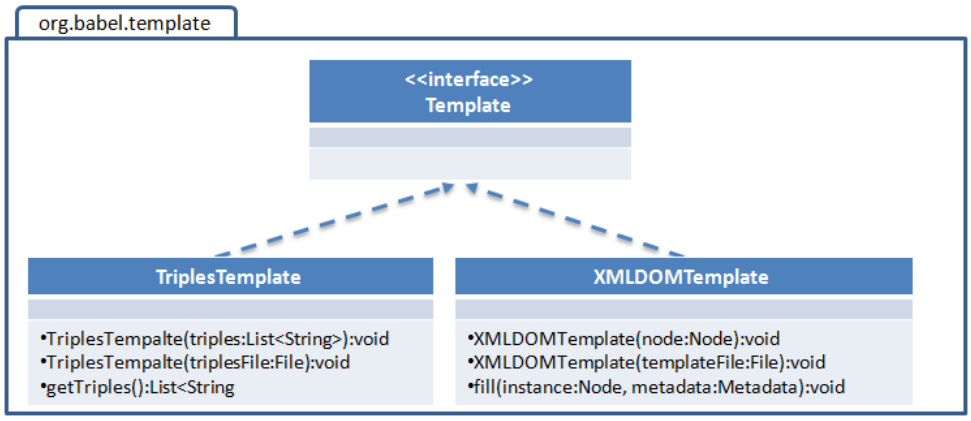

Diagrama 5. Diagrama de classes do pacote *tempalte,* do Diagrama 1.

6.2.1
Máquina de Processamento de Templates (Template Engine)

A máquina de processamento de templates é responsável por inserir os dados extraídos das coleções no template e atualmente ela permite processar tanto templates em XML quanto em triplas. O método de processamento do template pode ser sintetizado em dois passos:

- Coleta dos parâmetros: Nesse passo são colhidas as informações como o DataStores, Collections e o Template, que poderão ser fornecidos através do arquivo de configuração ou declarados programaticamente;
- Processamento: No processamento, os parâmetros obtidos durante a fase de coleta são verificados. Exceções podem ser lançadas caso seja encontrado algum erro nos parâmetros fornecidos que vão desde a estrutura do documento (template, arquivo de

configuração) até existência de algum parâmetro inválido. Caso não seja detectado nenhum erro, inicia-se o processamento dos parâmetros através de uma chamada à API, onde é selecionada a Máquina de Processamento adequada com base no formato do template fornecido.

Cada Máquina de Processamento recebe três parâmetros: o template, uma impressora (Printer) e um conjunto de DataStores. Durante o processamento, os dados são extraídos dos DataStores com base na restrição das Collections, indivíduo a indivíduo, são incorporados ao template, e ao final do processo, o resultado é impresso pela impressora fornecida. Não há uma ordem fixa para da execução das coleções ou impressão do resultado. Cada Máquina de Processamento é livre para implementar o algoritmo de processamento da forma que melhor lhe convir, dessa forma, a ferramenta pode ser facilmente integrada com ferramentas ou sistemas existentes, possibilitando a geração dinâmica de conteúdo para sites ou web services. A execução do processamento do template pode ser sintetizada pelo Diagrama 6.

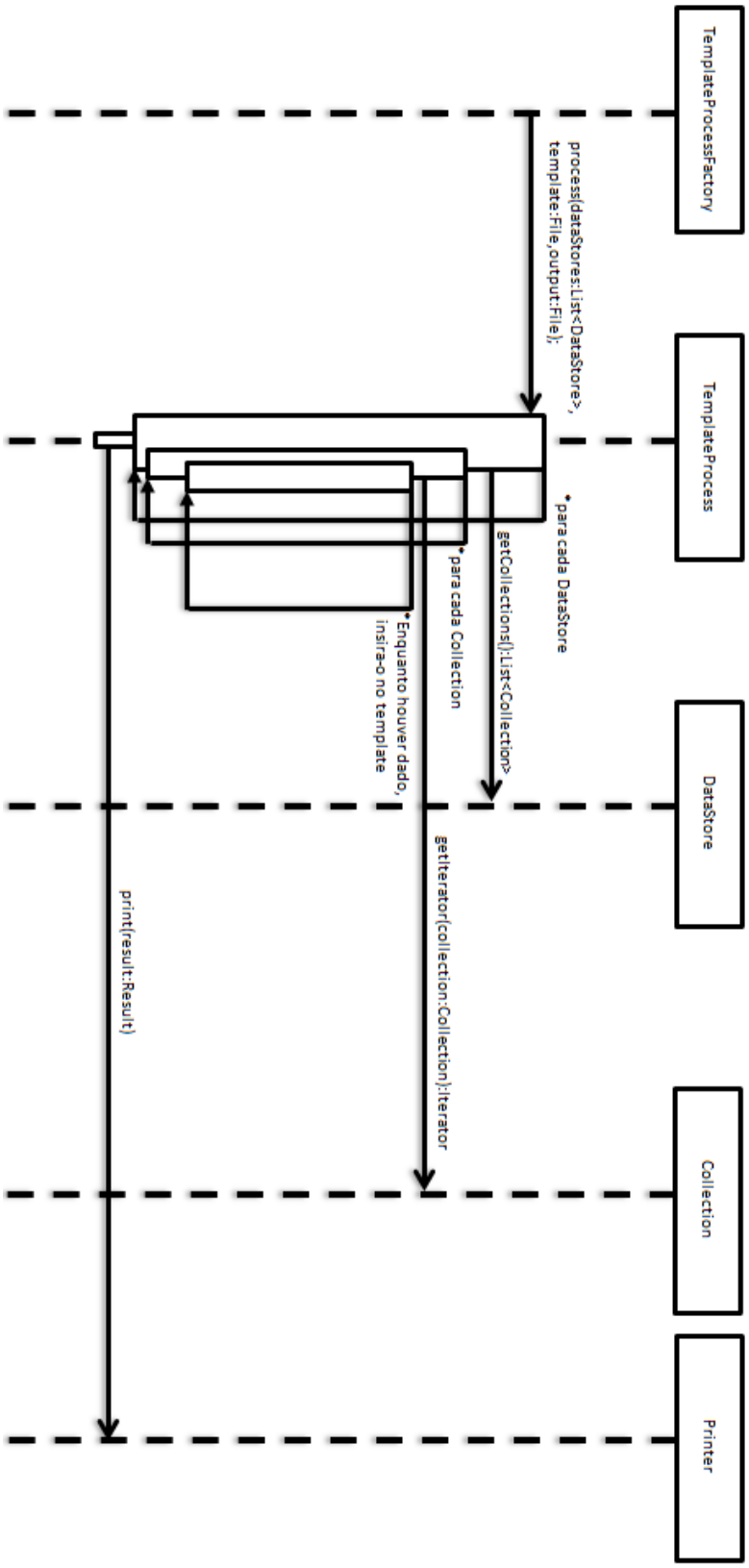

Diagrama 6. Exemplo de um diagrama de execução hipotético de um template pela Máquina de Processamento de Tempaltes.

A máquina de processamento de templates XML funciona através da API DOM, e por isso, possui restrições quanto ao volume de informações que pode manipular. Além disso, o resultado só é impresso após o processamento de todas as Collections. Já a máquina de processamento triplas, não encontra a mesma limitação, uma vez que cada template de tripla é processado e impresso separadamente do conjunto.

6.2.2 DataStore

A interface DataStore é o ponto de acesso da Máquina de Processamento com as diversas fontes de dados. Com ela é possível verificar se o DataStore pode processar determinada coleção, bem como, adquirir sua lista de indivíduos. Essas funcionalidades são possíveis através da implementação de cinco métodos:

- *getID* – retorna o rótulo da fonte de dados;
- *canProcess* – recebe como parâmetro uma coleção e retorna verdadeiro caso possa processá-la e falso, caso contrário;
- *getCollections* – retorna o conjunto de coleções do DataStore;
- *getMetadata* – recebe como parâmetro uma coleção e retorna seu conjunto de meta-informações, tais como: atributos, chaves primárias e estrangeiras, etc.;
- *getDataset* – recebe como parâmetro uma Collection e retorna seu conjunto de indivíduos;
- *getIterator* – recebe como parâmetro uma Collection e retorna seu interador.

Como é observado através do Diagrama 6, e da interface do DataStore, os dados podem ser carregados sob demanda por meio do interador (Iterator), o que possibilita manipular grandes volumes de informações.

6.2.2.1 JDBCDataStore, DataStore de Banco de dados relacionais

O DataStore para Banco de dados permite a extração de dados de uma variedade de implementações de Banco de Dados através do driver JDBC

(SQLServer, Postgres, MySQL e Oracle). É importante notar que para utilizar determinado *driver* é preciso incluí-lo no *classpath* do framework e, além disso, fornecer os parâmetros necessários para a conexão, como: nome do driver, endereço do servidor (host), porta de conexão, senha, usuário e nome do banco, através da interface desejada.

Os filtros das Collections de um DataStore de Banco de Dados são determinados pelo parâmetro *query*, como na Listagem 12. O valor do parâmetro *query* será uma consulta compatível com a versão do driver utilizado pelo DataStore. Dessa forma, o usuário poderá utilizar todas as facilidades que uma consulta possui para selecionar ou até mesmo criar as informações desejadas.

A fim de validar a implementação, foi realizada uma comparação com algumas ferramentas do mercado. Os testes foram realizados sob um banco MySQL em um computador Intel Q6600, contendo 3,23 GB, e consistiu na conversão dos dados de uma tabela em indivíduos *foaf:Person*. Foram gerados 65 mil RDFs foaf:Person contendo apenas nome e email, ilustrados no Gráfico 4. Todas as ferramentas analisadas foram executadas em linha de comando, com exceção do Triplify que foi executado em um servidor PHP. Dentre as ferramentas analisadas, Babel apresentou um desempenho um pouco superior ao segundo colocado, Triplify, evidência que pode ser justificada pela diferença do ambiente de execução e pelo fato de Java ser uma linguagem compilada, ao passo que Triplify é um *script* que é interpretado em tempo de execução. Com esse resultado, Babel demonstrou ser uma ferramenta adequada para a geração de conteúdo RDF para dados provenientes de banco de dados relacionais.

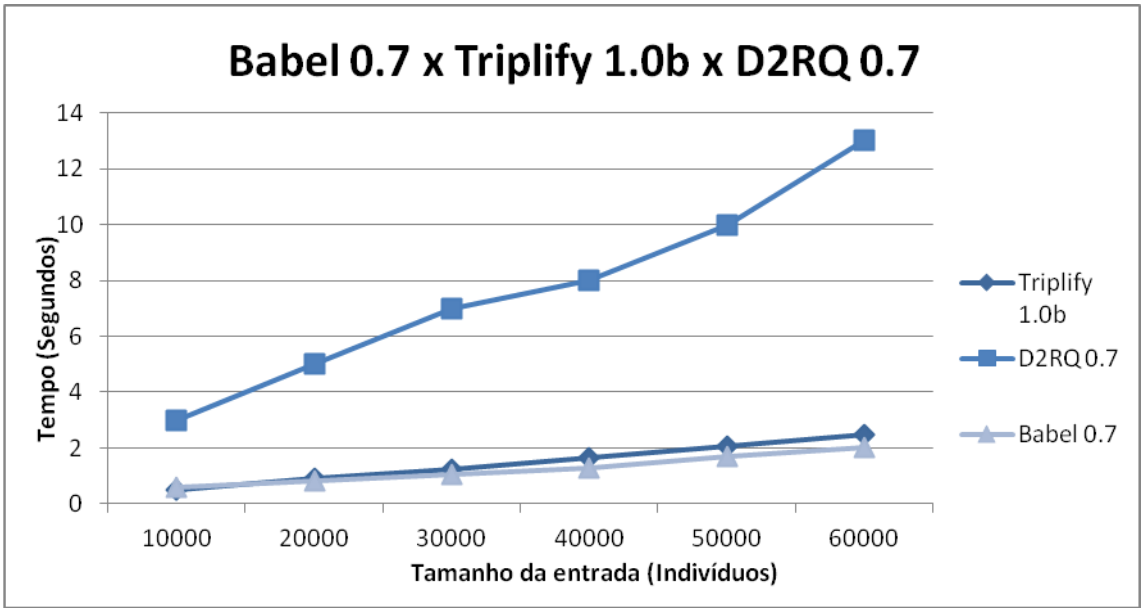

Gráfico 4. Comparação de desempenho entre Babel, Triplify e D2R.

```
<babel-config>
   <dataStore>
        <descriptor>
              <param name="DRIVER_CLASS"
value="sun.jdbc.odbc.JdbcOdbcDriver"/>
              <param name="URL_CONNECTION" value="jdbc:odbc:exemplo"/>
              <param name="PASSWORD" value=""/>
              <param name="USER_NAME" value=""/>
       </descriptor>
       <collection id="table">
          <param name="SQL" value="SELECT * from table "/>
       </collection>
   </dataStore>
</babel-config>
```

Listagem 12. Exemplo de arquivo de configuração declarando um DataStore de banco de dados relacional.

6.2.2.2 POIDataStore, DataStore de planilhas

O POIDataStore permite a extração de dados de planilhas através da API para documentos POI da Apache[36]. Embora a versão atual dê suporte apenas a planilhas, ela também permite extrair conteúdo de outros documentos (Word, Outlook, Powerpoint, Visio, dentre outros).

[36] http://poi.apache.org/ - API de manipulação de documentos da Apache

Para utilizar um DataStore de planilhas em um arquivo de configuração, basta incluir o parâmetro URI, contendo o caminho absoluto para o arquivo da planilha no descritor do DataStore.

Os filtros das Collections de um POIDataStore são determinados pelos parâmetros *ROWSTART*, *ROWEND*, *COLUMNNSTART*, *COLUMNEND*, e *SHEET*, como ilustrado na Figura 20 e Listagem 13. Os parâmetros ROWSTART e ROWEND, definem a linha de início e final respectivamente. Os parâmetros COLUMNSTART e COLUMNEND, as colunas de início e final respectivamente. Como um documento pode ter várias planilhas, o parâmetro SHEET, é utilizado para determinar a planilha que será utilizada.

Como pôde ser observado, o POIDataStore permite a seleção de uma ou mais colunas, bem como uma ou mais linhas, ao contrário de algumas implementações onde só é possível a seleção da coluna [95]. Essa propriedade é particularmente interessante porque a informação pode estar organizada de diferentes formas. Por exemplo, em uma planilha que contenha informações subdivididas em tempo e lugar, os usuários poderão selecionar tanto informações de um lugar quanto de uma data específica.

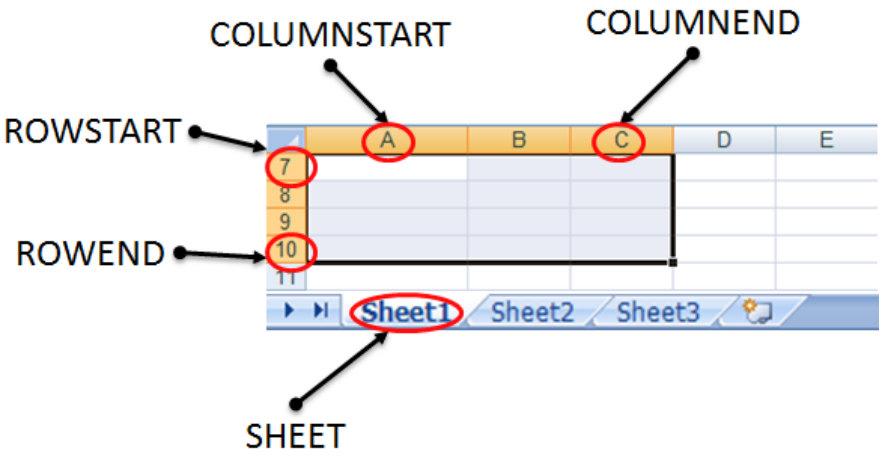

Figura 20. Parâmetros da planilha.

```
<babel-config>
	<dataStore>
		<descriptor>
			<param name="URI" value="C:\...\exemplo.xls"/>
		</descriptor>
		<collection id="empty">
			<param name="SHEET" value="Sheet1"/>
			<param name="ROWSTART" value="7"/>
			<param name="ROWEND" value="10"/>
			<param name="COLUMNSTART" value="0"/>
			<param name="COLUMNEND" value="3"/>
		</collection>
	</dataStore>
<babel-config>
```

Listagem 13. Exemplo de um arquivo de configuração declarando um DataStore de planilha.

7
Discussão e limitações

É incrível como ainda estejamos tentando resolver o mesmo problema de Hollerith no final do século 19. É bem verdade que o objeto mudou. Hollerith em 1889 estudava um mecanismo que possibilitasse a computabilidade do Censo Norte Americano, que em síntese são informações coletadas de indivíduos. A Web Semântica, por outro lado, estuda mecanismos que possibilitem a interligação de grandes conjuntos de dados, oriundos de bases diferentes, mas dada as devidas proporções, ainda são nossas informações.

Muitas conquistas e avanços foram alcançados nos mais de 100 anos que se passaram e a interligação de dados ainda parece ser um problema difícil de resolver. Esta dificuldade deve-se ao fato de que embora a tecnologia possibilite um processamento cada vez maior de informações, ela também possibilita que um número cada vez maior de dados possa ser produzidos, o que pode ser facilmente constatado pelo constante crescimento de dados publicados na Web.

7.1
Geral

Ainda que o framework ofereça uma interface Web para a publicação de informações, a adoção de protocolos e padrões tais como o protocolo SPARQL é um caminho importante na busca pela integração e padronização. Várias ferramentas já oferecem esse recurso, entretanto, muitas delas, não são compatíveis com as bases não-RDF dos usuários.

A utilização de templates abriu espaço não apenas para o mapeamento das bases para RDF, mas também para o caminho inverso, o que permite, em um futuro próximo, a identificação dos conjuntos de dados de uma base relacional em consultas SPARQL.

A Máquina de Processamento baseada em templates mostrou que é possível gerar conteúdo com base em uma Collection definida por uma cláusula restritiva. Um estudo mais aprofundado, utilizando as técnicas de conversão

SPARQLtoSQL existentes, pode levar a uma solução que possibilite a utilização da base relacional original, como exemplificado na Figura 21.

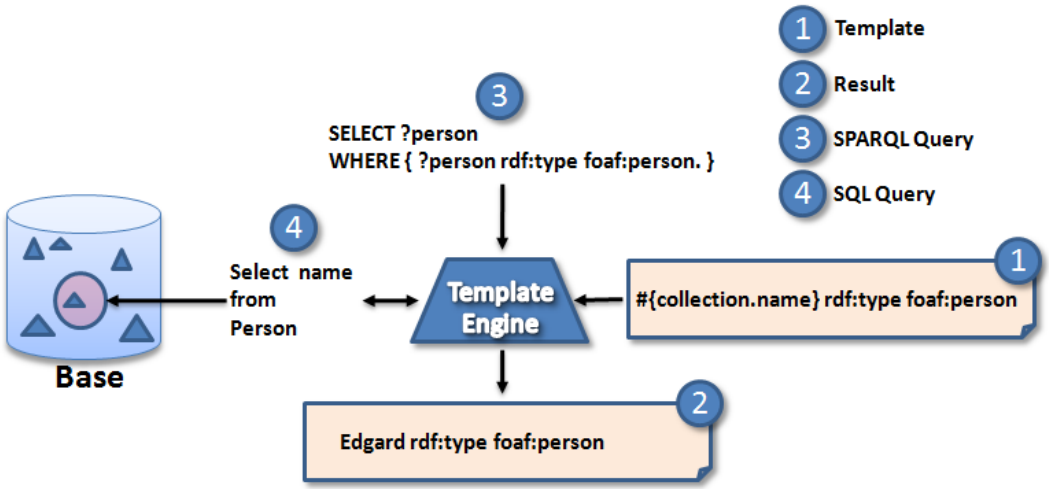

Figura 21. Ilustração da conversão de uma base para RDF e de uma consulta SPARQL para uma consulta SQL.

7.2 Como o culto do amador pode ajudar a Web Semântica

Estamos convencidos do gigantesco número de informações publicada diariamente na internet. É notória a existência de um imenso vazio a ser preenchido por técnicas e ferramentas que possibilitem com que usuários não especialistas contribuam para o crescimento da Web de Dados, entretanto algo que já conhecemos pode ajudar a diminuir esse vazio. Em *O culto do Amador 2007*, Andrew Keen, fala sobre a democratização da informação e de alguém que está revolucionando-a, você.

"A Business 2.0 July 2006 cover story asked who are the fifty people "who matter most" in the new economy. Leading the list was not Steve Jobs or Rupert Murdoch
or Sergey Brin and Larry Page, the two founders of Google. It was "YOU! The Consumer as Creator:

> *You—or rather, the collaborative intelligence of tens of millions of people, the networked you— continually create and filter new forms of content, anointing the useful, the relevant, and the amusing and rejecting the rest. . . . In every case, you've become an integral part of the action as a member of the aggregated, interactive, self-organizing, autoentertaining audience."*

"Who was Time magazine's 2006 Person of the Year? Was it George W Bush, or Pope Benedict XVI, or Bill Gates and Warren Buffett, who together contributed more than $70 billion of their wealth to improving life on earth? E None of the above. Time gave the award to YOU:

Yes, you. You control the Information Age. Welcome to your world."

Andrew Keen, 2007

A Web 2.0 deu a vocês, usuários, o poder não apenas de consumir, mas também de criar conteúdo, e o que isso tem a ver com a Web Semântica? O principal objetivo da Web Semântica é tornar essa imensa quantidade de informação acessível e legível para as máquinas, isso significa colocar toda essa informação em um caminho de links, uma *highway* de links que deverá ser percorrida pelas máquinas, daí a importância de se interligar bases de conhecimento.

Na Web emergente, a informação é identificada e mapeada através de URI´s, o mesmo conjunto de palavras e caracteres que você digita a cada vez que visita uma página na Web, o mesmo conjunto de caracteres que fazem parte dos links. Então pare e comece a pensar no potencial que um *conjunto infinito de macacos* de Andrew Keen teria, se eles pudessem interligar essas informações de uma forma semanticamente mais rica por meio da Web 2.0, o que estamos propondo é potencializar a nova Web utilizando a velha Web.

Isto é algo novo, mas você provavelmente já está, em algum grau, familiarizado. Veja, por exemplo, o caso do Facebook, você pode facilmente vincular o perfil de um amigo ao texto que estiver digitando, o que possibilita que outros usuários possam identificá-lo. Pensemos agora na internet, o que aconteceria se ao ler algum artigo, um catálogo de produtos ou mesmo um blog, pudéssemos vincular algo relevante, qualquer que seja, àquele conteúdo, seja: palavra, frase ou informação. Vários visitantes poderiam vincular conteúdo à página, e o link não mais teria um simples caminho, mas vários caminhos, sugeridos por vários usuários. Então, basicamente um link poderia ser uma espécie de menu, de uma forma que pudéssemos escolher o caminho desejado, talvez o mais clicado, o manual de um produto ou processos judiciais de um político corrupto. Estamos em um tempo de mudanças e a Web está se reinventando a cada dia.

8
Conclusão

O uso dos padrões da Web Semântica, como o RDF e RDFa, na publicação de informações na Web vêm demonstrando ser a única forma viável de garantir a interoperabilidade [26-31] de dados na Web. As soluções existentes, baseadas em padrões da Web Semântica, no entanto, ainda apresentam dificuldades no que tange a escalabilidade, extensão e apoio fornecido aos usuários, o que dificulta a adoção e disseminação da mesma.

Neste cenário propomos Babel, um framework que permite que especialistas, bem como não-especialistas, a converter informações armazenadas em planilhas, bancos de dados relacionais, ou texto, para o formato RDF (N3 e N4), por meio de templates.

Babel possui uma máquina de processamento de templates que permite trabalhar com formatos da família XML (RDFa, HTML, XHTML, RDF), bem como triplas (N3 e N4). Babel é extensível, o que é particularmente interessante para garantir a compatibilidade com outros tipos de fontes de dados, tais como arquivos em formatos proprietários.

Embora os problemas da visualização e da persistência de Linked Data venham sendo estudados como problemas distintos [77], nosso estudo demonstrou que a utilização da heurística baseada em templates permite endereçar ambos.

Nosso trabalho também permite realizar o mapeamento, não apenas de fontes de dados tradicionais (banco de dados relacionais e planilhas), mas também permite extensões de forma a incluir novos formatos de dados.

A fim de facilitar a publicação dos dados por usuários não especialistas, foi desenvolvida uma interface gráfica que facilita a seleção e publicação da informação seguindo uma abordagem orientada a objetos, onde é permitido, dentre outras, a utilização de algoritmos que utilizam técnicas de mapeamento na criação dos templates, além da publicação dos dados em repositórios de triplas bem difundidos, tais como o Virtuoso.

Também foi possível observar, através do Gráfico 3 (Capítulo 3), que a simples conversão da informação em RDF é feita de forma linear e não constitui o gargalo nos repositórios de triplas RDF.

Desta forma, a utilização de uma abordagem SPARQL, baseada na conversão e na tradução da consultas SPARQL para SQL, pode representar uma possível solução para o problema de desempenho. Neste caso, o fator limitador são técnicas eficientes que possibilitem a tradução de consultas SPARQL para SQL [78-79].

8.1 Interligando bases

Há várias razões para que indivíduos, empresas, instituições públicas ou privadas estejam interessadas em publicar suas informações de uma forma acessível e interligada. Pessoas desejam divulgar suas informações e atividades; Empresas desejam que usuários tenham acesso aos seus produtos e serviços; Instituições públicas querem tornar a sua administração transparente; Instituições privadas, interligar suas bases de conhecimento. Estas são algumas razões pelas quais a Web Semântica está tomando um lugar de destaque no cenário atual, isso porque, dentre outras coisas, ela possibilita que informações, antes inacessíveis, possam ser processadas, e vinculadas a um conjunto de outras informações armazenadas em repositórios distintos.

Web services, por exemplo, foram criados para oferecer serviços que possibilitem acessar ou processar informações vindas de uma base de dados própria, de terceiros, ou gerada a partir do processamento de várias bases. No entanto, a tecnologia de web services não descreve uma API de acesso padrão, o que leva, por exemplo, a necessidade de se conhecer diferentes APIs para acessar a informação que está por trás de cada serviço oferecido. De certa forma a Web Semântica resolveu este problema, fornecendo uma API padrão para a consulta e o acesso as fontes de dados, através do protocolo e da linguagem SPARQL. No entanto, ainda existem várias questões relativas a padronização dos modelos, que vêm sendo desenvolvidos de maneira *bottom-up*, ou seja, dos usuários para a indústria.

Há razões para crermos que este problema ainda se perpetuará durante algum tempo. Alguns vocabulários se consagrarão devido à sua maturidade outros pela grande adesão, a exemplo do FOAF[37] e DC[38].

8.2 Mashups

Em paralelo ao contínuo desenvolvimento da Web de hipertexto, testemunhamos um rápido crescimento da Web Dados, onde muitas empresas (Google, Flicker, Facebook, Amazon e outras) começaram a doponibilizar o seu conteúdo através de APIs, com o objetivo de disseminar conteúdo em RDF de uma maneira interligada[39]. Também assistimos ao lançamento do RDFa, tecnologia que permite o acesso e consumo de páginas HTML como fontes de dados estruturados. Apesar de disponível, esta riqueza de informações não é acessível através das formas de busca tradicionais porque estão disponíveis de uma forma não padronizada, em estruturas que variam em forma e conteúdo. É exatamente neste ponto que a técnica de Mashup pode ajudar, agrupando dados de diferentes bases, através do alinhamento de seus vocabulários, permittindo a compreensão do conjunto dos dados. Embora alguns editores de mashups tenham sido propostos (como o Google Mashups, Popfly da Microsoft, IBM sMash, e Yahoo Pipes), sua criação ainda depende de programadores habilidosos.

Em geral, um Mashup é um aplicação Web que consome dados provenientes de outros sistemas, recuperados através de APIs como SPARQL e Webservices, dentre outras. Devido aos diferentes vocabulários utilizados pelos diferentes conjuntos de dados, é difícil encontrar relações e concatenar essas informações. Tim Beners Lee, afirma que o objetivo da Web Semântica é fazer da Web uma uma rede de dados global, interligada e acessível [44]. A possibilidade de realizar Mashups dessas informações é o primeiro passo nessa direção. Para interligar e disponibilizar uma enorme quantidade de dados estruturados na Web, as pessoas devem ser capazes de consultar e interligar os dados de uma maneira

[37] http://xmlns.com/foaf/0.1/

[38] http://purl.org/dc/elements/1.1/

[39] http://www.linkeddata.org

eficiente. Há poucos padrões a este respeito, e pode ser muito difícil entender o contexto, definições, e o histórico dos dados que irão ser trabalhados[101].

É exatamente neste ponto que a ferramenta proposta pode ser útil. Babel permite acessar multiplas fontes de dados através de uma interface padrão, não só isso, as informações extraídas dessas fontes de dados podem ser interligadas através em um só vocabulário, como demonstrado na Figura 22. Dessa forma, por exemplo, dados pronvidos do IBGE poderiam ser interligados e convertidos para o mesmo vocabulário utilizado por Data.gov[40], ou ainda, converter os dados do IBGE e Data.gov para um terceiro vocabulário utilizado, por exemplo, pela DBPedia. O que possibilitaria o *mashup* e o processamento dessas informações, mesmo estando armazenadas em bases distintas e organizadas em estruturas diferentes.

O estado atual da Linked Open Data sugere que interligar a informação não é um processo trivial (apenas 11,5% da informação disponível na LOD está interligada, Tabela 1). Mais ainda, estudos indicam que a Surface Web cresce em escala de bilhares a cada ano, sem contar a gigantesca quantidade de informação que se encontra ainda inacessível na Deep Web.

"At the time of writing there are fifty-three million blogs on the Internet, and this number is doubling every six months."

"YouTube, is a portal of amateur videos that, at the time of writing, was the world's fastest-growing site, attracting sixty-five thousand new videos daily…"

Andrew Keen [96]

Esses números demonstram o imenso abismo que ainda nos separa do modelo da Web de Dados proposto por Tim Beners Lee. Não porque a informação não possa se tornar de fato legível para as máquinas, mas por que ainda não dispomos de mecanismos eficazes que permitam processá-la e interligá-la. Porém, sem sombra de dúvida, a Web Semântica está facilitando o acesso à informação

[40] Sitio de dados abertos oficial do governo dos Estados Unidos

de uma maneira que ela pode ser interligada por terceiros para ser analisada e processada das mais variadas formas.

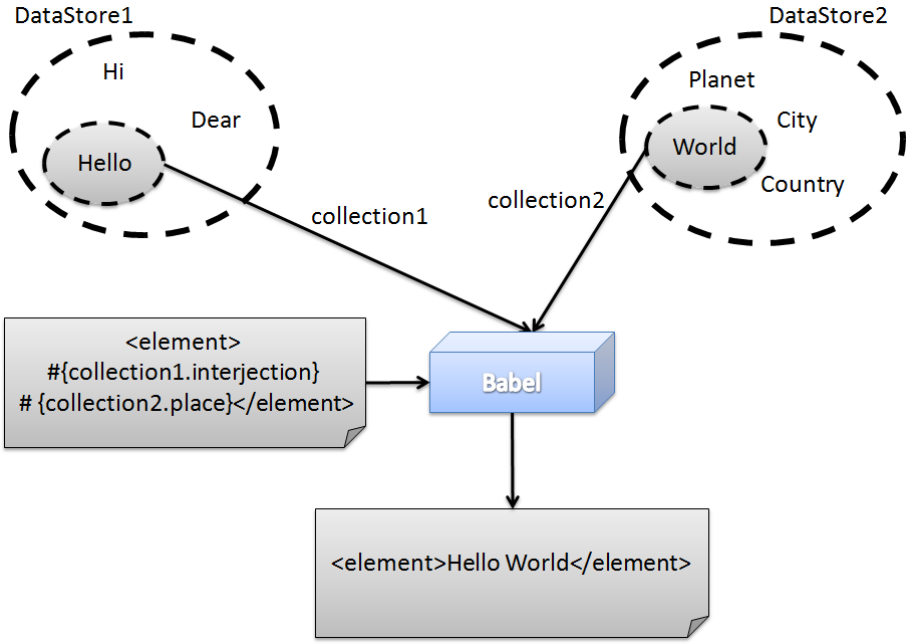

Figura 22. Figura ilustrativa de um Mashup realizado a partir de duas bases de dados utilizando Babel.

8.3 Trabalhos Futuros

Apesar dos resultados obtidos, ainda podemos fazer muito para melhorar a qualidade e usabilidade do Framework, e de seus diversos componentes.

8.3.1 Máquina de Processamento de Templates (Template Engine)

Embora seja possível adicionar novos DataStores de maneira não intrusiva, o processamento de novas estruturas de templates só é possível mediante uma alteração no código da Máquina de Processamento. Para tornar a ferramenta mais flexível seria necessário que as máquinas de processamento também fossem implementadas como plugins.

Outro aspecto importante, diz respeito ao suporte integral das regras estabelecidas pela TML, que possibilitará o uso de estruturas mais complezas

como templates. A versão atual da Máquina de Processamento não permite a utilização da segunda regra.

8.3.2 Interface Gráfica

Por mais que a Interface Gráfica atual tenha contribuído para que usuários pouco familiarizados com as tecnologias de Web Semântica pudessem definir Collections, criar templates e arquivos de configuração, há vários pontos que ainda podem ser melhorados:

- Em linhas gerais, algumas funcionalidades podem ser desenvolvidas de forma extensível, como:
    - A interface de seleção das Collections: a interface de seleção das coleções deverá sofrer alterações a cada nova implementação de DataStore adicionada. Atualmente a adição de novos DataStores só é permitida mediante intervenção manual no código;
    - Estratégias de mapeamento: com o passar do tempo, as estratégias de mapeamento poderão mudar. Estratégias poderão surgir, tornarem-se obsoletas ou simplesmente serem atualizadas. Seria importante que essas estratégias fossem isoladas em um módulo diferente do principal.
    - Repositórios de triplas: existem vários repositórios de triplas que variam quanto ao desempenho, escalabilidade e usabilidade, esses fatores podem levar um usuário a optar pela utilização de um determinado repositório a outro, facilitar a adição de novos repositórios promove uma maior versatilidade da ferramenta.
- A adição de outras técnicas de mapeamento, além do mapeamento direto, como, por exemplo, a utilização do Vocabulário RDF de Dados em Cubos[41] (*RDF Data Cube Vocabulary*) desenvolvido para mapeamento de dados estáticos como arquivos CSV´s e planilhas.

---

[41] http://publishing-statistical-data.googlecode.com/svn/trunk/specs/src/main/html/cube.html#dsd-example

- A seleção de outras coleções que não somente à fonte de banco de dados relacionais.
- Facilitar a criação do template através de uma interface orientada a objetos que possibilite que usuários não familiarizados com as estruturas pudessem criá-las de uma forma mais natural. Nesse sentido, é possível encontrar várias implementações que já foram propostas [30] [95].
- Introduzir algoritmos que selecionem o vocabulário de uma forma automática ou semi-automática, com base no esquema da fonte de dados, o que reduziria a lentidão e a necessidade de intervenções manuais no processo de publicação, que permitiria a usuários desconhecedores do domínio a utilizar a ferramenta.

9
Referências Bibliográficas

[1] Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumuller D. Triplify – Light-Weight Linked Data Publication from Relational Databases, 2008.

[2] C. Bizer and A. Seaborne. D2RQ - treating non-RDF databases as virtual RDF graphs. In ISWC2004 (posters), Novembro de 2004.

[3] C. Blakeley. Rdf views of sql data (declarative sql schema to rdf mapping), 2007.

[4] Sesame: RDF Schema Querying and Storage. Disponível em http://www.openrdf.org/. Acessado em Dezembro de 2010.

[5] A Semantic Web Framework for Java. Disponível em http://jenasourceforge .net/. Acessado em Dezembro de 2010.

[6] O. Erling and I. Mikhailov. RDF support in the Virtuoso DBMS. In Proceedings of the 1st Conference on Social Semantic Web, volume P-113 of GI-Edition - Lecture Notes in Informatics (LNI), ISSN 1617-5468. Bonner K□ollen Verlag, September 2007.

[7] StrixDB - Simple tools to manipulate middle sized RDF graphs. Available at http://www.strixdb.com. Accessed on December 2010.

[8] Virtuoso Sponger. Diponível em http://virtuoso.openlinksw.com/dataspace/ dav/wiki/Main/VirtSponger. Acessado em 2010.

[9] Openlink Supported Standard Non-RDF Data Formats. Available at http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtSpongerCartrid geSupportedDataSourcesNonRDF . Accessed on December 2010.

[10] Virtuoso Sponger Cartridge. Disponível em http://virtuoso.openlinksw.com/ dataspace/dav/wiki/Main/VirtSpongerCartridge. Accessed on December 2010.

[11] A RDF Converter tools index. Disponível em http://esw.w3.org/ ConverterToRdf. Acessado em Dezembro de 2010.

[12] OpenLink Virtuoso Universal Server. Disponível em http://virtuoso. openlinksw.com/pricing/. Acessado em Dezembro 2010.

[13] RDFa Primer - Bridging the Human and Data Webs. W3C Working Group Note 14 October 2008.

[14] RDB2RDF Working Group. Disponível em http://www.w3.org/2001/sw/ rdb2rdf/. Acessado em Dezembro 2011.

[15] Turtle - Terse RDF Triple Language. Disponível em http://www.w3.org/ TeamSubmission/turtle/. Acessado em Dezembro de 2010.

[16] The size of the World Wide Web. Diponível em http://www.world widewebsize.com. Acessado em Dezembro de 2010.

[17] Swoogle, a search engine for Semantic Web ontologies, documents, terms and data published on the Web. Disponível em http://swoogle.umbc.edu. Acessado em Dezembro de 2010.

[18] Bergman, M. - White Paper: The Deep Web: Surfacing Hidden Value.

[19] RDFa in XHTML: Syntax and Processing. Disponível em http://www.w3.org /TR/rdfa-syntax/. Acessado em Dezembro de 2010.

[20] Google Announces Support for Microformats and RDFa. Disponível em http://radar.oreilly.com/2009/05/google-announces-support-for-m.html. Acessado em Dezembro de 2010.

[21] Oracle JDBC Driver Implementation. Disponível em http://www.oracle.com/ technetwork/database/features/jdbc/index-091264.html. Acessado em Dezembro de 2010.

[22] Microsoft SQL Server JDBC Driver Implementation. Disponível em http://www.microsoft.com/downloads/en/details.aspx?FamilyID=e22bc83b-32ff-4474-a44a22b6ae2c4e17&DisplayLang=en. Acessado em Dezembro de 2010.

[23] Postgres JDBC Driver Implementation. Disponível em http://jdbc.postgresql. org/download.html. Acessado em December 2010.

[24] MySQL JDBC Driver Implementation. Disponível em http://dev.mysql.com/ downloads/connector/j/. Acessado em December 2010.

[25] Foaf Vocabulary Specification. Disponível em http://xmlns.com/foaf/spec/. Acessado em 2011.

[26] SPARQL Query Language for RDF. Disponível em http://www.w3.org/ TR/rdf-sparql-query/. Accessed on December 2010.

[27] Breitman, Karin K ; Casanova, M. A. ; Truszkowski, W. "Semantic Web: Concepts, Technologies and Applications". Londres: Springer, 2006. v. 1. 337 p.

[28] Casanova, M.A., Lauschner, T., Leme, L., Breitman, K., Furtado, A., Vidal, V. "A Strategy to Revise the Constraints of the Mediated Schema". ER 2009: 265-279.

[29] Leme, L.A.; Casanova, M.; Breitman, K.; Furtado, A. "OWL schema matching" Journal of Brazilian Computer Society, Springer Verlag 16(1): 21-34 (2010).

[30] Salas, P., Breitman, K., Viterbo J. and Casanova, M.A. "Interoperability by Design Using the Std-Trip Tool: an a priori approach"(Triplification Challenge Submission), In Proceedings of the 6th International Conference on Semantic Systems 2010 (ISEMANTICS' 10).

[31] Tim Berners-Lee - Putting Government Data online. Disponível em http://www.w3.org/DesignIssues/GovData.html. Acessado em 2010.

[32] Linked Data – Connect Distributed Data Across the Web. Disponível em http://linkeddata.org/. Acessado em December 2010.

[33] The Linked Open Data Community. Disponível em http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData. Acessado em Dezembro de 2010.

[34] Publishing open government data (2009). Disponível em http://www.w3.org/TR/gov-data/. Acessado em 2010.

[35] Bizer, C., Heath, T., Ayers, D. and Raimond, Y. Bizer, C., Heath, T., Ayers, D., Raimond, Y. "Interlinking Open Data on the Web"; Demonstrations Track at the 4th European Semantic Web Conference, Innsbruck, Austria. May 2007. Disponível em http://www.eswc2007.org/pdf/demo-pdf/LinkingOpenData.pdf. Acessado em Dezembro de 2010.

[36] Judith Bishop, David Notkin, Karin Breitman: First workshop on developing tools as plug-ins: (TOPI 2011). ICSE 2011: 1230-1231

[37] Charles W. Krueger. 1992. Software reuse. ACM Comput. Surv. 24, 2 (June 1992), 131-183.

[38] Pavel Shvaiko and Jerome Euzenat. 2008. Ten Challenges for Ontology Matching. In Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems (OTM '08), Robert Meersman and Zahir Tari (Eds.). Springer-Verlag, Berlin, Heidelberg

[39] Percy E. Salas, Karin Koogan Breitman, José Viterbo F., Marco A. Casanova: Interoperability by design using the StdTrip tool: an a priori approach. I-SEMANTICS 2010.

[40] A Direct Mapping of Relational Data to RDF, Marcelo Arenas, Eric Prud'hommeaux, Juan Sequeda, Editors. World Wide Web Consortium, 24 de Março de 2011. Disponível em http://www.w3.org/TR/rdb-direct-mapping/.

[41] Greg Goth, "Reaping Deep Web Rewards Is a Matter of Semantics," IEEE Internet Computing, vol. 13, no. 3, pp. 7-10, Maio/Junho de 2009.

[42] Jensen, R.J., Szulanski, G. Template Use and the Effectiveness of Knowledge Transfer. Journal Management Science, Volume 53 Issue 11, Novembro de 2007. Disponível em http://portal.acm.org/citation.cfm?id=1527720.1527725&CollDL&dl=GUIDE&CFID=28381526&CFTOKEN=22882858. Acessado em Dezembro de 2010.

[43] Tim Berners-Lee, James Hendler, and Ora Lassila, "The Semantic Web," in *Scientific American* 284(5): pp 34-43, 2001. See http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2.

[44] Ian Horrocks, "Semantic Web: The Story So Far", University of Manchester, Manchester, UK.

[45] Tim Berners-Lee, "Semantic Web on XML," at *XML 2000*, 6 de Dezembro, Washington, DC. See http://www.w3.org/2000/Talks/1206-xml2k-tbl

[46] Resource Description Framework (RDF) Model and Syntax Specification, W3C Proposed Recommendation 05 January 1999. Disponível em http://www.w3.org/TR/PR-rdf-syntax/. Acessado em 2010.

[47] Resource Description Framework (RDF), Disponível em http://www.w3.org/RDF/. Acessado em 2010.

[48] RDFa in XHTML: Syntax and Processing, A collection of attributes and processing rules for extending XHTML to support RDF. W3C Recommendation 14 October 2008. Disponível em http://www.w3.org/TR/rdfa-syntax/. Acessado em 2010.

[49] RDFa Primer, Bridging the Human and Data Webs. W3C Working Group Note 14 October 2008. Disponível em http://www.w3.org/TR/xhtml-rdfa-primer/. Acessado em 2010.

[50] R2RML: RDB to RDF Mapping Language. W3C Working Draft 20 September 2011. Disponível em http://www.w3.org/TR/r2rml/. Acessado em 2010.

[51] Semantics of R2RML. Disponível em http://www.w3.org/2001/sw/rdb2rdf/wiki/Semantics_of_R2RML. Acessado em 2010.

[52] RDB2RDF Working Group Charter. Disponível em http://www.w3.org/2009/08/rdb2rdf-charter.html. Acessado em 2010.

[53] Percy E. Salas, Edgard Marx, Alexander Mera, José Viterbo - RDB2RDF Plugin: Relational Databases to RDF plugin for Eclipse, ACM New York, NY, USA ©2011.

[54] Tim Bray**,** Jean Paoli, C. M. Sperberg-McQueen - Extensible Markup Language (XML) 1.0 W3C Recommendation 10 February 1998

[55] Document Object Model (DOM). Disponível em http://www.w3.org/DOM/. Acessado em 2010.

[56] SAX, *Simple API for XML*. Disponível em http://www.saxproject.org/. Acessado em 2010.

[57] Polanyi, Michael. 1962. *Personal knowledge: Towards a post-critical philosophy.* Chicago University Press, Chicago. IL.

[58] Nelson, Richard, Sidney Winter. 1982. *An evolutionary theory of economic change.* Belknap Press, Cambridge.

[59] Lippman, S. A.,R. P. Rumelt. 1982. Uncertain imitability: An analysis of interfirm differences in efficiency under competition. *Bell J. Economics*. 13 418-438.

[60] Datasets in the next LOD Cloud. Diponível em http://www4.wiwiss.fu-berlin.de/lodcloud/. Acessado em 2012.

[61] State of the LOD Cloud. Disponível em http://www4.wiwiss.fu-berlin.de/lodcloud/state/. Acessado em 2012.

[62] M. A. Casanova, K. Breitman, D. Brauner and A. Mar-ins. Database conceptual schema matching. IEEE Computer.

[63] C. Bizer, R. Cyganiak and T. Heath. How to publishlinked data on the web. Retrieved December 14, 2010. Disponível em http://www4.wiwiss. fuberlin.de /bizer/pub/LinkedDataTutorial/, 2007.

[64] K. Breitman, M. A. Casanova and W. Truszkowski. Se-mantic web: Concepts, technologies and applications (nasa mono-graphs in systems and

software engineering). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[65] Christian Bizer, Andy Seaborne: D2RQ -Treating Non-RDF Databases as Virtual RDF Graphs (Poster). 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, November 2004.

[66] Christian Bizer, D2R MAP - A Database to RDF Mapping Language. The Twelfth International World Wide Web Conference (WWW2003), Budapest, Hungary, May 2003.

[67] O. Erling and I. Mikhailov. Rdf support in the virtuoso dbms. Networked Knowledge-Networked Media, pages 7(24, 2009).

[68] J. Broekstra and A. Kampman. Sesame: A generic architecture for storing and querying RDF and RDF schema, October 2001. Disponível em http://sesame.aidministrator.nl/publications/del10.pdf.

[69] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. The Jena Semantic Web Platform: Architecture and design. Technical report, Hewlett Packard Laboratories, Palo Alto, California, 2003.

[70] Server-Side Support for Relational/XML Integration (FOR XML/OPENXML). Diponível em http://msdn.microsoft.com/en-US/library/ms345110%28v=SQL.90%29.aspx#sql_2k5xmlopt_topic5. Acessada em 2011.

[71] XML Data Type in SQL Server. Diponível em http://msdn.microsoft.com/en-US/library/ms345110%28v=SQL.90%29.aspx#sql_2k5xmlopt_topic6. Acessada em 2010.

[72] Greg Goth, "Reaping Deep Web Rewards Is a Matter of Semantics," IEEE Internet Computing, vol. 13, no. 3, pp. 7-10, May/June 2009

[73] Oracle Database Semantic Technologies. Diponível em http://www.oracle.com/technetwork/database/options/semantic-tech/index.html. Acessada em 2011.

[74] Tim Berners-Lee. Putting Government Data online. Diponível em http://www.w3.org/DesignIssues/GovData.html. Acessado em 2011.

[75] Mark H. Butler, Barriers to real world adoption of Semantic web Technologies**.**

[76] H. Hollerith. An Eletric Tabulating System. Diponível em http://www.columbia.edu/cu/computinghistory/hh/index.html. Acessada em 2011.

[77] LESS - Template-Based Syndication and Presentation of Linked Data Soren Auer, Raphael Doehring, and Sebastian Dietzold. Disponível em http://www.informatik.uni-leipzig.de/~auer/publication/semtem.pdf. Acessado em 2011.

[78] R. Cyganiak. A relational algebra for SPARQL. Technical Report HPL-2005-170. 2005. Diponível em http://www.hpl.hp.com/techreports/2005/HPL-2005-170.html. Acessado em 2011.

[79] S. Harris and N. Shadbolt. SPARQL query processing with conventional relational database systems. In *SSWS*, 2005.

[80] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. In *ISWC*, 2002.

[81] A. Chebotko, S. Lu, H. M. Jamil, and F. Fotouhi. Semantics Preserving SPARQL-to-SQL Query Translation for Optional Graph Patterns. Technical Report TR-DB-052006-CLJF. May 2006. http://www.cs.wayne.edu/ »artem/main/research/TR-DB-052006-CLJF.pdf.

[82] Huynh, D., Karger, D., Miller, R.: Exhibit: lightweight structured data publishing. In: Proceedings of the 16th international conference on World Wide Web, pp. 737–746. ACM Press, New York (2007).

[83] The Punched Card Tabulator. Herman Hollerith's first tabulating machines opened the world's eyes to the very idea of data processing. Along the way, the machines also laid the foundation for IBM. Diponível em http://www.ibm.com/ibm100/us/en/icons/tabulator/. Acessado em 2011.

[84] Pavel Shvaiko and Jerome Euzenat. 2008. Ten Challenges for Ontology Matching. In Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems (OTM '08), Robert Meersman and Zahir Tari (Eds.). Springer-Verlag, Berlin, Heidelberg.

[85] Percy E. Salas, Karin Koogan Breitman, José Viterbo F., Marco A. Casanova: Interoperability by design using the StdTrip tool: an a priori approach. I-SEMANTICS 2010.

[86] A Direct Mapping of Relational Data to RDF, Marcelo Arenas, Eric Prud'hommeaux, Juan Sequeda, Editors. World Wide Web Consortium, 24 March 2011. Disponível em http://www.w3.org/TR/rdb-direct-mapping/.

[87] M. Arenas; E. Prud'hommeaux and J. Sequeda, Editors, A Direct Mapping of Relational Data to RDF. W3C RDB2RDF Working Group. Disponível em http://www.w3.org/TR/rdb-direct-mapping/.

[88] S. S. Sahoo,W. Halb, S. Hellmann, K. Idehen, T. Thibodeau Jr, S. Auer, J. Sequeda and A. Ezzat. A survey of current approaches for mapping of relational databases to rdf. W3C RDB2RDF Incubator Group report, 2009.

[89] Das. S., Sundara S., Cyganiak, R., R2RML: RDB to RDF Mapping Language. W3C Working Draft 20 de Setembro 2011. Disponível em http://www.w3.org/TR/2011/WD-r2rml-20110920/.

[90] Das. S., Sundara S., Cyganiak, R., R2RML: RDB to RDF Mapping Language. W3C Working Draft 24 de Março 2011. Disponível em http://www.w3.org/TR/2011/WD-r2rml-20110324/.

[91] Das. S., Sundara S., Cyganiak, R., R2RML: RDB to RDF Mapping Language. W3C Working Draft, 28 Outubro, 2010. Disponível em http://www.w3.org/TR/2010/WD-r2rml-20101028/.

[92] Charles W. Krueger. 1992. Software reuse. ACM Comput. Surv. 24, 2 (June 1992), 131-183.

[93] Jensen, R.J., Szulanski, G. Template Use and the Effectiveness of Knowledge Transfer. Journal Management Science, Volume 53 Issue 11, November 2007. Disponível em http://portal.acm.org/citation.cfm?id=1527720.1527725

&coll=DL&dl=GUIDE&CFID=28381526&CFTOKEN=22882858.
Acessado em Dezembro de 2010.

[94] Judith Bishop, David Notkin, Karin Breitman: First workshop on developing tools as plug-ins: (TOPI 2011). ICSE 2011: 1230-1231

[95] Lushan Han, Tim Finin, Cynthia Parr, Joel Sachs, and Anupam Joshi. RDF123: From Spreadsheets to RDF. In 7th International Semantic Web Conference (ISWC2008), October 2008.

[96] Keen, A. (2007a). The Cult of the Amateur: How Today's Internet Is Killing Our Culture and Assaulting Our Economy. London: Nicholas Brealey Publishing.

[97] D. Merrill. Mashups: The new breed of Web app. Website, 08 2006. Disponível em http://www.ibm.com/developerworks/library/x-mashups.html; Acessado em Março de 2010.

[98] E. Marx, P. E. Salas, K. Breitman, and J. Viterbo, RDB2RDF Plugin: Relational Databases to RDF Plugin for Eclipse, 2011.

[99] Cyganiak, R., Harth, A., Hogan, A.: N-quads: Extending n-triples with context. http://sw.deri.org/2008/07/n-quads/ (2008)

[100] Bizer, C.: D2R MAP-A Database to RDF Mapping Language. In: Proceedings of the 12th International World Wide Web Conference (May 2003), pp. 20–24 (2003)

[101] Integrating Open Government Datasets to Create Mashups. Karin Breitman, Percy Salas, Daniel Saraiva, Vinícius Gama, Marco A. Casanova.

[102] W3C Opens Data on the Web with SPARQL. Diponível em http://www.w3.org/2007/12/sparql-pressrelease. Acessado em 2011.