

UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA  
CURSO DE SISTEMAS DE INFORMAÇÃO

GUILHERME NANDI TISCOSKI  
JORGE RENATO BORHNAUSEN DE SÁ

DISPONIBILIZAÇÃO DE DADOS GEOGRÁFICOS DO BRASIL UTILIZANDO OS  
PRINCÍPIOS DE LINKED DATA

FLORIANÓPOLIS – SC  
2011

UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA  
CURSO DE SISTEMAS DE INFORMAÇÃO

GUILHERME NANDI TISCOSKI  
JORGE RENATO BORHNAUSEN DE SÁ

DISPONIBILIZAÇÃO DE DADOS GEOGRÁFICOS DO BRASIL UTILIZANDO OS  
PRINCÍPIOS DE LINKED DATA

ORIENTADOR: PROF. DR. FERNANDO OSTUNI GAUTHIER  
COORIENTADOR: PROF. DR. JOSÉ LEOMAR TODESCO

Trabalho de conclusão de curso  
apresentado como parte dos requisitos  
para obtenção do grau de  
Bacharel em Sistemas de informação

FLORIANÓPOLIS – SC  
2011

Guilherme Nandi Tiscoski  
Jorge Renato Bornhausen de Sá

## **DISPONIBILIZAÇÃO DE DADOS GEOGRÁFICOS DO BRASIL UTILIZANDO OS PRINCÍPIOS DE LINKED DATA**

Este Trabalho de conclusão de curso foi julgado adequado para obtenção do Título de Bacharel em Sistemas de Informação, e aprovado em sua forma final pelo Curso de Graduação em Sistemas de Informação.

Florianópolis, dezembro de 2011.

---

PROF. DR. LEANDRO JOSÉ KOMOSINSKI  
Coordenador do Curso

Banca examinadora:

---

PROF. DR. FERNANDO OSTUNI GAUTHIER  
Orientador

---

PROF. DR. JOSÉ LEOMAR TODESCO  
Coorientador

---

PROF. DR. RONALDO DOS SANTOS MELLO  
Membro da banca avaliadora

## **Dedicatória**

Este trabalho é dedicado aos nossos pais, os quais sempre se esforçaram para prover o melhor ensino possível aos seus filhos.

# **Agradecimentos**

Agradecemos aos nossos amigos mais próximos, namoradas e todos aqueles que de alguma maneira, direta ou indireta, ajudaram no desenvolvimento deste trabalho.

## Resumo

Dados abertos (Open Data, originalmente) são uma vertente da tecnologia da informação que vem crescendo muito na última década, principalmente nos últimos cinco anos. Seu conceito se baseia na ideia de que alguns tipos de informação devem ser disponibilizados gratuitamente, de forma que qualquer pessoa ou instituição possa ter acesso e utilizar da forma que bem entender, sem restrições de direitos autorais ou qualquer mecanismo de controle. O objetivo desse movimento é criar uma cultura coletiva de construção, agrupamento e publicação de dados.

O conceito de web semântica surgiu ao público em 2001 com a publicação de um artigo na revista Scientific America com autoria de Tim Berners-Lee, James Hendler e Ora Lassila. Seus principais objetivos são: definição de padrões de formatação para a integração e combinação de dados provenientes de diversas fontes; atribuição de significado, mostrando como os dados se relacionam com objetos do mundo real. Portanto, se um dos objetivos da web semântica é a integração entre diversas fontes de dados, visando enriquecer as informações disponíveis, pode-se dizer que Linked Data é um dos pilares que a sustenta.

O termo Linked Data representa um conjunto de boas práticas para conectar e publicar dados estruturados na Web (BERNERS-LEE et al, 2009). A adoção dessas boas práticas tem conduzido à criação de um data space global, no qual informações de diversas fontes e domínios diferentes são relacionadas.

Este trabalho apresenta uma abordagem teórico-prática dos conceitos de publicação de dados abertos, seguindo os princípios de Linked Data. Como prova da aplicação destes conceitos disponibilizamos um conjunto de dados que seguem estes padrões, além de uma aplicação que os consome.

**Palavras-chave:** Dados Abertos. Linked Data. Web Semântica. Data Space Global. Publicação de dados abertos.

## Resumo em inglês

Open data is a part of information technology that has been growing over the past decade, especially in the last five years. Its concept is based on the idea that some types of information should be freely available, so that any person or institution can access and use it in any way they see fit, without restriction of copyright or other control mechanisms. The aim of this movement is to create a corporate culture of construction, assembly and publication of data.

The concept of semantic web has emerged publicly in 2001 with the publication of an article in Scientific American authored by Tim Berners-Lee, James Hendler and Ora Lassila. Its main objectives are: definition of format patterns for the integration and combination of data from various sources and the attribution of meaning to the data, showing how it relates to real world objects. So if one of the goals of the semantic web is the integration of different data sources to enrich the information available, we can say that Linked Data is a pillar that supports it.

The term Linked Data represents a set of good practices to connect and publish structured data on the Web (Berners-Lee et al, 2009). The adoption of these practices has led to the creation of a global data space, in which information from various sources and different domains are related.

This paper presents a theoretical and practical approach of the concepts of publishing open data following the principles of Linked Data. As evidence of the application of these concepts, we provide a set of data that follow these patterns, and an application that uses this data set.

**Key-Words:** Open Data. Linked Data. Semantic Web. Global Data Space. Open Data publishing.

# Lista de Figuras

Figura 1 – Exemplo de declaração de Namespaces em OWL

Figura 2 – Exemplo de declaração de classes em OWL

Figura 3 - Perfil de Richard Cyganiak e uma pesquisa por informações relacionadas à cidade de Berlim.

Figura 4 - Resultado obtido através da pesquisa realizada na Figura 1

Figura 5 – Evolução dos data sets de linked data entre 2007 e 2009

Figura 6 - Evolução dos data sets de linked data entre 2010 e 2011.

Figura 7 – Formato de uma tripla RDF

Figura 8 – Exemplo de documento RDF+XML

Figura 9 – Exemplo de documento HTML+RDF

Figura 10 – Comparação entre RDF/XML e N3

Figura 11 – Diagrama de publicação de linked data

Figura 12 – Exemplo de conteúdo de um arquivo de mapeamento D2RQ

Figura 13 – Exemplo de mapeamento entre base relacional e ontologia utilizando o NeOn Toolkit com plugin ODEMapster

Figura 14 – Arquitetura de funcionamento do Pubby.

Figura 15 – Exemplo de consulta SPARQL do tipo SELECT

Figura 16 – Resultados de uma busca por “Florianópolis” no SWSE

Figura 17 – Camadas da arquitetura de uma aplicação mashup

Figura 18 – Query SPARQL para obter-se a lista de URIs, atrelada ao nome, dos municípios catarinenses

Figura 19 – Parte do resultado da pesquisa, no formato HTML

Figura 20 - Modelo relacional final Fonte: Autores

Figura 21 – Ontologia GeoPoliticaBr Fonte: Autores

Figura 22 – Ontologia DadosBr Fonte: Autores

Figura 23 – Arquivo de mapeamento Fonte: Autores

Figura 24 – Arquivo RDF Fonte: Autores

Figura 25 – Interface fornecida pelo D2R Server Fonte: Autores

Figura 26 – Página inicial do Sesame Server Fonte: Autores

Figura 27 – Criação de um repositório Fonte: Autores

Figura 28 – Upload do arquivo RDF Fonte: Autores



Figura 29 – Página de consulta fornecida pelo Sesame Fonte: Autores

Figura 30 – Resultado de uma consulta sobre os dados do servidor Fonte: Autores

Figura 31 – Arquivo de configuração do Pubby Fonte: Autores

Figura 32 – Aplicação que consome os dados publicados Fonte: Autores

Figura 33 – Tela de detalhamento para um município Fonte: Autores

# Lista de Tabelas

Tabela 1 - Fontes de dados encontradas para utilização neste trabalho Fonte: Autores

# Lista de Siglas

API – Application Programming Interface  
CRUD - Create, Read, Update, and Delete  
CSV – Comma Separated Values  
ETL – Extract, Transform, Load  
FOAF - Friend-of-a-Friend  
HTML - HyperText Markup Language  
HTTP - HyperText Transfer Protocol  
IBGE - Instituto Brasileiro de Geografia e Estatística  
IDE – Integrated Development Environment  
JSON – JavaScript Object Notation  
JSON-RPC – JSON-Remote Procedure Call  
KML – Keyhole Markup Language  
LOD - Linked Open Data  
OGP – Open Government Partnership  
ORDBMS - Object-Relational Database Management System  
OWL - Ontology Web Language  
RDBMS - Relational Database Management System  
RDF - Resource Description Framework  
REST – Representational state transfer  
RPC – Remote Procedure Call  
SCOVO - Statistical Core Vocabulary  
SeRQL - Sesame RDF Query Language  
SOAP – Service Oriented Architecture Protocol  
SPARQL – SPARQL Protocol And RDF Query Language  
SQL – Structured Query Language  
URI - Uniform Resource Identifier  
W3C - World Wide Web Consortium  
XLS – Nome da extensão de arquivos da ferramenta Microsoft Excel  
XML - Extensible Markup Language  
XML-RPC – XML-Remote Procedure Call

WWW – World Wide Web

## SUMÁRIO

<b>Lista de Figuras .....</b>	<b>8</b>
<b>Lista de Tabelas.....</b>	<b>10</b>
<b>Lista de Siglas.....</b>	<b>11</b>
<b>1. Introdução .....</b>	<b>15</b>
<b>1.1. Problema.....</b>	<b>15</b>
<b>1.2. Objetivos .....</b>	<b>17</b>
1.2.1. Objetivo Geral .....	17
1.2.2. Objetivos Específicos .....	17
<b>1.3. Organização do texto .....</b>	<b>17</b>
<b>2. Fundamentação Teórica .....</b>	<b>19</b>
<b>2.1. Dados Abertos .....</b>	<b>19</b>
2.1.1 Iniciativas Brasileiras .....	19
2.1.1.1 Transparência Brasil (ONG).....	20
2.1.1.2 Portal Brasileiro de Dados Abertos .....	21
2.1.1.3. Portal da Transparência do Governo Federal .....	22
<b>2.2. World Wide Web .....</b>	<b>22</b>
<b>2.3. Web Semântica .....</b>	<b>23</b>
<b>2.4. Ontologia.....</b>	<b>23</b>
2.4.1 OWL .....	24
2.4.1.1. Namespaces.....	25
2.4.1.2. Cabeçalhos .....	26
2.4.1.3. Classes.....	26
2.4.1.4. Indivíduos .....	26
2.4.1.5. Propriedades .....	27
<b>2.5. Linked Data .....</b>	<b>27</b>
<b>2.6. Data Sets.....</b>	<b>29</b>
2.6.1. DBpedia .....	32
2.6.2. GeoNames.....	32
2.6.3. Ligação entre recursos idênticos em ontologias distintas .....	33
<b>2.7. RDF.....</b>	<b>33</b>
2.7.1. RDF/XML .....	34
2.7.2. RDFa .....	35
2.7.3. N3.....	36
2.7.4. Turtle.....	37
2.7.5. N-Triple .....	37
<b>2.8. Publicação de Linked Data .....</b>	<b>37</b>
2.8.1. Tipos de dados.....	39
2.8.2. Preparação e Armazenamento dos Dados em RDF .....	39
2.8.2.1. D2RQ dump-rdf .....	40
2.8.2.2. NeOn Toolkit.....	41

2.8.3. Publicação dos Dados .....	42
2.8.3.1. Sesame .....	43
2.8.3.2. Pubby .....	43
2.8.3.3. Virtuoso Universal Server .....	44
2.8.4. Consumo dos Dados .....	45
2.8.4.1. Consultas SPARQL .....	45
2.8.4.2. Motores de Busca de Linked Data .....	46
2.8.4.3. Navegadores de Linked Data .....	47
<b>2.9. Aplicações Mashup .....</b>	<b>47</b>
2.9.1. Tipos de Mashup.....	48
2.9.2. Arquitetura .....	50
2.9.3. Desafios no desenvolvimento .....	51
<b>3. Publicação dos Dados Lincados .....</b>	<b>53</b>
<b>3.1. Proposta do trabalho.....</b>	<b>53</b>
3.1.1. Obtenção dos dados .....	53
3.1.2. Definição da ontologia.....	53
3.1.3. Migração dos dados para uma base relacional .....	53
3.1.4. Alinhamento dos datasets .....	54
3.1.5. Geração do RDF.....	54
3.1.6. Publicação e visualização dos dados.....	54
<b>3.2. Execução do trabalho .....</b>	<b>55</b>
3.2.1. Dados a serem publicados.....	55
3.2.1.1. Relacionamento com dados externos .....	56
3.2.1.2. Modelo Relacional .....	57
3.2.2. Ambiente de desenvolvimento.....	58
3.2.3. Geração das ontologias .....	59
3.2.3.1. GeoPoliticaBr .....	59
3.2.3.2. DadosBr.....	60
3.2.4. Geração do arquivo RDF.....	61
3.2.4.1. NeOn Toolkit + ODEMapster.....	61
3.2.4.2. Dump-rdf .....	61
3.2.5. Publicação dos dados.....	64
3.2.5.1. Publicando o arquivo estático .....	64
3.2.5.2. Publicando os dados a partir de um wrapper RDB2RDF .....	65
3.2.5.3. Publicando os dados a partir de um servidor RDF .....	66
3.2.6. Consumo dos dados publicados .....	72
<b>4. Conclusões.....</b>	<b>75</b>
<b>4.1. Avaliação do Resultado.....</b>	<b>75</b>
<b>5. Trabalhos Futuros .....</b>	<b>77</b>
<b>Referência bibliográfica .....</b>	<b>78</b>
<b>Apêndices .....</b>	<b>80</b>

# 1. Introdução

Dados abertos (Open Data, originalmente) são uma vertente da tecnologia da informação que vem crescendo muito na última década, principalmente nos últimos cinco anos. Seu conceito se baseia na ideia de que alguns tipos de informação devem ser disponibilizados gratuitamente, de forma que qualquer pessoa ou instituição possa ter acesso e utilizar da forma que bem entender, sem restrições de direitos autorais ou qualquer mecanismo de controle.

O objetivo desse movimento é criar uma cultura coletiva de construção, agrupamento e publicação de dados. Com a disponibilização desse conhecimento, qualquer entidade, seja ela uma pessoa ou uma organização, pode utilizar essas fontes e criar novos serviços que geram benefícios para quem os cria e quem os utiliza.

Mesmo que a proposta de disponibilização de dados abertos exista e seja uma tendência, ela por si só não é suficiente. Atualmente, esse conteúdo aberto é oferecido nos mais variados formatos, como CSV, XLS, APIs, arquivos de texto e outros. Sendo assim, existe uma dificuldade na hora de relacionar informações de um domínio com a de outras fontes de dados, além de não permitir a interpretação das informações e dados por parte das máquinas.

Tim Berners Lee, diretor do consórcio World Wide Web, propõe que seja utilizado um padrão para publicação e ligação de dados, denominado Linked Data. Esse padrão é baseado em ligações entre os dados, algo que permite a atribuição de semântica aos relacionamentos e, consequentemente, o seu entendimento por máquinas.

Esse projeto apresenta uma abordagem teórica de como utilizar os conceitos apresentados acima, para disponibilizar dados seguindo os princípios de linked data. Na parte prática aplicamos estes conceitos para a publicação de dados geográficos brasileiros, relacionando-os com outros repositórios de dados. Por fim, desenvolvemos uma aplicação que utiliza esses dados como prova de conceito.

## 1.1. Problema

O movimento Dados Abertos (Open Data) tem uma importância muito grande na transparência de organizações como governos. Ele promove a publicação dos dados, permitindo que todos tenham acesso e os utilizem para criar novos serviços. Ainda assim, existe um problema: a não padronização das formas de publicação de dados.

Linked data é um padrão de publicação de dados proposto por Tim Bernes-Lee, o criador da web, que permite que os dados sejam disponibilizados de uma só forma. Tais dados possuem ligações entre si, permitindo a atribuição de semântica para esses relacionamentos. Essa conexão permite que máquinas compreendam os significados dos dados e seus atributos.

No Brasil, já existe um esforço do governo para a publicação de dados abertos, chamado Portal Brasileiro de Dados Abertos. Através dele já é possível encontrar algumas informações sobre pontos específicos, de governos estaduais, ou municipais, que aderiram ao movimento de dados abertos.

De acordo com o projeto, para serem considerados abertos, os dados devem seguir oito princípios estabelecidos. São eles:

- **Completos:** Todos os dados públicos estão disponíveis. Dado público é o dado que não está sujeito a limitações válidas de privacidade, segurança ou controle de acesso.
- **Primários:** Os dados são apresentados tais como os coletados na fonte, com o maior nível de granularidade e sem agregação ou modificação.
- **Atuais:** Os dados são disponibilizados tão rapidamente quanto necessário à preservação do seu valor.
- **Acessíveis:** Os dados são disponibilizados para o maior alcance possível de usuários e para o maior conjunto possível de finalidades.
- **Compreensíveis por máquinas:** Os dados são razoavelmente estruturados de modo à possibilitar o processamento automatizado.
- **Não discriminatórios:** Os dados são disponíveis para todos, sem exigência de requerimento ou cadastro.
- **Não proprietários:** Os dados são disponíveis em formato sobre o qual nenhuma entidade detenha controle exclusivo.
- **Livres de licenças:** Os dados não estão sujeitos a nenhuma restrição de direito autoral, patente, propriedade intelectual ou segredo industrial. Restrições sensatas relacionadas à privacidade, segurança e privilégios de acesso devem ser permitidas.

Entre os estados brasileiros, Santa Catarina é um dos que mais fornece dados abertos para a população. Recentemente, o poder executivo lançou o portal estadual da transparência, que permite o acompanhamento da arrecadação das receitas e a aplicação dos recursos públicos. No entanto, grande parte dos dados disponibilizados de maneira aberta no estado é do domínio financeiro, pois sua publicação faz parte de iniciativas governamentais de



transparência. Enquanto isso, informações estatísticas sobre o estado, embora disponíveis abertamente, encontram-se dispersas e em formatos diferentes, dificultando a sua utilização.

Esse trabalho busca aplicar os conceitos propostos por Tim Berners-Lee, referentes à publicação e consumo de dados na *web*, algo que ainda não foi devidamente explorado no cenário nacional. Serão identificados e recuperados dados geográficos e estatísticos sobre o estado de Santa Catarina e seus municípios, os quais passarão por uma transformação para um formato que permitirá a ligação destes dados com outras bases de dados. Uma vez que este passo seja concluído, os dados serão, então, disponibilizados abertamente na *web*, visando criar uma nova fonte de dados que poderá ser reutilizada por terceiros, além de permitir a criação de novas ligações e expandir o conhecimento geral sobre o assunto em questão.

## **1.2. Objetivos**

### **1.2.1. Objetivo Geral**

Efetuar uma pesquisa teórica sobre publicação de dados ligados, aplicar os conceitos aprendidos em um grupo de dados geográficos abertos do Brasil, relacionando-os com outros datasets, e torná-los públicos. Também faz parte do projeto o desenvolvimento de uma aplicação que utiliza estes dados, como prova de conceito.

### **1.2.2. Objetivos Específicos**

- Apresentar os padrões de publicação de dados ligados
- Localizar e extrair dados abertos pertencentes ao domínio do projeto, a partir de fontes oficiais
- Aplicar os conceitos de publicação nesses dados, disponibilizando-os abertamente na *web*
- Desenvolver uma aplicação que utilize esses dados já nos padrões de dados ligados

## **1.3. Organização do texto**

Este trabalho está organizado em 5 capítulos. Os dois primeiros abrangem a introdução e a fundamentação teórica dos conceitos abordados ou relacionados ao trabalho. Os demais capítulos tratam do desenvolvimento do trabalho e as conclusões obtidas.

No capítulo 2 encontra-se a fundamentação teórica dos conceitos abordados ou relacionados ao trabalho, baseada na revisão da literatura sobre o estado da arte de cada conceito.

No capítulo 3 são detalhados os passos que serão seguidos para se realizar a publicação dos dados

No capítulo 4 encontram-se as conclusões obtidas ao término do trabalho, as quais incluem as limitações encontradas e as considerações finais dos autores.

No capítulo 5 estão descritas oportunidades para trabalhos futuros.

## 2. Fundamentação Teórica

Para entender a proposta deste trabalho, é necessário conhecer alguns conceitos relacionados ao tema em questão. Embora alguns deles, como Web Semântica e Ontologias, possam parecer familiares para certas pessoas, grande parte dos usuários que navegam na web não possui conhecimento sobre o assunto. Portanto, este capítulo visa esclarecer os conceitos que serão abordados durante a execução deste trabalho.

### 2.1. Dados Abertos

Dados Abertos, como explicado no capítulo de introdução deste trabalho, é a ideia de que certos tipos de dados devem estar disponíveis gratuitamente para que todos possam utilizar e republicar, sem restrições de copyright, patentes ou outros mecanismos de controle.

#### 2.1.1 Iniciativas Brasileiras

O acesso à informação está previsto na Constituição Federal e na Declaração Universal dos Direitos Humanos. Sendo assim, o movimento de dados abertos não pode ser considerado um assunto tão recente no Brasil. Algumas iniciativas de difusão de dados abertos na internet, por parte do governo brasileiro, datam do ano de 2000, como o portal ComprasNet<sup>1</sup>, que disponibiliza à sociedade informações referentes às licitações promovidas pelo governo federal (OGP, 2011).

No entanto, até 2011, esse movimento não recebia muito destaque no cenário nacional. A partir desse ano, entretanto, o Brasil passou a ser membro-fundador da Open Government Partnership (OGP), visando expandir suas ações de transparência e combate à corrupção.

A OGP é uma parceria internacional entre governos que realiza iniciativas multilaterais e busca um forte comprometimento das instituições dos governos participantes para promover a transparência, aumentar a participação cívica, combater a corrupção e aproveitar novas tecnologias para tornar o governo mais transparente, eficaz e responsável.

Embora maior parte das iniciativas brasileiras em dados abertos sejam governamentais, como o portal brasileiro de dados abertos e o portal da transparência do governo federal, existem organizações não governamentais que possuem destaque no setor. Um bom exemplo

---

<sup>1</sup> <http://www.comprasnet.gov.br/>

é a Transparência Brasil<sup>2</sup>, que foi referência nas eleições de 2010, ao disponibilizar os históricos da vida pública e as informações das candidaturas de todos os candidatos cadastrados no TSE, evidenciando aqueles que se enquadravam na Lei da Ficha Limpa.

### **2.1.1.1 Transparência Brasil (ONG)**

Transparência Brasil é uma organização independente e autônoma. Foi fundada no ano 2000, por um grupo formado por indivíduos e organizações comprometido com o combate à corrupção.

A tarefa da Transparência Brasil é ajudar organizações civis e governos de todos os níveis no desenvolvimento de metodologias e atitudes voltadas ao combate à corrupção. De forma a cumprir sua missão, a organização atua em diferentes frentes, incluindo a internet.

No âmbito da internet, a Transparência Brasil disponibiliza instrumentos e dados que permitem monitorar o fenômeno da corrupção. São alguns deles:

- **Excelências:** Disponibiliza informações sobre todos os parlamentares em exercício nas Casas legislativas das esferas federal e estadual, e mais os membros das Câmaras Municipais das capitais brasileiras. Os dados são extraídos de fontes públicas (as próprias Casas legislativas, o Tribunal Superior Eleitoral, tribunais estaduais e superiores, tribunais de contas e outras) e de outros projetos mantidos pela Transparência Brasil. Esses dados incluem o histórico de toda vida pública dos parlamentares federais e estaduais, os processos à que respondem na justiça, declaração de bens, noticiários sobre corrupção os envolvendo, multas recebidas pelos tribunais de contas, padrões de financiamento eleitoral, frequência de trabalho entre outras. Visando evitar mal-entendidos ou equívocos, é disponibilizado um espaço para que qualquer político retratado no site possa apresentar seus argumentos e/ou justificativas sobre as informações divulgadas. No ano de 2006, o projeto Excelências recebeu o prêmio Esso de Jornalismo na categoria “Melhor Contribuição à Imprensa“. Devido à boa recepção, reconhecimento recebido e serviço prestado contra a corrupção, o projeto passou a ser financiado pelo Fundo das Nações Unidas para a Democracia.
- **Deu no Jornal:** Um banco de dados composto por noticiários sobre corrupção e seu combate, atualizado diariamente. As notícias são extraídas de 63 dos principais jornais e revistas nacionais. Permite que o usuário pesquise notícias

---

<sup>2</sup> <http://www.transparencia.org.br/index.html>

por período de publicação, publicação, partido do envolvido, estado da ocorrência e palavras-chave.

- **Às Claras:** Banco de dados com informações e análises sobre o financiamento eleitoral de candidatos que participaram de eleições a partir do ano de 2002. Possui um detalhamento das fontes que contribuíram para o financiamento do candidato, custo por voto e outros indicadores.

Além desses projetos, a Transparência Brasil também tem participação em outras iniciativas, como um projeto em conjunto com Tribunal de Contas do estado de Santa Catarina, para acompanhamento e verificação das licitações realizadas no estado, dentre outros.

Porém, embora trabalhe com dados abertos, a Transparência Brasil ainda não adota o padrão de Linked Data.

### **2.1.1.2 Portal Brasileiro de Dados Abertos**

Em 15 de setembro de 2011, o governo brasileiro, em parceria com a OGP, formalizou o seu plano de ação de governo aberto. Um dos compromissos assumidos nesse plano era a criação do Portal Brasileiro de Dados Abertos<sup>3</sup> (CGU, 2011).

Além disso, em 18 de novembro de 2011, foi sancionada a Lei de Acesso à Informação Pública (Lei 12.527/2011), que regula o acesso a dados e informações detidas pelo governo, seja federal, estadual ou municipal. Em outras palavras, a lei assegura a qualquer pessoa o direito de requisitar qualquer informação de interesse público ao governo, que, por sua vez, tem que ser transparente. Essa lei é considerada como um marco para a democratização da informação pública.

O Portal Brasileiro de Dados Abertos, portanto, é a ferramenta que o governo criou e disponibilizou para que as pessoas possam pesquisar, encontrar e utilizar as informações públicas. Além disso, permite a interlocução entre membros da sociedade com integrantes do governo, visando discutir a melhor utilização dos dados.

Como o objetivo do portal é disponibilizar tudo e qualquer tipo de dado, ele foi projetado como um grande catálogo, facilitando a busca e utilização das informações disponibilizadas pelos órgãos governamentais.

Embora a quantidade de dados abertos disponíveis no portal ainda não seja grande, como a própria página do portal afirma, a Lei de Acesso à Informação entrou em vigência a partir de 180 dias de sua publicação, o que aconteceu no dia 16 de maio de 2012. No entanto, o

---

<sup>3</sup> <http://dados.gov.br/>

plano estratégico do portal prevê que os dados publicados por todos os órgãos do governo, das três esferas, sejam disponibilizados dentro dos próximos três anos.

Assim como acontece com os dados disponibilizados pela Transparência Brasil, os dados encontrados no Portal Brasileiro de Dados Abertos não atendem aos requisitos de Linked Data. Sendo assim, quem quiser relacionar os dados ainda precisará aplicar técnicas de ETL nos dados desejados.

### **2.1.1.3. Portal da Transparência do Governo Federal**

O Portal da Transparência do Governo Federal<sup>4</sup> foi lançado em 2004, através da Controladoria-Geral da União. As informações disponibilizadas nesse portal permitem que o cidadão consiga realizar um acompanhamento e fiscalizar a utilização do dinheiro público. Assegurando, dessa maneira, a correta aplicação dos recursos públicos e aumentando a transparência da gestão pública.

## **2.2. World Wide Web**

Alguns autores, como Karin Breitman (2005), argumentam que a World Wide Web, como ela foi proposta e existe atualmente, pode ser chamada de Web Sintática. Para que se possa compreender essa nomenclatura, é necessário entender a proposta da Web e como é sua estrutura. No ano de 1989, Tim Berners-Lee propôs, pela primeira vez, a Web. Segundo a proposta, a Web tinha como objetivo utilizar a conexão à internet para disponibilizar documentos com hiperlinks, os quais seriam utilizados para acessar e ligar informações, dos mais variados tipos, formando uma teia de hiperlinks, sendo que esta poderia ser navegada à vontade pelo usuário. Desta maneira, utilizando um navegador web, o usuário poderia visualizar inúmeras páginas, as quais poderiam conter diversos tipos de informações, sejam textos, vídeos, fotos e outros tipos de mídia, e navegar livremente, entre elas, utilizando os hiperlinks.

Inicialmente, apenas programadores eram capazes de criar páginas e disponibilizá-las na Web (BREITMAN, 2005). Porém, devido à facilidade de comunicação e compartilhamento de informações que essas páginas forneciam, a Web foi ganhando popularidade (BREITMAN, 2005). Desta maneira, foram surgindo ferramentas que permitiam aos usuários, sem conhecimento de programação, a possibilidade de criar suas próprias páginas (BREITMAN, 2005).

---

<sup>4</sup> <http://www.portaltransparencia.gov.br/sobre/>

Porém, desde o início da Web, as informações compartilhadas sempre foram direcionadas à outros humanos, possuindo baixa ou nenhuma possibilidade de serem interpretadas por máquinas, uma vez que apenas humanos são capazes de identificar o contexto da informação que está sendo visualizada. Justificando, assim, a nomenclatura Web Sintática, uma vez que não há como uma máquina extrair a semântica dos documentos, mesmo que estes estejam estruturados.

Para permitir que dados e informações sejam interpretadas por máquinas, visando inúmeras melhorias para aplicações já existentes, como motores de busca, além da invenção de novas aplicações que facilitariam a vida de todos conectados na internet, foi proposta a Web Semântica em 2001.

## **2.3. Web Semântica**

O conceito de web semântica surgiu ao público em 2001 com a publicação de um artigo na revista Scientific American com autoria de Tim Berners-Lee, James Hendler e Ora Lassila (BERNERS-LEE et al, 2001). A principal idéia da proposta é estender a web atual, mudando foco de documentos para dados, e superar as limitações que o modelo de hoje possui.

A web semântica possui dois objetivos principais. O primeiro é definição de padrões de formatação para a integração e combinação de dados provenientes de diversas fontes. O segundo é a atribuição de significado, mostrando como os dados se relacionam com objetos do mundo real. Esses dois pontos em conjunto permitem que não só pessoas, mas também máquinas possam navegar através de bases de dados, que possuem um relacionamento entre si, estabelecido pelo significado dos seus conteúdos.

As tecnologias relacionadas com a Web Semântica atuam no plano de fundo das aplicações ao invés de causarem um impacto visual durante a navegação. Suas aplicações resultam em uma melhor experiência para o usuário que a utiliza. Já existem alguns projetos que se beneficiam com essas práticas, como o portal de finanças do Yahoo.

Web semântica é uma peça chave para a próxima geração de sistemas de informações, onde as informações recebem um significado bem definido, permitindo que pessoas e programas trabalhem em cooperação.

## **2.4. Ontologia**

Ontologia é um assunto que vem sendo objeto de pesquisa tanto por filósofos quanto por cientistas. Nos últimos anos a comunidade de ciências da computação entrou no estudo

desse tópico, tornando-o assim, um tema multidisciplinar, com estudos em áreas como engenharia do conhecimento, integração de inteligência, gerência de conhecimento e sistemas de informação cooperativos, etc.

O estudo de ontologia pela computação tem sua origem no domínio da Inteligência Artificial para a representação da existência de "coisas", mas outras áreas da computação também começaram a usar ontologias para estabelecer uma terminologia conjunta entre máquinas e humanos.

A definição mais usada para esse termo é dada por Gruber (1993), caracterizando ontologia como "uma especificação formal de uma conceitualização", sendo que esta é dita como "uma visão abstrata e simplificada do mundo que desejamos representar para um propósito qualquer".

Outra definição, mais clara, de ontologia para a área da computação é dada abaixo:

*“Uma ontologia define os termos básicos e relações relacionadas a uma área de conhecimento, assim como as regras para combinar estes termos e relações a fim de definir extensões do vocabulário”* (NECHES et al, 1991, tradução).

O objetivo de uma ontologia é prover um vocabulário compartilhado para o entendimento de um domínio. Ela possui definições para os conceitos e seus relacionamentos, que podem ser compreendidos por um computador (BREITMAN, 2005). Desta forma uma ontologia pode ser usada para comunicação e troca de conhecimento entre máquinas e humanos, interoperabilidade entre sistemas, etc.

Existem diferentes linguagens disponíveis que podem ser utilizadas para a representação de ontologias, incluindo a Simple HTML Ontology Extension (SHOE), Ontology Interface Layer (OIL), DARPA Agent Markup Language (DAML), Resource Description Framework (RDF), RDF-Schema e Web Ontology Language (OWL) (BREITMAN, 2005). Uma vez que o objetivo deste trabalho não é realizar uma análise comparativa dessas linguagens, apenas OWL e RDF são abordadas devido ao fato de serem utilizadas no desenvolvimento deste trabalho. OWL é abordada nesta seção, enquanto RDF possui uma seção própria.

### **2.4.1 OWL**

Web Ontology Language (OWL) é a linguagem projetada pela W3C que visa atender as necessidades de aplicações criadas para Web Semântica. Surgiu como uma revisão das linguagens OIL e DAML.

OWL é utilizada para criar ontologias e permite o fornecimento de informações sobre os conceitos e propriedades que as formam. Outra característica é a sua capacidade de explicitar



fatos referentes a um domínio determinado, fornecendo informações sobre os indivíduos do domínio. É possível também utilizá-la para racionalizar sobre as ontologias e seus fatos, ou seja, determinar as consequências do que foi construído ou explicitado.

Existem três linguagens OWL, com níveis crescentes de expressividade. São elas:

- OWL Lite
- OWL DL
- OWL Full

O objetivo da OWL Lite é dar suporte à migração de taxonomias para ontologias. Para isso, possui um conjunto reduzido de operações, mas que oferecem o suporte a criação de hierarquias simplificadas de classificação, incluindo suas restrições. Por sua vez, a OWL DL tem suporte a lógica descritiva, enquanto que a OWL Full suporta o máximo de expressividade, mantendo uma completude computacional.

A linguagem OWL possui alguns elementos básicos, como namespaces, classes, indivíduos e propriedades, que serão abordados nessa seção. Demais elementos podem ser consultados no guia de OWL da W3C.

#### 2.4.1.1. Namespaces

Os namespaces na OWL servem para declarar quais são os vocabulários que serão utilizados. Normalmente, os namespaces compõem o primeiro bloco de informações em um arquivo OWL.

```
<rdf:RDF
  xmlns      = "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
  xmlns:vin  = "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
  xml:base   = "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
  xmlns:food = "http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#"
  xmlns:owl  = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf  = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd  = "http://www.w3.org/2001/XMLSchema#">
```

Figura 1 – Exemplo de declaração de Namespaces em OWL

A declaração de namespaces, que é feita utilizando a tag `rdf:RDF`, permite com que os identificadores presentes na ontologia possam ser interpretados sem ambiguidade. Na figura X, a primeira linha da declaração indica o namespace padrão, que será utilizado sempre que um nome não tiver um prefixo atribuído. As demais linhas deixam explícitos quais são os demais prefixos suportados e seus namespaces.

### 2.4.1.2. Cabeçalhos

O cabeçalho de uma ontologia é declarado utilizando a tag owl:Ontology, e vem logo após a declaração dos namespaces.

O cabeçalho contém alguns meta-dados sobre a ontologia. Esses meta-dados incluem o nome da ontologia, declarado através do atributo rdf:about, que pode ser declarado vazio, um comentário sobre a ontologia, sua versão anterior, quais recursos ela importa e uma etiqueta, que permite a categorização da ontologia em uma língua natural.

### 2.4.1.3. Classes

Classes são utilizadas para descrever conceitos de um domínio, pois representam um conjunto ou coleção de indivíduos que compartilham características em comum, as quais os distinguem dos demais.

Em OWL, classes são utilizadas para conceitos fundamentais de um determinado domínio, os quais servirão como raízes para as mais diversas taxonomias. De modo que sempre exista uma raiz em comum para todas as taxonomias, uma classe descrita em OWL será, implicitamente, sempre uma subclasse de owl:Thing.

```
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
```

Figura 2 – Exemplo de declaração de classes em OWL

É muito simples declarar uma classe em OWL, basta-se utilizar a tag owl:Class seguida do atributo rdf:ID, que identificará o nome da classe, como observado na figura X. Porém, uma simples declaração de classe diz absolutamente nada sobre ela, além de seu nome. Ou seja, ao se declarar uma classe com o identificador “região” e outra com o identificador “comida”, não será possível distinguir seus significados, pois suas características são as mesmas. Desta forma, é necessário declarar de forma explícita todas as características que definem a classe em questão para poder distingui-la das demais.

No exemplo “região”, pode-se supor que a classe tem o potencial para abranger um continente, país, estado, município e até mesmo um bairro. Além disso, uma região pode possuir área, população, data de fundação, latitude, longitude e outras propriedades. Em OWL, para dizermos que um município é um tipo de região, utilizaríamos a tag rdf:subClassOf na classe município, indicando que é uma subclasse de região. A seção 2.4.1.5 fala sobre a declaração de propriedades, como nesse caso, a área de um município.

### 2.4.1.4. Indivíduos

Objetos do mundo real são representados como indivíduos em uma ontologia. Indivíduos pertencem à uma classe e se relacionam com outras classes e indivíduos através de propriedades.

A declaração de um indivíduo ocorre através da utilização de uma tag com o nome de sua classe seguida do atributo `rdf:ID`, que identifica o indivíduo. Alternativamente, é possível utilizar uma declaração de `owl:Thing`, uma vez que todas classes são subclasses de `Thing`, identificando a subclasse com a tag `rdf:type` seguida do `rdf:resource` com o endereço da classe.

#### **2.4.1.5. Propriedades**

São utilizadas como maneira de expressar fatos. Existem dois tipos de propriedades, as que se referem ao relacionamento entre classes, chamadas de `ObjectProperty`, e as que tratam do relacionamento entre indivíduos de uma classe e literais, expressos em RDF e datatypes do XML Schema.

Assumindo, como um simples exemplo, que um humano bebe água, para expressar essa relação em OWL deve-se criar um `ObjectProperty` chamado “bebe”, cujo o domínio será a classe “Humano” e o intervalo será a classe “Água”. Continuando com o exemplo da classe “Humano”, caso queira-se indicar que este possui um ano de nascimento em OWL, será necessária a declaração de uma `DatatypeProperty`, que possua o nome “anoNascimento”, cujo domínio será a própria classe “Humano” e o intervalo esperado será um valor inteiro e positivo, como por exemplo “`xsd:positiveInteger`”.

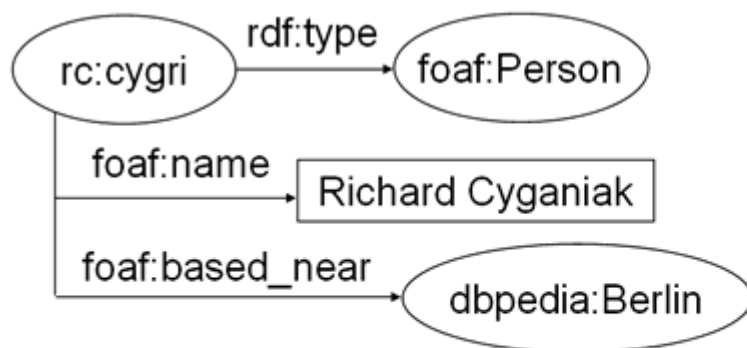
## **2.5. Linked Data**

Ao analisar a Web atual, é fácil perceber que existe uma relação entre os documentos e páginas visitadas. Essa relação é criada através da inserção de *Hyperlinks*, que direcionam os usuários às páginas ou documentos desejados. Porém, embora haja a relação entre os documentos, não existe uma relação entre as informações disponíveis. Portanto, se um dos objetivos da web semântica é a integração entre diversas fontes de dados, visando enriquecer as informações disponíveis, pode se dizer que Linked Data é um dos pilares que a sustenta.

O termo Linked Data representa um conjunto de boas práticas para conectar e publicar dados estruturados na Web (BERNERS-LEE et al, 2009). A adoção dessas boas práticas tem conduzido à criação de um data space global, no qual informações de diversas fontes e domínios diferentes são relacionadas.

A expansão de Linked Data abre portas para inúmeras novas aplicações e serviços. Já existem navegadores e motores de buscas específicos para esse padrão de dados. O tipo de

aplicação mais beneficiado pela expansão da adoção de linked data são as aplicações de mashup. Diferentemente de aplicações mashups já existentes, que trabalham em cima de um conjunto de dados limitado, as aplicações que fizerem uso de dados ligados não terão um limite visível, uma vez que estarão se conectando e consumindo dados de um data space global, permitindo sempre mostrar novas informações à medida que elas são publicadas na web. A figura abaixo demonstra um exemplo de utilização de linked data.



**GET /resource/Berlin HTTP/1.0**  
**Accept: application/rdf+xml**

**Figura 3 - Perfil de Richard Cyganiak e uma pesquisa por informações relacionadas à cidade de Berlim.**

Na figura 1, podemos ver que existe no dataset “RC” um registro chamado “cygri”. Este registro identifica Richard Cyganiak, uma pessoa, como definido no vocabulário FOAF<sup>5</sup>, que mora em Berlim. Porém, percebe-se que cygri não está no mesmo dataset que pessoa, assim como ambos também não se encontram no mesmo dataset de Berlim, a DBpedia. Ao analisar o perfil de Richard Ciganyak e descobrir que ele mora em Berlim, através da ligação entre os datasets, o usuário resolveu pesquisar por informações relacionadas à Berlim, como demonstrado na pelo comando GET executado na figura 1.

<sup>5</sup> FOAF Vocabulary Specification 0.98 <http://xmlns.com/foaf/spec/>

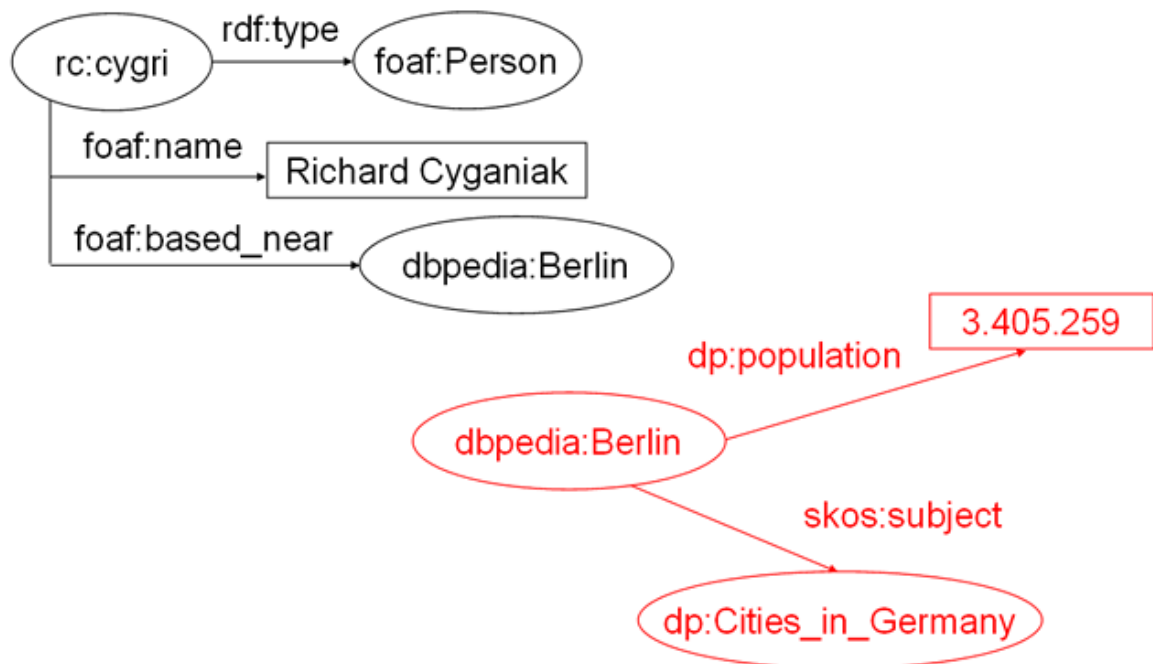


Figura 4 - Resultado obtido através da pesquisa realizada na Figura 3

Ao receber o resultado do seu comando GET, o usuário agora tem informações sobre Berlim que foram encontradas no dataset da DBpedia. Neste resultado, uma nova ligação aparece, e o usuário agora descobre que Berlim é uma das cidades da Alemanha. Dessa maneira, ele pode agora pesquisar por cidades alemãs e continuar navegando de registro em registro, como é o objetivo de linked data.

## 2.6. Data Sets

A ligação entre informações, como descrita na seção anterior, é o que forma o data space global que sustenta a web semântica. Em maio de 2007, apenas 12 data sets possuíam informações publicadas no formato de linked data, como visto na figura 5.

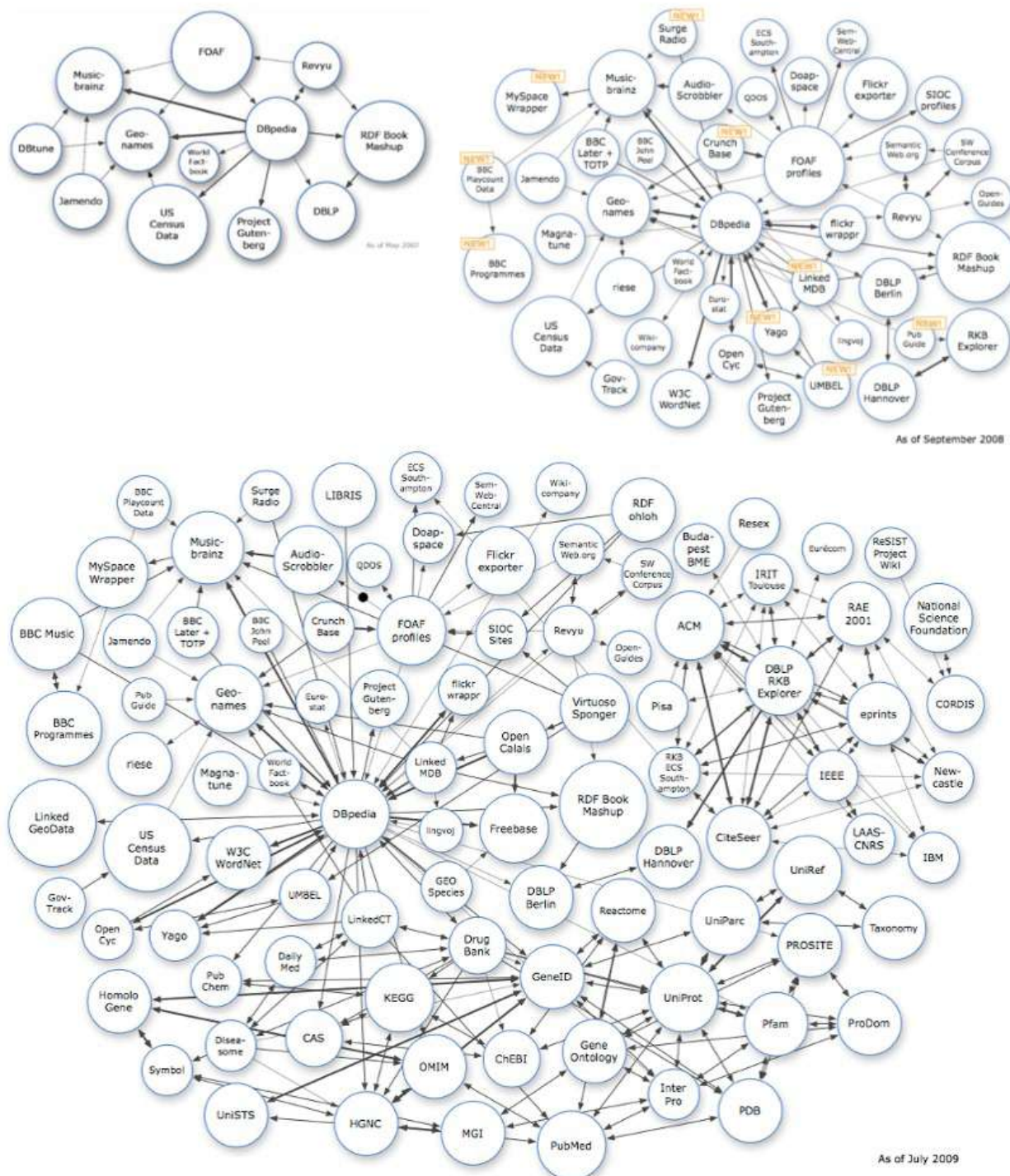


Figura 5 – Evolução dos data sets de linked data entre 2007 e 2009

Esse número vem crescendo exponencialmente ao longo dos últimos anos. O que eram, inicialmente, 12 data sets em 2007 tornaram-se 295 data sets, na última medição realizada, no

ano de 2011. A figura abaixo demonstra visivelmente esse crescimento ao longo dos últimos anos.

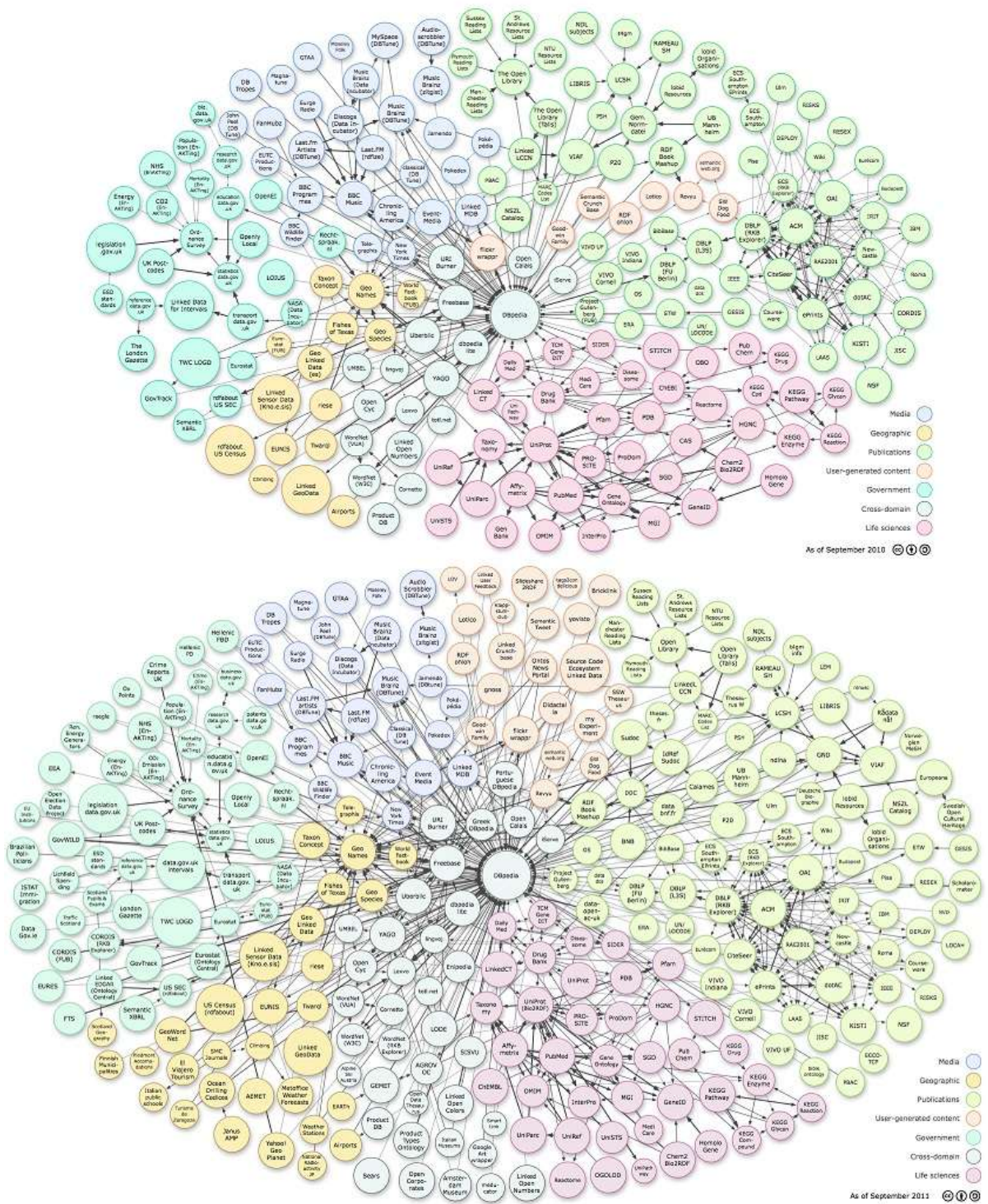


Figura 6 - Evolução dos data sets de linked data entre 2010 e 2011.

Dentre os data sets disponíveis, alguns podem ser considerados como pilares do data space global, por serem os que possuem o maior número de referências em outros datasets, ou seja, são os que possuem o maior número de ligações com informações de outras fontes. No escopo deste trabalho, os data sets que se destacam são a DBpedia e o GeoNames, por possuírem uma gama vasta de informações relacionadas aos municípios catarinenses, as quais são utilizados neste trabalho.

### **2.6.1. DBpedia**

DBpedia<sup>6</sup> trata-se de um esforço comunitário de extração de informações estruturadas da Wikipedia, sendo o dataset com o maior número de ligações entre todos os datasets, como visto na Figura 4. As informações disponíveis na base de conhecimento da DBpedia descrevem mais de 3,64 milhões de itens, pertencentes aos mais diversos domínios, em até noventa e sete línguas diferentes, sendo composta por um bilhão de triplas RDF, as quais podem ser acessadas de graça, através de download ou consultas em um endpoint Sparql.

Por possuir uma base tão extensa de conhecimento, existem muitas informações disponíveis sobre os municípios catarinenses na DBpedia. Dentre as informações disponíveis, encontram-se propriedades como o nome do município, sua área, população, apelidos e nome do prefeito. Além disso, estão, também, expressas relações com outros recursos, como por exemplo a ligação com pessoas famosas que nasceram no município, times de futebol que atuam no município e pessoas de destaque que nele residem.

### **2.6.2. GeoNames**

Embora o GeoNames<sup>7</sup> não seja o maior dos datasets de informações geográficas, em número de triplas, ele continua sendo o dataset do domínio geográfico mais referenciado por outros datasets. Assim como a DBpedia, as informações disponíveis no GeoNames são gratuitas, podendo ser acessadas através de webservices ou download. Esse dataset contém mais de dez milhões de nomes geográficos, sendo 2,8 milhões deles lugares povoados e 5,5 milhões de nomes alternativos para os recursos.

Por se tratar de um data set sobre dados geográficos, possuir o maior número de ligações entre informações do domínio e receber contribuições diretamente do IBGE, o GeoNames é uma escolha óbvia para análise neste trabalho.

---

<sup>6</sup> <http://dbpedia.org/About>

<sup>7</sup> <http://www.geonames.org/about.html>



### 2.6.3. Ligação entre recursos idênticos em ontologias distintas

Para se realizar a ligação entre recursos que tratam de um mesmo assunto, mas que estão localizados em ontologias distintas, é declarada uma DatatypeProperty do tipo owl:sameAs, cujo o domínio será a classe do objeto na ontologia local e o intervalo esperado será uma referência para o recurso idêntico na ontologia alvo.

Existem diferentes maneiras de se utilizar a propriedade owl:sameAs, sendo a mais simples utilizar a URI, do objeto que será referenciado, como ligação com o objeto local. Imagine que seja declarada uma ontologia sobre restaurantes, e um dos indivíduos dessa ontologia é o “Restaurante do João”. Caso esse restaurante possua um recurso equivalente na ontologia da DBpedia, pode-se obter a URI da DBpedia que referencie o restaurante e utilizá-la como valor na propriedade owl:sameAs do recurso “Restaurante do João” na ontologia local.

Para se obter a URI desejada, pode-se realizar uma busca manual no dataset, procurando os identificadores dos recursos desejados ou utilizar algum algoritmo que automatize esse processo.

## 2.7. RDF

Para permitir que várias aplicações sejam capazes de processar o conteúdo da Web, é importante que exista uma padronização no formato desse conteúdo.

RDF é um modelo padrão para intercâmbio de dados na Web. Possui características que facilitam a composição de dados de schemas diferentes mesmo que esses sejam diferentes. Além disso suporta a evolução desses esquemas sem a necessidade de alterações para quem consome os dados.

O modelo estende a estrutura de ligações da Web, usando URIs para nomear os relacionamentos entre as coisas, além dos recursos que estão sendo ligados. Utilizar esse modelo permite que dados estruturados e semi-estruturados sejam agrupados, expostos e compartilhados através de aplicações web.

RDF permite que qualquer pessoa ou máquina crie sentenças sobre qualquer recursos. De modo geral não é garantido que todas as informações dos recursos estejam disponíveis. Além disso, RDF não previne que sejam feitas relações sem-sentido ou inconsistentes. Essa garantia de consistência fica sob responsabilidade do desenvolvedor da aplicação que utiliza RDF.

A estrutura de qualquer expressão em RDF é formada por uma tripla, cada uma com um sujeito, predicado e um objeto, como podemos ver na imagem abaixo:

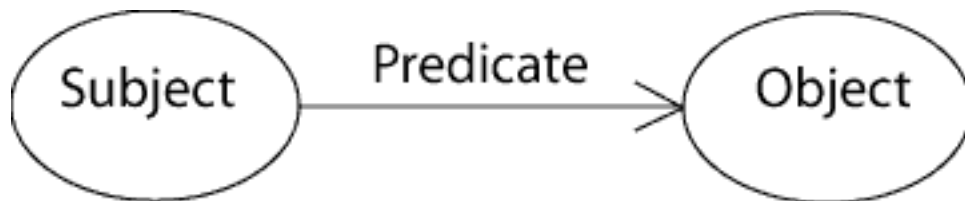


Figura 7 – Formato de uma tripla RDF

Cada tripla representa uma sentença de um relacionamento entre as coisas representadas pelos nós que estão ligados. Cada tripla contém três partes:

- a. Um sujeito
- b. Um objeto
- c. Um predicado, também chamado de propriedade, que representa o relacionamento

O sujeito da tripla é uma URI que referencia o recurso. O objeto pode ser tanto uma outra URI de outro recurso quanto um valor literal, como um número ou um texto. O predicado é sempre uma URI que representa como sujeito e predicado se relacionam.

Para publicar o grafo RDF na Web, é preciso que ele seja serializado, utilizando uma sintaxe RDF. Isso nada mais é que escrever esse grafo em um arquivo, a partir de uma sintaxe específica. Existem dois formatos de serialização padronizados pela W3C: RDF/XML e RDFa. Além desses, outros modelos não padronizados também são utilizados para atender necessidades específicas.

### **2.7.1. RDF/XML**

A sintaxe RDF/XML é padronizada pela W3C e é muito usada para a publicação de dados linkados na Web. Ela tem como característica o difícil entendimento para humanos na leitura e escrita, de forma que esse fator deve ser levado em consideração na escolha da forma de serialização quando existe a necessidade de interação humana.

O MIME (Multipurpose Internet Mail Extensions), que deve ser utilizado em transações do protocolo HTTP, é `application/rdf+xml`.

Exemplo:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:foaf="http://xmlns.com/foaf/0.1/">
5
6   <rdf:Description rdf:about="http://biglynx.co.uk/people/dave-smith">
7     <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
8     <foaf:name>Dave Smith</foaf:name>
9   </rdf:Description>
10
11 </rdf:RDF>
```

Figura 8 – Exemplo de documento RDF+XML

## 2.7.2. RDFa

RDFa é um formato de serialização que adiciona triplas RDF em documentos HTML. O dado RDF não é adicionado nos comentários dentro do documento, como era comum em formas primitivas de unir RDF e HTML. Ao invés disso, são entrelaçados com o HTML DOM (Document Object Model). Desta forma, o conteúdo existente na página pode ser marcado com RDFa modificando o código HTML e expondo dados estruturados na Web.

RDFa é popular em contextos onde os os publicadores são capazes de modificar templates HTML mas tem pouco controle sobre a infraestrutura de publicação. Por exemplo, muitos sistemas gerenciadores de conteúdo permitem que se configure as templates usadas para expor as informações, mas não são flexíveis o suficiente para suportar redirecionamentos 303<sup>8</sup> e negociação de conteúdo HTTP.

Quando se usa RDFa para publicar dados linkados na Web é importante manter o cuidado para não permitir ambiguidades entre a descrição dos dados do mundo real descritos e o documento HTML+RDFa que o descreve.

Um exemplo de código RDFa é mostrado na Figura 9.

---

<sup>8</sup> <http://tools.ietf.org/html/rfc2616#section-10.3.4>

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">
3
4 <head>
5   <meta http-equiv="Content-Type" content="application/xhtml+xml;
  charset=UTF-8"/>
6   <title>Profile Page for Dave Smith</title>
7 </head>
8
9 <body>
10  <div about="http://biglynx.co.uk/people#dave-smith" typeof="foaf:Person">
11    <span property="foaf:name">Dave Smith</span>
12  </div>
13 </body>
14
15 </html>

```

Figura 9 – Exemplo de documento HTML+RDF

### 2.7.3. N3

Notation3, ou N3, é um formato que fornece abreviação não-XML da serialização de modelos RDF, projetada com a legibilidade humana em mente. Outras características do formato vão além da serialização de modelos RDF, como o suporte a regras RDF. Esse formato está em desenvolvimento por Tim Berners-Lee e outros contribuidores da comunidade da web semântica.

Descrição de um recurso em RDF/XML

```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">
    <dc:title>Tony Benn</dc:title>
    <dc:publisher>Wikipedia</dc:publisher>
  </rdf:Description>
</rdf:RDF>

```

Descrição do mesmo recurso em N3

```

@prefix dc: <http://purl.org/dc/elements/1.1/>.

<http://en.wikipedia.org/wiki/Tony_Benn>
  dc:title "Tony Benn";
  dc:publisher "Wikipedia".

```

Figura 10 – Comparação entre RDF/XML e N3

Pode-se observar na figura X que a utilização do formato N3 realmente abrevia e facilita a legibilidade do RDF.

#### **2.7.4. Turtle**

Turtle é mais um formato de serialização de grafos RDF, sendo um subconjunto de N3. Permite que grafos RDF sejam escritos de uma maneira compacta e natural, com abreviações para padrões e tipos de dados comumente utilizados. Possui compatibilidade com os formatos N3, N-Triple e a sintaxe da linguagem SPARQL.

Esse formato foi submetido ao W3C para ser aceito como padrão em março de 2011. Em agosto do mesmo ano foi aceito como primeiro rascunho de trabalho pelo RDF Working Group, do W3C.

#### **2.7.5. N-Triple**

É um subconjunto de Turtle e permite a serialização, baseada em linhas, de informações de modelos RDF em texto puro. Foi projetado para ser mais simples que N3 e Turtle, mas, por não possuir atalhos existentes em outros formatos de serialização de RDF, pode tornar onerosa a tarefa de inserir manualmente grandes montantes de informação, além de dificultar a leitura de informações presentes em grande volume.

### **2.8. Publicação de Linked Data**

A publicação de Linked Data na web pode ocorrer de maneiras diferentes, que variam de acordo com o tipo dos dados, passos realizados e locais de armazenamento e publicação, itens que são abordados nesta seção. Levando isso em consideração, pessoas interessadas em publicar dados neste formato devem sempre obedecer quatro princípios fundamentais, estabelecidos por Berners-Lee (2006). São eles:

1. Utilizar URIs como nome para coisas;
2. Utilizar HTTP URIs para que pessoas possam encontrar esses nomes;
3. Quando alguém procura uma URI, prover informações úteis, utilizando os padrões RDF e SPARQL;
4. Incluir ligações com outras URIs, para que essas pessoas possam descobrir mais coisas.

Christian Bizer e Tom Heath, autores do livro *Linked Data: Evolving the Web into a Global Data Space*<sup>9</sup>, uma das maiores referências sobre o assunto e disponível gratuitamente na internet, elaboraram um diagrama que explica os métodos existentes para publicação de linked data, como visto na figura abaixo.

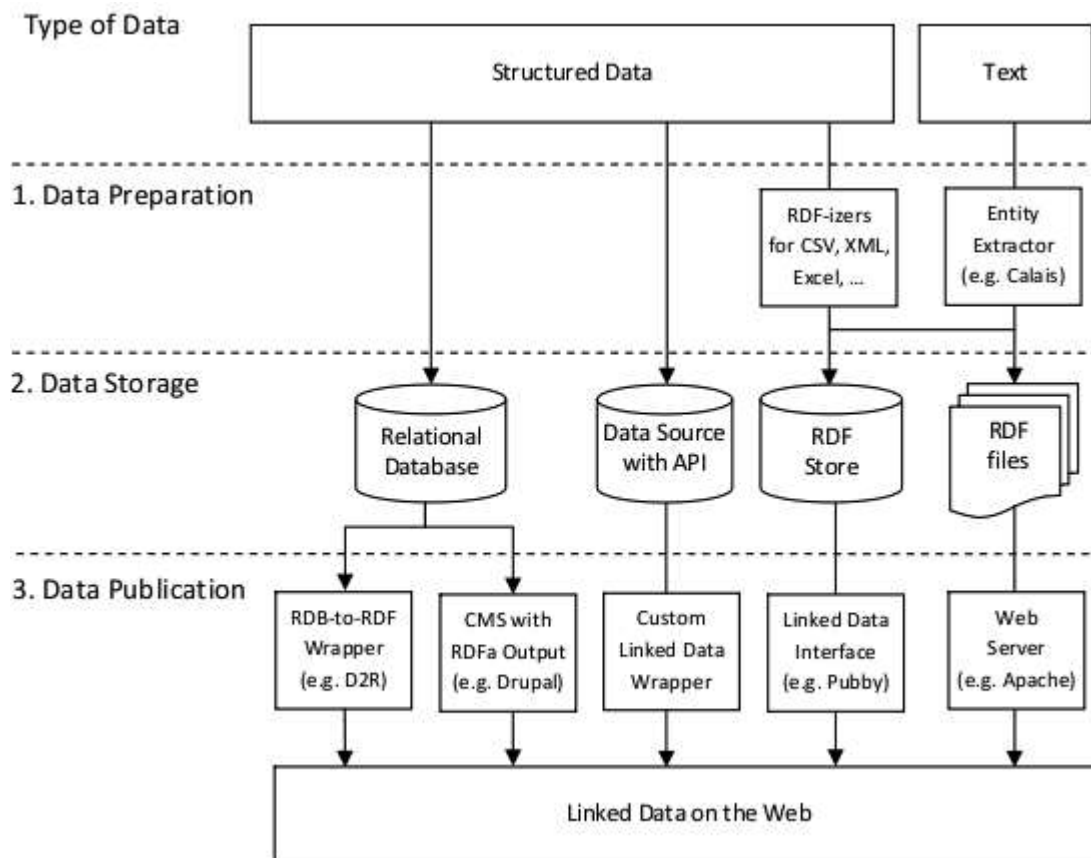


Figura 11 – Diagrama de publicação de linked data Fonte: *Linked Data Book*

Como demonstrado no diagrama, diferentes tipos de dados precisam passar por etapas distintas no processo de publicação. Como o objetivo deste trabalho é criar um arquivo RDF, aplicando os princípios de linked data, com dados geográficos e estatísticos dos municípios catarinenses, os quais podem ser encontrados abertamente e de forma estruturada diretamente nas suas fontes, e disponibilizá-lo em um servidor RDF, a ênfase desta seção será dada no fluxo que faz correspondência no diagrama, passando por RDF-izers, RDF Store e Linked Data Interface.

<sup>9</sup> <http://linkeddatabook.com/editions/1.0/>

Embora esta abordagem seja o foco do trabalho, nem sempre será a melhor escolha na hora de publicar Linked Data. Na hora de escolher, devem-se levar em conta alguns aspectos, como o volume e a frequência de atualização dos dados que serão publicados.

Para um volume de dados pequeno, a criação de um arquivo RDF estático provavelmente será a melhor opção, pois oferece uma complexidade tecnológica menor na hora de armazenar, consultar e atualizar os dados, comparada à utilização de bancos relacionais, APIs para acesso ou servidores de RDF. No entanto, ao passo que o volume de dados aumenta, começam a surgir problemas com a gestão do arquivo RDF estático. Sua velocidade de geração, atualização e acesso cairá proporcionalmente ao volume total dos dados.

Além disso, a frequência de atualização das informações também influencia de maneira relevante na escolha da abordagem. Ao se utilizar arquivos estáticos, caso os dados possuam uma frequência alta de atualização, o sistema, ou responsável, terá que, sempre que houver uma atualização, gerar um novo arquivo RDF. Isso pode ser custoso e até complexo. Portanto, nesse caso, é recomendada a utilização de um servidor RDF, que facilita esse tipo de gestão. Por outro lado, quando a frequência de atualização dos dados é baixa, a menor complexidade tecnológica oferecida pelos arquivos estáticos deve ser levada em consideração.

### **2.8.1. Tipos de dados**

Os dados que servirão de entrada no processo de publicação podem ser encontrados de maneira estruturada ou em texto livre. Logicamente, quando encontrados em forma estruturada, existe uma facilidade maior na hora de trabalhar e preparar os dados para o passo seguinte. Diferentemente, dados encontrados em formato de texto livre precisarão ser extraídos e transformados em algo estruturado, para então estarem disponíveis para o próximo passo do processo de publicação.

### **2.8.2. Preparação e Armazenamento dos Dados em RDF**

A preparação dos dados é um passo que ocorre para ambos os tipos de dados abordados anteriormente, quando é visado o armazenamento das informações diretamente em arquivos ou servidores RDF.

Neste passo são utilizadas ferramentas chamadas RDFizers, cuja função é transformar os dados de entrada, encontrados em um formato específico, como por exemplo csv, xls ou até mesmo em uma base de dados relacional, ou em texto livre, para o formato RDF.

Devido ao foco na criação de um arquivo RDF e sua disponibilização em um servidor específico para RDFs, foram levantadas algumas ferramentas que auxiliam nesse processo.

### 2.8.2.1. D2RQ dump-rdf

Trata-se de uma ferramenta pertencente à plataforma D2RQ<sup>10</sup>. Tem a capacidade de despejar todo o conteúdo de um banco relacional dentro de um arquivo RDF.

Para realizar a ligação entre o dado no banco relacional e o dado no formato RDF, pode-se criar um arquivo de mapeamento D2RQ. Caso o arquivo de mapeamento não seja criado, será utilizada a ferramenta generate-mapping, da mesma plataforma, para realizar a análise do esquema do banco de dados e criar um arquivo de mapeamento padrão.

```
# D2RQ Namespace
@prefix d2rq:      <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
# Namespace of the ontology
@prefix :         <http://annotation.semanticweb.org/iswc/iswc.daml#> .

# Namespace of the mapping file; does not appear in mapped data
@prefix map:     <file:///Users/d2r/example.ttl#> .

# Other namespaces
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .

map:Database1 a d2rq:Database;
  d2rq:jdbcDSN "jdbc:mysql://localhost/iswc";
  d2rq:jdbcDriver "com.mysql.jdbc.Driver";
  d2rq:username "user";
  d2rq:password "password";
  .

# -----
# CREATE TABLE Conferences (ConfID int, Name text, Location text);

map:Conference a d2rq:ClassMap;
  d2rq:dataStorage map:Database1.
  d2rq:class :Conference;
  d2rq:uriPattern "http://conferences.org/comp/confno@@Conferences.ConfID@";
```

Figura 12 – Exemplo de conteúdo de um arquivo de mapeamento D2RQ

A figura 12 exemplifica o conteúdo de um arquivo de mapeamento D2RQ. Nesse arquivo, encontram-se referências aos namespaces das ontologias utilizadas, além das configurações de conexão com o banco de dados e da especificação da relação entre uma coluna do banco com o equivalente na ontologia.

O dump-rdf<sup>11</sup> pode ser configurado para gerar o documento RDF de saída em diferentes formatos, incluindo RDF/XML (padrão da ferramenta), RDF/XML abreviado, N-TRIPLE e N3.

<sup>10</sup> <http://d2rq.org/>

<sup>11</sup> <http://d2rq.org/dump-rdf>



### **2.8.2.2. NeOn Toolkit**

NeOn Toolkit<sup>12</sup> é um ambiente de engenharia de ontologias, multiplataforma e de código aberto, baseado na IDE Eclipse. Foi desenvolvido durante o projeto NeOn, que foi financiado pela União Européia. Atualmente é mantido e distribuído pela NeOn Technologies Foundation.

Permite o desenvolvimento de ontologias no formato OWL/RDF, com suporte compreensível para todo o ciclo de vida de engenharia da ontologia. Além disso, provê um conjunto extenso de plug-ins (mais de 40 no total), que cobrem uma variedade das atividades do ciclo de vida de engenharia da ontologia, categorizados em tipos, que incluem anotação e documentação, interação humano-ontologia, aquisição de conhecimento, relacionamento de ontologias e etc.

Para que se utilizar o NeOn Toolkit como RDFizer, é necessária a instalação de um plugin chamado ODEMapster. Este plugin fornece uma interface gráfica que permite ao usuário criar, executar e consultar mapeamentos entre ontologias e bancos de dados. Esses mapeamentos são descritos na linguagem R2O, uma linguagem de mapeamento entre ontologias e bancos de dados que possui um conjunto de operações primitivas completo e extensível. Múltiplos mapeamentos R2O podem ser criados para cada ontologia. Esse plugin funciona com ontologias OWL/RDF e bancos de dados MySQL e Oracle.

---

<sup>12</sup> [http://neon-toolkit.org/wiki/About\\_Us](http://neon-toolkit.org/wiki/About_Us)

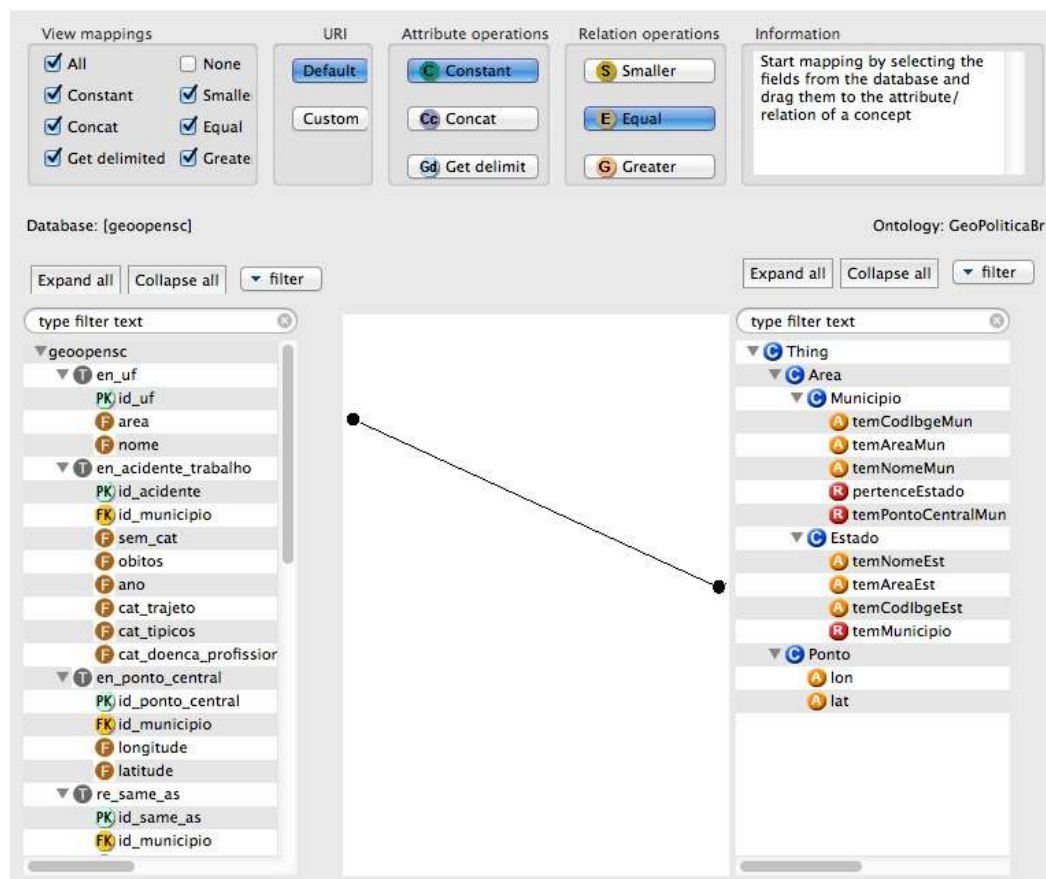


Figura 13 – Exemplo de mapeamento entre base relacional e ontologia utilizando o NeOn Toolkit com plugin ODEMapster

Como visto na figura acima, uma vez que a conexão com o banco de dados seja estabelecida, e a ontologia a ser utilizada esteja definida, o ODEMapster permite com que o usuário realize o mapeamento através da interface gráfica.

Uma vez que o mapeamento esteja concluído, é possível executar a geração do RDF a partir do NeOn Toolkit.

### 2.8.3. Publicação dos Dados

A partir do momento que os dados estão armazenados no formato RDF, eles podem, então, ser publicados. Na abordagem escolhida, a publicação ocorre através de um servidor específico de RDF.

Alguns servidores de RDF utilizam uma estrutura própria para armazenagem do RDF, outros utilizam bancos relacionais. Existem servidores que permitem ambos os tipos de armazenagem. Uma característica comum a todos servidores RDF é o suporte a consultas no padrão SPARQL (capítulo 2.8.4.1) através de um endpoint SPARQL. Pelo fato de suportar

consultas SPARQL, caso o servidor RDF utilizado não possua uma interface própria de linked data, os dados ainda podem ser expostos utilizando ferramentas de interface linked data, como o Pubby<sup>13</sup>.

Como mencionado anteriormente, quando o volume de dados for grande, servidores RDF devem ser levados em consideração, pois são ferramentas robustas. Devido ao grande número de opções, apenas dois servidores serão abordados.

### **2.8.3.1. Sesame**

Sesame<sup>14</sup> é um framework, de código aberto, para consulta e análise de dados em formato RDF, desenvolvido pela empresa Aduna.

Faz parte desse framework, um servidor RDF e um endpoint com suporte a pesquisas realizadas em SPARQL e SeRQL. Outro componente do framework se chama Alibaba, que permite o mapeamento entre classes Java e ontologias e a geração de código fonte Java a partir de ontologias. Essas funcionalidades permitem com que um desenvolvedor acesse ontologias, como FOAF, diretamente no Java.

A API do Sesame difere de outras APIs similares na medida que oferece uma interface empilhável, através da qual funcionalidades podem ser adicionadas, e em que o mecanismo de armazenamento pode ser abstraído da interface de pesquisa. Outros servidores, incluindo o Virtuoso Universal Server, podem ser utilizados através da API do Sesame. Servidores acessados pela API também podem receber funcionalidades novas através da interface empilhável. Um exemplo de uso seria adicionar funcionalidades de indexação para um servidor que não possua isso nativamente.

Embora possua a vantagem da interface empilhável, o Sesame não possui uma interface nativa para representação de linked data, impedindo a navegação através de navegadores de Linked Data. Portanto, a utilização de ferramentas, como o Pubby, se faz necessária para tal fim.

### **2.8.3.2. Pubby**

O Pubby não é um servidor RDF e nem um endpoint SPARQL, mas sim uma aplicação frontend desenvolvida em Java Web para um endpoint SPARQL. Além da interface de linked data para um endpoint SPARQL, local ou remoto, o Pubby também fornece uma interface HTML simples para representar os dados disponíveis para cada recurso e cuida do tratamento

---

<sup>13</sup> <http://www4.wiwiw.fu-berlin.de/pubby/>

<sup>14</sup> <http://www.openrdf.org/>

de redirecionamentos 303 (HTTP) e negociação de conteúdo entre HTML, RDF/XML e Turtle sobre o mesmo recurso. É compatível com os servidores de aplicação Tomcat e Jetty.

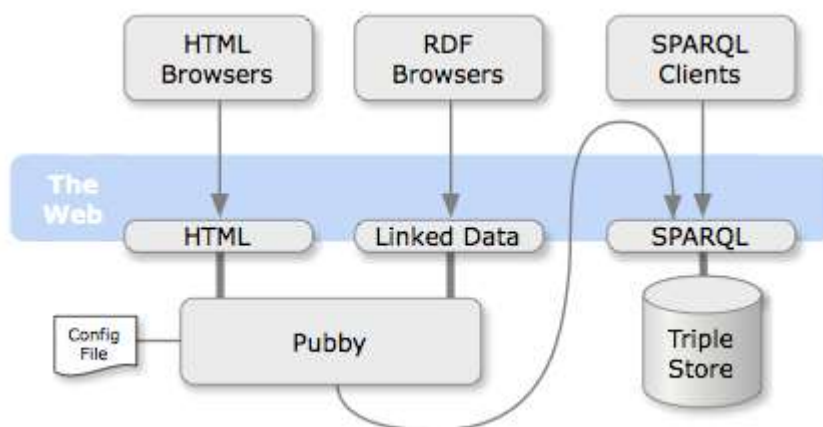


Figura 14 – Arquitetura de funcionamento do Pubby.

O funcionamento do pubby é explicado pela sua capacidade de dereferenciar URIs. Ao configurar o Pubby, é necessário criar um mapeamento que irá traduzir URIs não-dereferenciáveis, encontradas na maioria dos datasets, para URIs dereferenciáveis que serão manipuladas pelo Pubby. Dessa maneira, ele será capaz de lidar com as requisições realizadas, envolvendo essas novas URIs, conectando diretamente no endpoint SPARQL e realizando consultas sobre as URIs originais, retornando os resultado para o usuário.

### 2.8.3.3. Virtuoso Universal Server

O Virtuoso Universal Server<sup>15</sup> foi desenvolvido e é mantido pela OpenLink Software. Trata-se de um sistema híbrido entre middleware e mecanismo de banco de dados, e recebe este nome pois combina, em uma única plataforma, as funcionalidades tradicionais encontradas em RDBMS, ORDBMS, bancos de dados federados, RDF, XML, servidores de aplicações web, servidores de arquivos e servidores de pesquisa em texto livre, com suporte a indexação de texto completa ou parcial.

Uma das vantagens do Virtuoso para o armazenamento e publicação de linked data é que ele oferece a possibilidade de servir os dados desejados em formato RDF a partir de um arquivo RDF, hospedado no servidor, ou diretamente de um banco de dados. É capaz de trabalhar com os mais diversos tipos de RDF, incluindo HTML+RDFa, RDF-JSON, N3, Turtle, TriG, TriX, e RDF/XML. Outras funcionalidades incluem interface REST para operações de

<sup>15</sup> <http://virtuoso.openlinksw.com/>

CRUD, rastreamento e dereferenciamento em tempo real de objetos de Linked Data a partir de suas URIs, indexação de texto completo, etc.

Por se tratar de um servidor multi-modelo de dados, o Virtuoso Universal Server é destinado às empresas que precisam de soluções robustas e eficazes para gestão de dados de diversas fontes, não sendo oferecido gratuitamente na sua versão completa. No entanto, existe uma versão comunitária, de código aberto, com funcionalidades reduzidas, mas que incluem o servidor RDF, necessário para a publicação dos dados, e o endpoint com suporte a consultas SPARQL, necessárias para exploração e análise dos dados.

## 2.8.4. Consumo dos Dados

De nada adianta publicar dados seguindo os padrões de Linked Data, se estes não puderem ser explorados por usuários, consumidos por aplicações ou referenciados por outros data sets. Nesta seção são explicadas algumas maneiras de como navegar por esses dados.

### 2.8.4.1. Consultas SPARQL

SPARQL, cujo nome é um acrônimo recursivo para SPARQL Protocol And RDF Query Language, é uma linguagem de consultas, capaz de recuperar e manipular dados que se encontram no formato RDF. Essa linguagem foi estabelecida como norma pelo RDF Data Access Group (DAWG) do W3C, sendo considerada uma das tecnologias-chave da Web Semântica. No ano de 2008, a linguagem SPARQL 1.0 se tornou uma recomendação oficial da W3C.

Consultas realizadas em SPARQL consistem de triplas, conjunções, disjunções e cláusulas opcionais. Dependendo da finalidade da consulta, o usuário pode optar por quatro tipos diferentes disponibilizados pela linguagem.

- **Consultas SELECT:** Utilizadas para realizar a extração de valores brutos de um endpoint SPARQL. O resultado retorna em formato tabular;
- **Consultas ASK:** Utilizadas apenas para se obter uma resposta verdadeiro/falso de acordo com a consulta realizada no endpoint SPARQL;
- **Consultas DESCRIBE:** Esse tipo de consulta extrai informações de um determinado endpoint SPARQL, retornando os resultados em formato RDF válido.

Adicionalmente, consultas SPARQL utilizam um bloco de instruções WHERE, com a intenção de restringir a consulta, de maneira similar ao SQL, com execução das consultas do tipo DESCRIBE, onde o bloco é considerado opcional.

```

PREFIX abc: <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
  ?x abc:cityname ?capital ;
     abc:isCapitalOf ?y .
  ?y abc:countryname ?country ;
     abc:isInContinent abc:Africa .
}

```

Figura 15 – Exemplo de consulta SPARQL do tipo SELECT

Na figura X está exemplificada uma consulta SPARQL do tipo SELECT, onde o usuário que obter os nomes dos países e suas respectivas capitais situados no continente africano, dentro no namespace apelidado de abc, cujo endereço está descrito no prefixo da consulta.

Embora seja uma linguagem completa, usuários avançados e pessoas com necessidades muito específicas podem encontrar algumas pequenas limitações. Uma delas é a impossibilidade de se utilizar negações. Por exemplo: não é possível realizar uma consulta cujo resultado deve ser uma tabela com todas as pessoas que conhecem João mas que não conhecem Maria.

#### 2.8.4.2. Motores de Busca de Linked Data

Motores de busca são projetados para buscar informações na Web. Os resultados encontrados geralmente são retornados no formato de lista e variam entre páginas da web, imagens, vídeos e outros tipos de documentos. Existem diversos motores de buscas disponíveis atualmente, sendo o que possui maior participação no mercado o motor de busca da Google.

Entretanto, motores de busca convencionais ainda não possuem a capacidade de realizar buscas que retornem resultados em RDF. Desta forma, surge a necessidade da criação de motores de busca para linked data.

Esse tipo de motor de busca pode ser dividido em dois grupos. Em um deles, ficam os motores de busca cuja função é servir como provedor de informações para aplicações mashup. Para atuar dessa maneira, eles realizam buscas por documentos RDF que referenciam a URI repassada pela aplicação, eliminando, dessa maneira, a necessidade da aplicação mashup possuir uma infraestrutura de busca própria. Alguns exemplos desse tipo de motores de busca são Sindice e Swoogle.

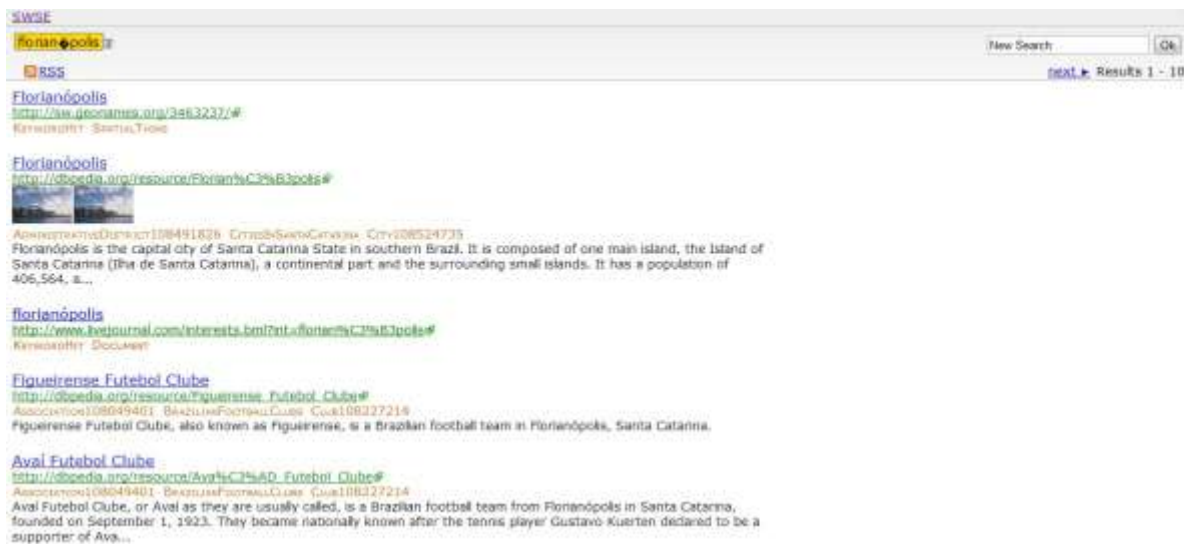


Figura 16 – Resultados de uma busca por “Florianópolis” no SWSE

Compondo o segundo grupo, estão os motores de busca de linked data que se desempenham função similar a de motores convencionais, realizando buscas por palavras-chave e retornando uma lista de resultados. Estes incluem o SWSE<sup>16</sup> e Sig.ma<sup>17</sup>, entre outros.

#### 2.8.4.3. Navegadores de Linked Data

Navegadores Web são as ferramentas utilizadas por usuários da web para navegar no seu conteúdo. Trabalham com diversos padrões de documento, sendo HTML o mais comum. Entretanto, da mesma maneira que motores de busca, navegadores convencionais não possuem uma interface que interprete linked data nativamente. Dessa maneira, embora o usuário possa navegar entre documentos utilizando hiperlinks, ele não poderá navegar entre dados ligados utilizando os links RDF. Logo, navegadores específicos precisaram ser criados para permitir a navegação neste novo padrão, como Tabulator<sup>18</sup> e Marbles<sup>19</sup>.

Navegadores de Linked Data, além de permitir a navegação entre ligações RDF, também possuem a capacidade de agregar informações sobre o mesmo dado provenientes de diversas fontes, facilitando a análise do assunto por parte do usuário.

## 2.9. Aplicações Mashup

<sup>16</sup> <http://swse.org/>

<sup>17</sup> <http://sig.ma/>

<sup>18</sup> <http://www.w3.org/2005/ajar/tab>

<sup>19</sup> <http://marbles.sourceforge.net/>

Antes do advento da web 2.0, o modelo de negócios das empresas que utilizavam a internet ditava que as informações dos seus clientes (consumidores) deveriam ser armazenadas em portais, os quais eram atualizados regularmente. Isso fazia com que o controle estivesse completamente na mão das organizações, forçando os clientes a usarem seus produtos e serviços para obter as informações desejadas.

Com o surgimento da web 2.0, houve uma mudança no comportamento dos usuários da web e suas expectativas, o que ocasionou uma mudança de paradigma no desenvolvimento de aplicações. Desenvolvedores começaram a apostar mais em aplicações web, ao invés das tradicionais aplicações desktop. Como resultado dessa migração, as APIs dessas aplicações passam a ser disponibilizadas para os usuários da web, permitindo que as informações e funcionalidades disponibilizadas pelas APIs possam ser reaproveitadas em outras aplicações, as quais podem ser desenvolvidas por terceiros que não possuam relação alguma com a aplicação original.

O termo aplicação mashup, embora não seja definido formalmente por qualquer órgão de normalização, faz referência a aplicações que, ao utilizarem técnicas de combinação, agregação e visualização em cima de dados provenientes de duas ou mais fontes diferentes, criam uma experiência integrada, antes não existente, para o usuário final.

### 2.9.1. Tipos de Mashup

Nessa seção, são explicadas as diferentes maneiras de se categorizar aplicações mashup. Aplicações mashups variam muito entre si, possuindo diversos tipos de finalidades. Sendo assim, podem ser categorizadas por tipo, gênero e até pelo tipo das APIs utilizadas na sua criação.

Em relação ao tipo, as mashups podem ser divididas entre:

- **Mashups Empresariais:** Essas aplicações geralmente combinam recursos e dados próprios de uma organização com web services externos. Prezam pela segurança da informação e possuem uma interface gráfica rica, que facilita o uso por parte do usuário final e aumenta sua produtividade.
- **Mashups para Consumidores:** São criadas para o público geral da web. Combinam diferentes tipos de dados e serviços, provenientes de múltiplas fontes, em uma única interface. Serve como exemplo uma aplicação que combine um serviço de cartografia com uma fonte de dados sobre criminalidade.
- **Mashups de Dados:** Sua única diferença para as mashups para consumidores é o fato de combinarem tipos de dados similares, mas de fontes diferentes, em



uma única interface. Por exemplo: uma aplicação que apresente notícias sobre um tema desejado, mas de fontes diferentes, em uma interface própria.

Além desses tipos, mashups podem ser distinguidas por gênero. Dentre os mais comuns, estão:

- **Mashups de Mapeamento:** São as mashups que utilizam APIs de cartografia para representar geograficamente um data set que possui informações associadas a uma localidade. A disponibilização da API do Google Maps fez com que esse tipo de mashup ganhasse imensa popularidade, por permitir aos desenvolvedores a possibilidade de mapear os mais diversos tipos de informações. Exemplos incluem mashups que referenciam geograficamente informações sobre ofertas de imóveis, desastres nucleares, temas em evidência em redes sociais e etc. Logo após a liberação da API do Google Maps, outras APIs do gênero se tornaram disponíveis, como a Visual Earth (Microsoft), Yahoo Maps (Yahoo) e MapQuest (AOL).
- **Mashups de Fotos e Vídeos:** Este tipo de mashup tem origem na emergência de serviços de hospedagem de fotos, como o Flickr. As APIs de compartilhamento de fotos oferecem metadados sobre as fotos, que podem indicar as pessoas presentes na foto, quem foi o fotógrafo, a localização de onde a foto foi tirada, a sua data, comentários, tags associadas e etc. Com essa diversidade de informações em mãos, desenvolvedores podem criar diversos tipos de aplicações como, por exemplo, uma linha do tempo de um determinado ponto turístico, mostrando quem foram os visitantes que por lá passaram e tiraram fotos ou gravaram vídeos.
- **Mashups de Buscas e Compras:** Antes mesmo do termo mashup vir à tona, já existiam serviços que permitiam a agregação de resultados de pesquisa e a comparação de preços de mercadorias, que são as finalidades dessa categoria. Sites de compras online, como Amazon e eBay, disponibilizam APIs para acesso às informações de suas mercadorias. No Brasil, serve como exemplo o site BuscaPé.
- **Mashups de Notícias:** Consiste de aplicações que utilizam diversas fontes para disponibilizar informações aos usuários relacionadas aos seus temas de interesse. Também incluem aplicações que permitem que o usuário indique as fontes das quais quer receber notícias atualizadas. Geralmente utilizam feeds RSS. Um exemplo desse tipo de mashup é o Google Reader.

Categorização por tipo de API divide as mashups entre as que usam APIs de dados, funções ou ambas. APIs de dados podem fornecer diferentes informações, como dados indexados (documentos, imagens, vídeos, etc.), dados geográficos, feeds de notícias. Por sua vez, as APIs de função também possuem uma grande diferenciação entre si, oferecendo recursos como convertidores de informações (tradutores, processadores de dados, etc.), serviços de comunicação (e-mail, notificações, mensagens instantâneas, etc.), serviços de pagamentos e demais.

## 2.9.2. Arquitetura

Normalmente a arquitetura de uma aplicação mashup consiste de três camadas diferentes.

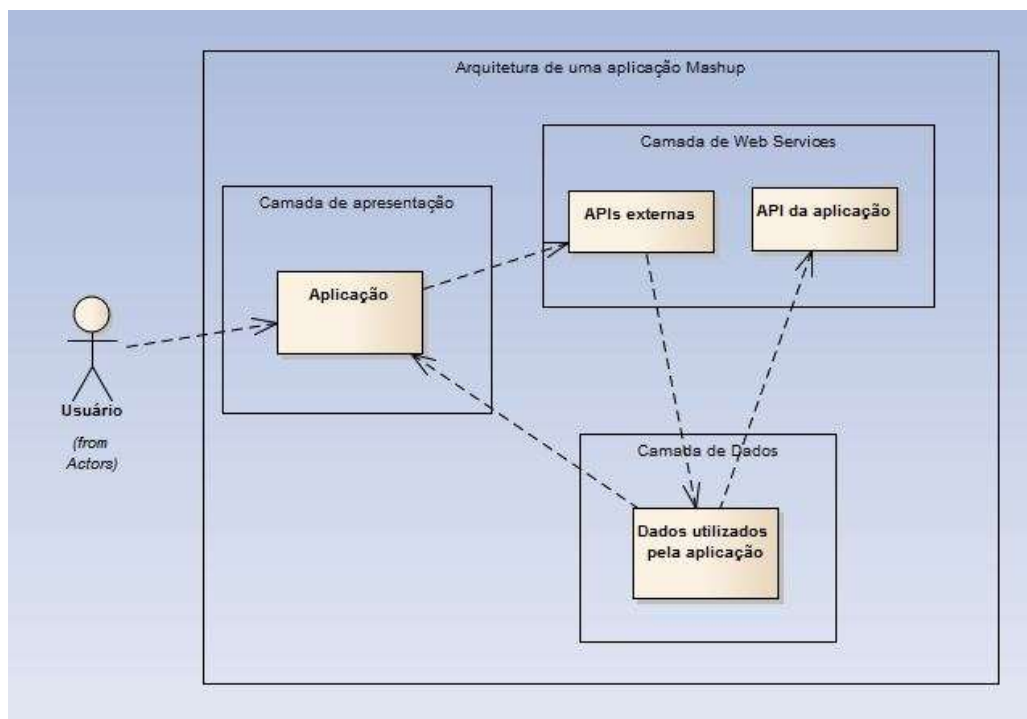


Figura 17 – Camadas da arquitetura de uma aplicação mashup

A primeira é a camada de apresentação. Nessa camada está a interface gráfica, onde ocorre a interação com o usuário final. As tecnologias mais comumente utilizadas nessa camada são HTML/XHTML, CSS, Javascript e Ajax.

Na camada de Web Services, ocorre a interação com os web services e APIs externas que são utilizadas para o funcionamento da aplicação. Além disso, também é onde a API, caso exista, da aplicação será disponibilizada. Nessa camada são utilizadas tecnologias como XMLHttpRequest, XML-RPC, JSON-RPC, SOAP, REST.

Por último, existe a camada de Dados, onde ocorre o armazenamento, recebimento e envio dos dados utilizados pela aplicação. As tecnologias utilizadas nesta camada são XML, JSON e KML.

Além das camadas descritas, existe também uma distinção relacionada ao local onde ocorre o processamento das informações. As aplicações mashup podem ser do tipo *web-based*, onde a combinação e formatação dos dados ocorre no navegador do usuário. Do outro lado, encontram-se as aplicações mashup *server-based*, as quais realizam o processamento em um servidor remoto e transmitem as informações no seu estado final para o navegador do usuário.

### **2.9.3. Desafios no desenvolvimento**

Por trabalhar com informações de múltiplas fontes, existem alguns desafios de integração das diferentes informações durante o desenvolvimento de uma aplicação mashup.

Grande parte das informações é descrita em texto, o que facilmente pode acarretar ambiguidades. Um simples exemplo é a *IBM*, multinacional sediada nos Estados Unidos da América. Pode-se encontrar uma notícia de uma determinada fonte que faça referência à *IBM*, enquanto que em outra fonte, encontra-se uma notícia sobre a *International Business Machines Corporation*. Ambas estão fazendo referência à mesma organização, mas com escritas diferentes. Além disso, por estarem expressas em linguagem humana explícita, sem metadados sobre a informação, o processamento por máquinas é dificultado. Dessa maneira, o desenvolvedor precisa utilizar técnicas e softwares especializados em integração de dados para conseguir realizar a ligação entre as informações que deseja utilizar no mashup.

Além disso, dados estruturados estão disponibilizados em uma infinidade de formatos. O primeiro passo, então, é elevá-los a um formato comum. Mesmo com os dados disponíveis em um formato comum, na prática, as fontes diferem na maneira de expressar o mesmo fato. Essa diferença ocorre tanto em nível de objetos individuais quanto em nível de esquema. Por exemplo, um município brasileiro na DBpedia será identificado pela sua URI, enquanto para o IBGE ele será identificado por um número identificador. Sendo assim, o desenvolvedor precisa ter a sua disposição métodos para conciliar as diferentes representações de objetos e esquemas.

Existem também dificuldades relacionadas ao nível de abstração utilizada pelas fontes na hora de disponibilizar suas informações. Uma fonte de informações normalmente utilizará uma taxonomia ou classificação diferente de outra fonte. Um problema relacionado é o uso de moedas locais (Real vs. Euro), que têm de ser conciliadas, a fim de tornar os dados de diferentes fontes comparáveis e passíveis de análise.

Por ultimo, a qualidade dos dados pode causar dificuldades para o desenvolvedor. Dados costumam estar propensos a erro, e a integração entre dados de fontes autônomas geralmente agrava esse problema. Dados errados por si já são um problema, mas tornam-se ainda piores quando são utilizados como base para inferir novos dados. Tornam-se necessários métodos e técnicas para que o desenvolvedor seja capaz de aferir a integridade dos dados, sua precisão, sua coerência e etc. Existe também o cenário onde fontes diferentes podem disponibilizar informações conflitantes sobre um mesmo objeto específico e o desenvolvedor da mashup precisa resolver qual das informações tem mais valor sobre a outra.

## **3. Publicação dos Dados Lincados**

Como visto no capítulo 2, para publicarmos os dados abertos encontrados na forma de dados lincados, estes precisam estar de acordo com os 8 princípios dos dados abertos, além de seguirem os 4 princípios fundamentais de linked data. O propósito deste capítulo é apresentar o processo adotado para que estes requisitos fossem cumpridos, permitindo a publicação de um conjunto de dados geográficos do Brasil que estivessem de acordo com os objetivos propostos pelo projeto (seção 1.2.1).

### **3.1. Proposta do trabalho**

#### **3.1.1. Obtenção dos dados**

Para publicar dados na web, primeiro temos que identificar o domínio a ser trabalhado. Neles encontram-se recursos cujas propriedades e relacionamentos queremos descrever neste trabalho.

Como o domínio de dados estatísticos e geográficos, escolhido para publicação neste trabalho, abrange uma diversidade de dados muito grande, é necessário também delimitar o escopo dos dados que serão trabalhados. Uma vez que o escopo de dados esteja delimitado, estes deverão ser obtidos de fontes oficiais que os disponibilizem abertamente.

#### **3.1.2. Definição da ontologia**

Com o domínio de dados definido, será necessário representá-lo através de uma, ou mais, ontologias. A utilização de ontologias se faz necessária na representação do domínio, pois estas fornecem um vocabulário com os conceitos e relacionamentos existentes no domínio que são passíveis de compreensão por máquinas. Sendo assim, também permitem a comunicação entre humanos e máquinas e a interoperabilidade entre sistemas.

De modo a evitar ambiguidades, o reuso de ontologias já existentes, cujo vocabulário expresse o domínio escolhido, será uma alternativa analisada durante a escolha da ontologia. Essa atitude também faz com que não seja necessário um estudo mais detalhado sobre o domínio, a fim de criar-se uma nova ontologia que o defina, evitando redundâncias.

#### **3.1.3. Migração dos dados para uma base relacional**

Embora os dados desejados estejam disponíveis de forma estruturada, fontes de dados diferentes utilizam maneiras diferentes para disponibilizar seus dados. Portanto, a migração desses dados para uma base relacional será realizada.

Esse passo vai permitir o mapeamento entre os dados e a ontologia escolhida, o que facilitará a geração do arquivo RDF.

Utilizaremos algum SGBD que esteja disponível gratuitamente e possua reconhecimento no mercado.

### **3.1.4. Alinhamento dos datasets**

Como um dos princípios de Linked Data é o relacionamento entre dados, é necessário um passo no qual os recursos do RDF que será gerado nesse trabalho sejam relacionados a recursos de outros data sets.

Para isso, teremos que identificar quais data sets possuem recursos que serão referenciados. A partir daí, será necessária a extração das URIs desses recursos, para que sejam referenciadas no nosso RDF.

### **3.1.5. Geração do RDF**

A geração do RDF dependerá do mapeamento entre a base relacional e a ontologia escolhida.

Ferramentas de mapeamento foram descritas no capítulo anterior, assim como ferramentas que automatizam a geração do RDF. Uma delas deverá ser escolhida para a realização deste passo. Além de mapear o relacionamento entre dados e ontologia, será necessário também mapear o relacionamento entre as URIs, obtidas no passo anterior, com os recursos serão gerados.

### **3.1.6. Publicação e visualização dos dados**

Com o RDF em mão, precisaremos de uma ferramenta que permita sua publicação na web. Um servidor de RDF será escolhido nesse passo. Caso o servidor escolhido não possua uma interface de linked data nativa, escolheremos também uma ferramenta que disponibilize essa funcionalidade para o nosso data set.

Em cima dessa estrutura será desenvolvida uma aplicação mashup. O objetivo dessa aplicação será agregar as informações em uma interface que permita a visualização e análise

dos dados de maneiras intuitivas para o usuário. Esta interface possuirá um mapa, que será proveniente de alguma API disponível gratuitamente na web.

Além da aplicação, será oferecido um endpoint SPARQL para que usuários realizem consultas nesse formato, caso desejem.

## 3.2. Execução do trabalho

O primeiro passo deste processo foi a busca e em seguida a seleção dos dados que posteriormente seriam publicados. Dado que as fontes e formatos dos dados encontrados eram os mais diversos tivemos o esforço de reuni-los em uma base de dados relacional. A partir deste modelo desenvolvemos duas ontologias para representar o conjunto de conceitos do domínio dos dados encontrados.

Tendo a base de dados populada com os dados e as ontologias que representam este conjunto, utilizamos a ferramenta dump-rdf (capítulo 4.4.2) para gerar os dados triplificados no formato RDF/XML (seção 2.5.1). Nesta etapa também fizemos os links com outros datasets. Através de consultas no endpoint SPARQL do DBPEDIA, encontramos várias URIs que representavam os mesmos dados que tínhamos na nossa base.

Estes dados já nos padrões de linked data foram publicados seguindo 3 abordagens, cada uma com suas vantagens e desvantagens:

- Publicação do arquivo RDF estático
- Exposição direta do banco de dados relacional, utilizando o D2R Server para mapear o modelo relacional para o ontológico
- Sesame como repositório RDF
  - Endpoint SPARQL para consultas,
  - Interface RESTful HTTP com suporte a SPARQL
  - Utilização do Pubby como interface linked data

Utilizando a interface RESTful HTTP fornecida pelo Sesame como provedora dos dados, desenvolvemos uma aplicação que fornece uma forma mais clara de visualizar os dados.

### 3.2.1. Dados a serem publicados

Os dados utilizados neste trabalho são referentes aos municípios do estado de Santa Catarina. Inicialmente tínhamos um arquivo XLS, contendo o nome dos municípios e seu respectivo código do IBGE. Esse código foi importante durante todo o desenvolvimento, pois a maioria dos fornecedores dos dados o utiliza como identificador para o município evitando o uso

de outras técnicas para fazer a relação Dado x Município. Além disso, a adoção deste código pode ajudar outros projetos que precisem fazer o link com o nosso dataset.

Como fontes de dados priorizamos aquelas de origem governamental, pois garantem a disponibilização de dados atuais de forma aberta, facilitando a adoção dos princípios de disponibilização de dados abertos.

Após finalizarmos as buscas chegamos ao seguinte conjunto de dados:

Dado	Formato	Fonte
Divisão territorial	XLS	IBGE
Censo 2010	XLS	IBGE
Acidentes de trabalho	XLS	Ministério da previdência
Arrecadação de ICMS	XLS	Secretaria de Estado da Fazenda de Santa Catarina
Repasse de ICMS	XLS	Secretaria de Estado da Fazenda de Santa Catarina
Latitude, Longitude, Área e Altitude	DBF	Departamento de Informática do SUS
Densidade, Distancia a capita, IDH 1991, IDH 2000, Gentílico, Desmembrado de	XLS	Secretaria de Estado da Fazenda de Santa Catarina

**Tabela 1 - Fontes de dados encontradas para utilização neste trabalho Fonte: Autores**

Os arquivos foram disponibilizados em vários formatos, principalmente CSV e XLS. Em virtude desse motivo tivemos que utilizar uma ferramenta ETL, neste caso o Kettle, da suite Pentaho, que nos permitiu fazer a integração de dados, tratando-os quando necessário, para em seguida inserí-los na base de dados.

Para armazenar os dados na forma relacional utilizamos o banco de dados MySQL. Além de ser um dos mais robustos, é totalmente aberto (open source e gratis). Porém, o principal motivo da escolha deste sistema foi a sua compatibilidade com a ferramenta NeonToolkit (em conjunto com os plugins R2O e ODEmapster) responsável pelo mapeamento entre o modelo relacional e o ontológico.

### **3.2.1.1. Relacionamento com dados externos**

Com todos os dados desejados já disponíveis, resolvemos identificar data sets com dados relevantes para realizar a ligação entre eles, seguindo os princípios de linked data. Escolhemos a DBpedia como data set ideal para essa ligação, pois, como observado durante a



fundamentação teórica, sua ontologia possui uma quantidade enorme de informações sobre municípios.

Como a relação entre recursos ocorre através da propriedade owl:sameAs, fez-se necessária a obtenção das URIs dos municípios catarinenses presentes na wikipedia. Para esse fim, utilizamos o endpoint SPARQL da dbpedia e realizamos uma consulta que retornasse a URI dos municípios catarinenses junto de seus nomes, como visto na figura abaixo.

```

Query Text
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?uri ?label WHERE {
?uri rdf:type dbo:Place .
?uri dbo:isPartOf <http://dbpedia.org/resource/Santa_Catarina_%28state%29> .
OPTIONAL {?uri rdfs:label ?label . FILTER (lang(?label) = "pt")} .
} ORDER BY ?uri

```

Figura 18 – Query SPARQL para obter-se a lista de URIs, atrelada ao nome, dos municípios catarinenses

O retorno dessa consulta foi uma lista com a URI de 263 municípios catarinenses, dos 293 existentes. Para armazenar as URIs no banco, relacionando-as com as entidades já existentes, foi utilizada novamente a ferramenta Kettle e criada uma tabela adicional para sameAs.

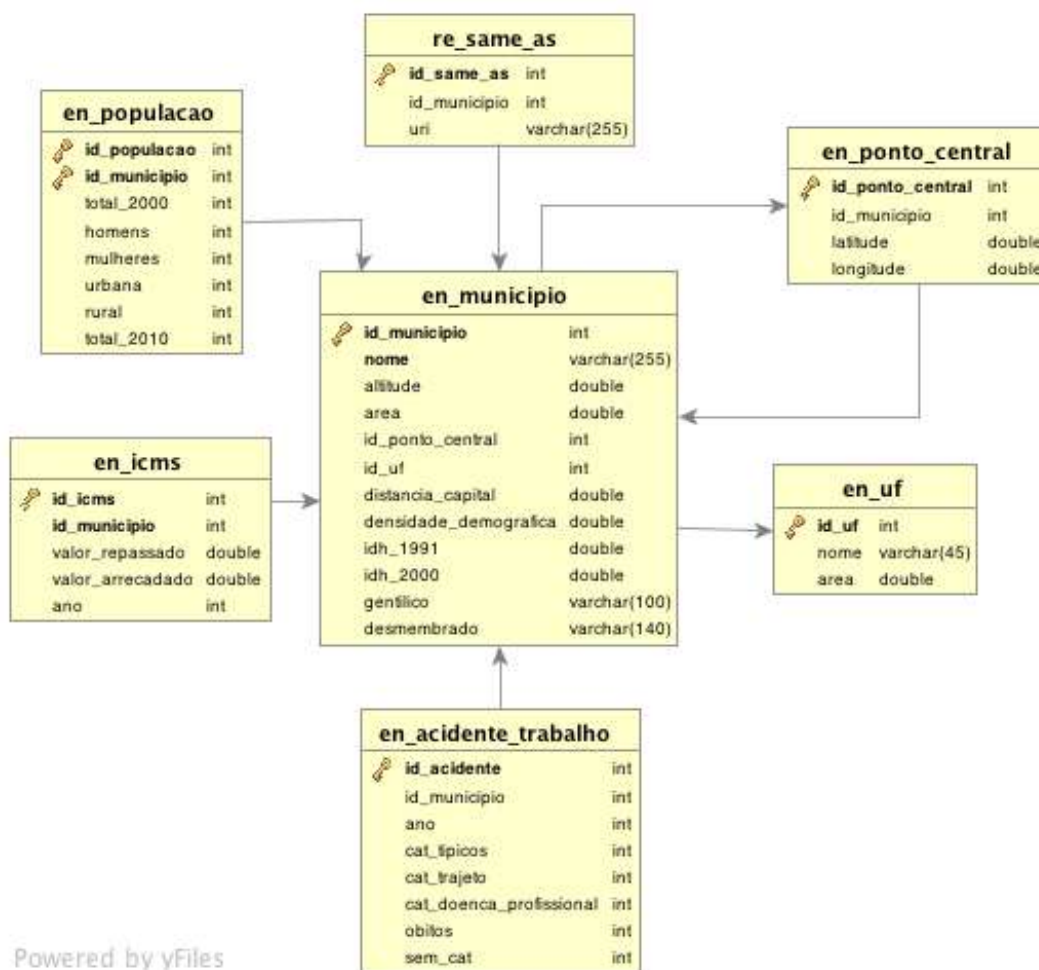
uri	label
http://dbpedia.org/resource/%C3%81gua_Doce	"Água Doce"@pt
http://dbpedia.org/resource/%C3%81guas_Frias	"Águas Frias (Santa Catarina)"@pt
http://dbpedia.org/resource/%C3%81guas_Mornas	"Águas Mornas"@pt

Figura 19 – Parto do resultado da pesquisa, no formato HTML

Tentamos também obter as URIs dos recursos do data set GeoNames. No entanto, como o GeoNames não disponibiliza um endpoint SPARQL, apenas um conjunto de web services, acabamos não relacionando os recursos desse data set com os nossos recursos.

### 3.2.1.2. Modelo Relacional

Uma vez que todas as fontes de dados estavam convertidas para o formato relacional, chegamos a seguinte modelagem:



Powered by yFiles

**Figura 20 - Modelo relacional final Fonte: Autores**

Apesar do escopo deste trabalho abordar somente dados dos municípios de Santa Catarina, definimos um modelo de dados que suportasse a inclusão do restante de municípios do Brasil.

### 3.2.2. Ambiente de desenvolvimento

Esta breve seção tem como objetivo apresentar o ambiente de desenvolvimento que foi utilizado durante todo o projeto.

Por uma questão de flexibilidade, custo, principalmente facilidade, instanciamos uma máquina virtual na plataforma Amazon Cloud EC2, rodando a versão 2008 do Windows Server.

Os softwares necessários para a realização do projeto foram instalados neste ambiente.

São eles:

- Apache Tomcat 7 (Porta 8080) como servidor HTTP e contêiner web
- Java 7 SDK – Plataforma Java Standard Development Kit
- MySQL (Porta 3660) – SGBD

Sesame (Rodando no container Tomcat ) – Servidor rdf  
D2R Server (Servidor wrapper RDB2RDF linked data, dump de arquivos rdf);  
Neon Toolkit (+ Plugin ODEMapster)– Ferramenta para criação e edição de ontologias, mapeamento entre modelo relacional e ontológico  
Protegé – Ferramenta para criação e edição de ontologias

Toda a gerência foi feita através de acesso remoto visual, disponibilizado pelo próprio sistema operacional.

### **3.2.3. Geração das ontologias**

Com todos os dados já reunidos em um banco de dados, o passo seguinte foi criar as ontologias que representassem esse domínio.

#### **3.2.3.1. GeoPoliticaBr**

Começamos esta etapa utilizando a ferramenta NeOn Toolkit para a geração e manutenção das ontologias. A primeira ontologia gerada, denominada GeoPoliticaBr, representa a divisão territorial do Brasil para estados e municípios. Ela mapeia algumas características como, latitude, longitude, área. Para este caso, usamos uma abordagem mais genérica, buscando a criação de um modelo que pudesse ser utilizado em outros trabalhos que também precisam representar estes conceitos. A figura 21 mostra a estrutura de classes da ontologia GeoPoliticaBR

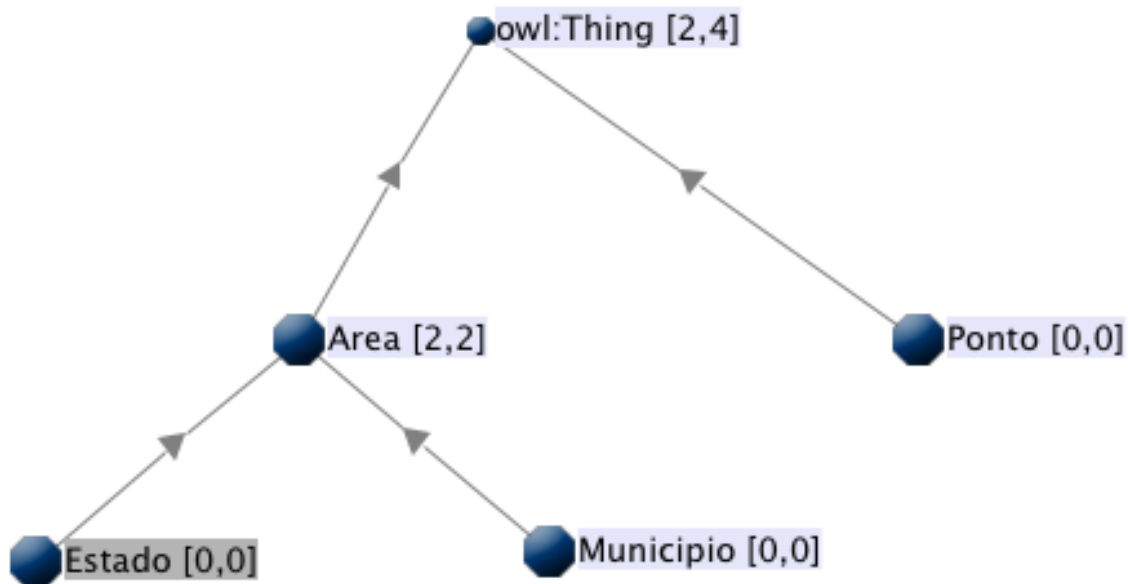


Figura 21 – Ontologia GeoPolíticaBr Fonte: Autores

### 3.2.3.2. DadosBr

Até este ponto a ferramenta utilizada atendia as necessidades, porém começamos a enfrentar muitos problemas de instabilidade, com frequente perda de dados. Por este motivo resolvemos mudar de editor, passando a utilizar o Protegé, um framework para gerência de conhecimento e edição de ontologias desenvolvido pela Stanford University School of Medicine.

Para representar os dados específicos do nosso projeto, foi preciso desenvolver uma nova ontologia, já que a GeoPolíticaBr foi pensada para ser reutilizada em outros trabalhos de modo que não aborda todo o nosso conjunto de dados. Criamos então a DadosBr, uma ontologia que representa somente as propriedades de um município específicas ao nosso trabalho.

Apesar de ser uma nova ontologia, a DadosBr também representa os dados de um município, cuja definição está na ontologia GeoPolíticaBr. Desta forma, tivemos que importar a GeoPolíticaBr, permitindo caracterizar que as propriedades contidas na DadosBr são referentes a mesma classe Município.

As propriedades representadas pela ontologia DadosBr podem ser vistas na imagem a seguir:



Figura 22 – Ontologia DadosBr Fonte: Autores

### 3.2.4. Geração do arquivo RDF

Tendo todos os dados já reunidos em um banco de dados, o passo seguinte foi realizar o mapeamento entre o modelo relacional e o modelo ontológico, resultando em um arquivo rdf.

#### 3.2.4.1. NeOn Toolkit + ODEMapster

A princípio tentamos gerar o arquivo através da ferramenta NeOn Toolkit junto com o plugin ODEMapster, que permitem através de uma interface gráfica realizar o mapeamento entre o dados do banco de dados e a ontologia para em seguida exportar um arquivo rdf.

Nos casos mais simples, como dados que se encontravam em uma só tabela, a ferramenta atendeu as necessidades, gerando corretamente o arquivo. Porém, para as situações onde havia um relacionamento entre tabelas através de chave estrangeira na maior parte dos casos, a ferramenta apresentou dificuldades que nos impediram de dar continuidade.

#### 3.2.4.2. Dump-rdf

Como alternativa a ferramenta descrita no capítulo anterior encontramos outra, `dump-rdf`, que também permite a criação de um arquivo `rdf` a partir de um arquivo de mapeamento.

Um pré-requisito para a utilização da ferramenta `dump-rdf` é a existência de um arquivo que faça o mapeamento entre a base de dados relacional e as ontologias. Para isso utilizamos uma outra ferramenta, também da plataforma D2RQ, chamada **generate-mapping**. Ela nos permitiu a geração de um arquivo de mapeamento default para nossa base de dados.

Utilizamos o seguinte comando para executar a ferramenta:

```
generate-mapping -u root jdbc:mysql:///geopencsc
```

Os parâmetros utilizados são:

- `-u` : usuário do banco de dados
- `jdbc:mysql:///geopencsc` : conexão com o banco de dados

Com base neste arquivo fizemos algumas alterações necessárias para atender as especificidades do nosso projeto. As principais foram :

- adicionar os namespaces das ontologias geradas por nós
- adicionar url para as classes das ontologias
- adicionar o mapeamento `sameAs`, responsável pela ligação com outro dataset

Feitas as modificações chegamos ao arquivo final, que pode ser visto em parte na figura a seguir:

```

@prefix map: <d2r-mappings/testeSaida.ttl#> .
@prefix db: <> .
@prefix vocab: <http://www.lodkem.ufsc.br/vocab/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix jdbc: <http://d2rq.org/terms/jdbc/> .
@prefix geo: <http://lodkem.ufsc.br/onto/GeoPoliticaBr#> .
@prefix dados: <http://lodkem.ufsc.br/onto/DadosBr#> .

map:database a d2rq:Database;
  d2rq:jdbcDriver "com.mysql.jdbc.Driver";
  d2rq:jdbcDSN "jdbc:mysql://127.0.0.1/geopencsc";
  d2rq:username "root";
  d2rq:password "";
  jdbc:autoReconnect "true";
  jdbc:zeroDateTimeBehavior "convertToNull";
  .

# Table en_municipio
map:en_municipio a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#Municipio/@@en_municipio.namelurlify@";
  d2rq:class geo:Municipio;
  d2rq:classDefinitionLabel "en_municipio";
  .
map:en_municipio_id_municipio a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:en_municipio;
  d2rq:property geo:temCodIbgeMun;
  d2rq:propertyDefinitionLabel "en_municipio id_municipio";
  d2rq:column "en_municipio.id_municipio";
  d2rq:datatype xsd:int;
  .

```

**Figura 23 – Arquivo de mapeamento Fonte: Autores**

Tendo este arquivo, o passo seguinte foi utilizar a ferramenta dump-rdf para exportar os dados da base relacional para um arquivo rdf, passando como parâmetro o arquivo de mapeamento. O comando responsável pela chamada da ferramenta foi o seguinte:

**dump-rdf -m testeSaida.ttl -f “RDF/XML” -o municipios.rdf**

Os parâmetros utilizados são:

- -m : arquivo de mapeamento entre a base relacional e ontologias
- -f : formato de saída do arquivo. Escolhemos RDF/XML pois este possui maior suporte em relação aos outros
- -o : nome do arquivo rdf que será gerado

O resultado deste passo foi a geração do arquivo rdf com os dados da base relacional traduzido para o formato RDF/XML. Uma parte deste arquivo pode ser conferida na seguinte imagem:

```

<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:map="file:///Users/guilhermetiscoski/Documents/d2rq-0.8/d2r-mappings/testeSaida.ttl#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:geo="http://lodkem.ufsc.br/onto/GeoPoliticaBr#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:db="file:///Users/guilhermetiscoski/Documents/d2rq-0.8/municipios-sameAs10.rdf"
  xmlns:dados="http://lodkem.ufsc.br/onto/DadosBr#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:vocab="http://www.lodkem.ufsc.br/vocab/resource/" >
  <rdf:Description rdf:about="http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#Municipio/Rio_dos_Cedros">
    <geo:pertenceEstado rdf:resource="http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#Estado/Santa_Catarina"/>
    <geo:temPontoCentralMun rdf:resource="http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#Ponto/216"/>
    <dados:temPopulacaoHomens rdf:datatype="http://www.w3.org/2001/XMLSchema#int">5267</dados:temPopulacaoHomens>
    <dados:temPopulacaoMulheres rdf:datatype="http://www.w3.org/2001/XMLSchema#int">5013</dados:temPopulacaoMulheres>
    <dados:temPopulacao2010 rdf:datatype="http://www.w3.org/2001/XMLSchema#int">10280</dados:temPopulacao2010>
    <geo:temNomeMun rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Rio dos Cedros</geo:temNomeMun>
    <dados:temIDH2000 rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.817E0</dados:temIDH2000>
    <dados:temDistanciaCapital rdf:datatype="http://www.w3.org/2001/XMLSchema#double">555.654E0</dados:temDistanciaCapital>
    <dados:temGentilico rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Riocedrense</dados:temGentilico>
    <dados:temDensidadeDemografica rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.817E0</dados:temDensidadeDemografica>
    <dados:temIDH1991 rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.738E0</dados:temIDH1991>
    <geo:temAreaMun rdf:datatype="http://www.w3.org/2001/XMLSchema#double">555.654E0</geo:temAreaMun>
    <geo:temCodIbgeMun rdf:datatype="http://www.w3.org/2001/XMLSchema#int">4214706</geo:temCodIbgeMun>
    <dados:foiDesmembradoDe rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Timbó</dados:foiDesmembradoDe>
    <rdf:type rdf:resource="http://lodkem.ufsc.br/onto/GeoPoliticaBr#Municipio"/>
    <dados:temCatTrajeto rdf:datatype="http://www.w3.org/2001/XMLSchema#int">11</dados:temCatTrajeto>
    <dados:temSemCat rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</dados:temSemCat>
    <dados:temObitosAcidenteTrabalho rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</dados:temObitosAcidenteTrabalho>
    <dados:temCatDoencaProfissional rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</dados:temCatDoencaProfissional>
    <dados:temCatTipico rdf:datatype="http://www.w3.org/2001/XMLSchema#int">47</dados:temCatTipico>
    <dados:temArrecadacaoICMS rdf:datatype="http://www.w3.org/2001/XMLSchema#double">5970812.94E0</dados:temArrecadacaoICMS>
    <dados:temRepassoICMS rdf:datatype="http://www.w3.org/2001/XMLSchema#double">5122010.18E0</dados:temRepassoICMS>
    <owl:sameAs rdf:resource="http://dbpedia.org/resource/Rio_dos_Cedros"/>
  </rdf:Description>

```

Figura 24 – Arquivo RDF Fonte: Autores

## 3.2.5. Publicação dos dados

### 3.2.5.1. Publicando o arquivo estático

Gerar arquivos rdf estáticos e disponibilizá-los em um servidor web é a maneira mais simples e rápida de publicar dados ligados. Seguindo uma boa prática de publicação de arquivos rdf estáticos, utilizamos o formato de serialização RDF/XML, pois é o mais suportado em ferramentas que consomem dados ligados.

Já tendo o arquivo rdf gerado com sucesso, como explicado no capítulo 4.4, bastou somente publicá-lo em um servidor web. Por ser somente um arquivo, escolhemos disponibilizá-lo através do Apache Tomcat, um contêiner web para aplicações que também fornece um serviço de servidor http.

Para disponibilizar o arquivo através do Tomcat, foi preciso fazer uma simples configuração a fim de mapear a pasta local aonde que o arquivo estava localizado e o contexto da requisição web. Este ajuste consistiu na adição de uma linha, entre a tag <Host>, no arquivo de configuração do servidor chamado server.xml.

```
<Context docBase="C:\dados" path="/geopensc" reloadable="false"/>
```



As propriedades que foram preenchidas são as seguintes:

- docBase : pasta local onde o arquivo rdf se encontra
- path : contexto da requisição web para acessar o arquivo. Ex:  
`http://localhost:8080/geoopensc/arquivo.rdf`
- reloadable : deixamos a opção default, **false**

Feito isso, foi preciso somente ativar o servidor para que o arquivo pudesse ser acessado.

Para facilitar a descoberta deste arquivo, adicionamos um link que aponta para ele na aplicação mashup desenvolvida, apresentada mais adiante.

### **3.2.5.2. Publicando os dados a partir de um wrapper RDB2RDF**

Esta abordagem também demonstrou ser bem simples para publicação de linked data. Ela é vantajosa quando o grupo de dados é alterado com frequência, tornando inviável a criação de arquivos rdf a cada mudança.

Através do D2R Server, outro componente da plataforma D2RQ, foi possível disponibilizar todo o conteúdo da nossa base de dados, sem precisar gerar o arquivo rdf.

O D2R Server aproveita o mesmo arquivo de mapeamento utilizado na geração do arquivo rdf, capítulo 4.4.2, porém ao invés da saída ser um registro rdf estático o serviço fornece uma interface amigável ao usuário. Ela consiste em páginas web pré-formatadas, que são preenchidas com os dados dinamicamente.

Para instanciar o D2R Server executamos o seguinte comando:

```
d2r-server -b http://177.71.183.112:9090 -p 9090 testeSaidaFinal.ttl
```

Os parâmetros utilizados são:

- -b : url onde o serviço estará disponível (obrigatório para tornar acessível de máquinas ou se a porta utilizada for diferente da porta padrão 2020)
- -p : porta a ser utilizada ao invés da porta padrão (2020)
- testeSaidaFinal.ttl : arquivo de mapeamento

Um exemplo da interface disponibilizada pelo D2R Server pode ser visto na imagem a seguir:

## Description of

<http://177.71.183.112:9090/resource/municipio/Florian%C3%B3polis>  
 Resource URI: <http://177.71.183.112:9090/resource/municipio/Florian%C3%B3polis>



[Home](#) | [All en municipio](#)

Property	Value
dados:foiDesmembradoDe	Laguna (xsd:string)
geo:pertenceEstado	< <a href="http://177.71.183.112:9090/resource/estado/Santa_Catarina">http://177.71.183.112:9090/resource/estado/Santa_Catarina</a> >
owl:sameAs	< <a href="http://dbpedia.org/resource/Florian%C3%B3polis">http://dbpedia.org/resource/Florian%C3%B3polis</a> >
geo:temAreaMun	433.317E0 (xsd:double)
dados:temArrecadacaoIcms	2.16994264401E9 (xsd:double)
dados:temCatDoencaProfissional	99 (xsd:int)
dados:temCatTipico	1341 (xsd:int)
dados:temCatTrajeto	436 (xsd:int)
geo:temCodIbgeMun	4205407 (xsd:int)
dados:temDensidadeDemografica	0.875E0 (xsd:double)
dados:temDistanciaCapital	433.317E0 (xsd:double)
dados:temGentilico	Florianopolitano (xsd:string)
dados:temIDH1991	0.824E0 (xsd:double)
dados:temIDH2000	0.875E0 (xsd:double)
geo:temNomeMun	Florianópolis (xsd:string)
dados:temObitosAcidenteTrabalho	3 (xsd:int)
geo:temPontoCentralMun	< <a href="http://177.71.183.112:9090/resource/ponto/88">http://177.71.183.112:9090/resource/ponto/88</a> >
dados:temPopulacao2010	421203 (xsd:int)
dados:temPopulacaoHomens	203093 (xsd:int)
dados:temPopulacaoMulheres	218110 (xsd:int)
dados:temRepasseIcms	1.1526503857E8 (xsd:double)
dados:temSemCat	1267 (xsd:int)
rdf:type	geo:Municipio

Generated by [D2R Server](#)

**Figura 25 – Interface fornecida pelo D2R Server Fonte: Autores**

### 3.2.5.3. Publicando os dados a partir de um servidor RDF

Para esta etapa, foi preciso analisar dentre as opções possíveis, qual servidor atenderia melhor as necessidades do nosso projeto. Depois de uma pesquisa, chegamos a dois candidatos: Virtuoso Universal Server e Sesame.

Como visto no capítulo 2.8.3.3 o Virtuoso Universal Server é uma ferramenta muito robusta, que atende aos mais diversos casos. Porém, toda essa quantidade de serviços oferecidos apresenta uma complexidade maior de implementação, tornando-se inviável para o nosso projeto.

Já o Sesame, visto no capítulo 2.8.3.1, é um framework que além de outras funcionalidades, fornece um servidor para armazenar triplas RDF. Sua complexidade de implementação é baixa, pois roda em um contêiner web como o Apache Tomcat, que já vinha sendo usado no projeto. Apesar de não oferecer uma interface Linked Data, é possível contornar este problema utilizando o Pubby, como será explicado no capítulo 4.5.3.1. Por atender as nossas necessidades aliado a uma baixa complexidade de implementação, escolhemos o Sesame para ser nosso servidor RDF.

#### **3.2.5.3.1. Deploy do servidor**

O processo de deploy do Sesame é muito simples. Basta adicionar os arquivos `openrdf-sesame.war`, `openrdf-workbench.war` no diretório `webapps` do contêiner web Apache Tomcat. Ao iniciá-lo, o deploy das do sesame será realizado automaticamente.

O próximo passo é armazenar o arquivo RDF no servidor de triplas RDF do Sesame. Para isso existem diversas opções: interface web, ferramenta de linha de comando e requisições SPARQL. Por ser a mais intuitiva, escolhemos a interface web. Para acessá-la basta acessar a página web através do endereço:

**`http://endereço_onde_o_contêiner_web_esta_rodando/openrdf-workbench/`**

A página inicial pode ser conferida na imagem a seguir:

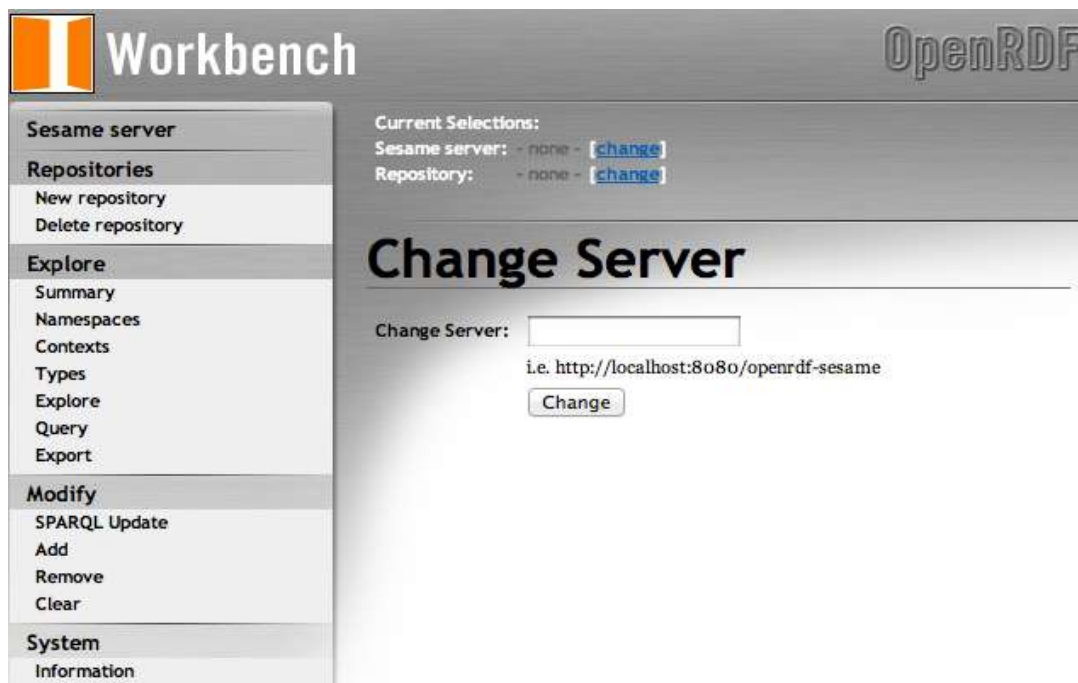


Figura 26 – Página inicial do Sesame Server Fonte: Autores

### 3.2.5.3.2. Hospedando o arquivo RDF

Com o deploy do servidor pronto, o passo seguinte foi realizar o upload do nosso arquivo RDF no Sesame server.

Para hospedar um arquivo RDF o Sesame precisa de um repositório. A imagem a seguir mostra um exemplo de como criá-lo:



Figura 27 – Criação de um repositório Fonte: Autores

Com o repositório criado, utilizamos a interface web para realizar o upload do arquivo rdf no servidor. Para adicionar utilizamos a opção “Modify > Add”, presente no menu de itens na lateral esquerda da página. Existem três opções para fornecer o arquivo RDF: um link de um arquivo hospedado na web, upload do arquivo da máquina local e através de um campo de texto adicionar o conteúdo explícito do arquivo. Utilizamos a segunda opção. A figura a seguir mostra a configuração utilizada para realizar esta etapa:

The screenshot shows the 'Add RDF' interface. On the left, a sidebar menu has 'Modify' selected, with sub-items 'SPARQL Update', 'Add', 'Remove', and 'Clear'. The main area is titled 'Add RDF' and contains the following fields and controls:

- Base URI:** file://
- Context:** <file://>
- Data format:** RDF/XML
- Upload Location:** Two radio buttons. The first is 'Location of the RDF data you wish to upload' (unselected). The second is 'Select the file containing the RDF data you wish to upload' (selected).
- RDF Data URL:** An empty text input field.
- RDF Data File:** A 'Choose File' button followed by the text 'municipios-dados.rdf'.
- RDF Content:** A large, empty text area for pasting content.
- Upload:** A button at the bottom of the form.

Figura 28 – Upload do arquivo RDF Fonte: Autores

### 3.2.5.3.3. Utilização do endpoint SPARQL

Uma vez que o arquivo já está hospedado no Sesame, o endpoint SPARQL já pôde ser testado. Utilizando a opção “Explore > Query” no menu de itens foi possível acessar a página que permite a entrada e execução de uma consulta SPARQL sobre os dados existentes no repositório. As imagens a seguir mostram a página da consulta e o resultado de sua execução:

# Query Repository

Query Language:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://lodkem.ufsc.br/onto/GeoPoliticaBr#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX dados: <http://lodkem.ufsc.br/onto/DadosBr#>
```

Query:

```
SELECT ?nomeMun ?areaMun ?codMun ?temRepass
WHERE {
  ?municipio
    geo:temNomeMun ?nomeMun ;
    geo:temCodIbgeMun ?codMun ;
    dados:temRepasselCMS ?temRepass;
    geo:temAreaMun ?areaMun .
}
```

Limit results:

Include inferred statements

Figura 29 – Página de consulta fornecida pelo Sesame Fonte: Autores

# Query Result (50)

Limit results:  The results shown maybe truncated.

NomeMun	AreaMun	CodMun	TemRepass
<a href="#">"Abdon Batista"^^xsd:string</a>	<a href="#">235.6E0</a>	<a href="#">"4200051"^^xsd:int</a>	<a href="#">2053885.1500000001E0</a>
<a href="#">"Abelardo Luz"^^xsd:string</a>	<a href="#">955.368E0</a>	<a href="#">"4200101"^^xsd:int</a>	<a href="#">9947673.8500000001E0</a>
<a href="#">"Agrolândia"^^xsd:string</a>	<a href="#">207.119E0</a>	<a href="#">"4200200"^^xsd:int</a>	<a href="#">4296712.89E0</a>
<a href="#">"Agronômica"^^xsd:string</a>	<a href="#">135.923E0</a>	<a href="#">"4200309"^^xsd:int</a>	<a href="#">2955466.98E0</a>
<a href="#">"Alfredo Wagner"^^xsd:string</a>	<a href="#">732.277E0</a>	<a href="#">"4200705"^^xsd:int</a>	<a href="#">3210465.1799999997E0</a>
<a href="#">"Alto Bela Vista"^^xsd:string</a>	<a href="#">103.592E0</a>	<a href="#">"4200754"^^xsd:int</a>	<a href="#">2529219.37E0</a>
<a href="#">"Anchieta"^^xsd:string</a>	<a href="#">228.58E0</a>	<a href="#">"4200804"^^xsd:int</a>	<a href="#">2883367.18E0</a>
<a href="#">"Angelina"^^xsd:string</a>	<a href="#">499.947E0</a>	<a href="#">"4200903"^^xsd:int</a>	<a href="#">2507316.17E0</a>
<a href="#">"Anita Garibaldi"^^xsd:string</a>	<a href="#">588.612E0</a>	<a href="#">"4201000"^^xsd:int</a>	<a href="#">6778971.579999999E0</a>

Figura 30 – Resultado de uma consulta sobre os dados do servidor Fonte: Autores

Para facilitar o acesso deste endpoint SPARQL, colocamos um link na aplicação mashup desenvolvida.

#### 3.2.5.3.4. Utilização do Pubby como interface linked data

Como visto no capítulo 2.8.3.2, o Pubby é uma aplicação frontend que fornece uma interface Linked Data para servidores que não possuem essa funcionalidade, como é o caso do servidor que utilizamos. Ele utiliza o endpoint SPARQL fornecido pelo Sesame para buscar os dados requisitados.

A configuração do Pubby é bem simples, consistindo apenas de um arquivo, que pode ser conferido na imagem a seguir:

```
@prefix conf: <http://richard.cyganiak.de/2007/pubby/config.rdf#> .
@prefix geo:<http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#> .
@prefix dados:<http://www.lodkem.ufsc.br/onto/DadosBr#> .
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:<http://www.w3.org/2001/XMLSchema#> .
@prefix owl:<http://www.w3.org/2002/07/owl#> .

<> a conf:Configuration;
    conf:projectName "OpenLinkedSC";
    conf:projectHomepage <http://lodkem.ufsc.br/>;
    conf:webBase <http://177.71.183.112:8080/visualizer/>;

    conf:dataset [
        conf:sparqlEndpoint <http://177.71.183.112:8080/openrdf-|
sesame/repositories/repositorio-final>;
        conf:datasetBase
<http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#>;
        conf:addSameAsStatements "true";
    ];

    conf:labelProperty rdf:type, xsd:string;
    conf:indexResource
<http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#Municipio/Forquilha>;
    .
```

**Figura 31 – Arquivo de configuração do Pubby Fonte: Autores**

O deploy foi realizado no mesmo contêiner web utilizado em outras etapas, o Apache Tomcat. Bastou reiniciar o serviço do contêiner para que a interface do Pubby pudesse ser acessada. Desta forma é possível navegar pelos dados fornecidos pelo servidor Sesame de forma amável ao usuário. Ela pode ser conferida na imagem a seguir:

**Joinville** at OpenLinkedSC  
<http://177.71.183.112:8080/visualizer/Municipio/Joinville>

Property	Value
?:foiDesmembradoDe	▪ São Francisco do Sul ()
?:pertenceEstado	▪ < <a href="http://177.71.183.112:8080/visualizer/Estado/Santa_Catarina">http://177.71.183.112:8080/visualizer/Estado/Santa_Catarina</a> >
?:sameAs	▪ < <a href="http://dbpedia.org/resource/Joinville">http://dbpedia.org/resource/Joinville</a> >
?:temAreaMun	▪ 1130.878E0 ()
?:temArrecadacaoIcms	▪ 8.12732113E8 ()
?:temCatDoencaProfissional	▪ 117 ()
?:temCatTipico	▪ 2822 ()
?:temCatTrajeto	▪ 690 ()
?:temCodIbgeMun	▪ 4209102 ()
?:temDensidadeDemografica	▪ 0.857E0 ()
?:temDistanciaCapital	▪ 1130.878E0 ()
?:temGentilico	▪ Joinvilense ()
?:temIDH1991	▪ 0.779E0 ()
?:temIDH2000	▪ 0.857E0 ()
?:temNomeMun	▪ Joinville ()
?:temObitosAcidenteTrabalho	▪ 7 ()
?:temPontoCentralMun	▪ < <a href="http://177.71.183.112:8080/visualizer/Ponto/137">http://177.71.183.112:8080/visualizer/Ponto/137</a> >
?:temPopulacao2010	▪ 515250 ()
?:temPopulacaoHomens	▪ 255763 ()
?:temPopulacaoMulheres	▪ 259487 ()
?:temRepasselIcms	▪ 2.9446520317999995E8 ()
?:temSemCat	▪ 781 ()
?:type	▪ < <a href="http://lodkem.ufsc.br/onto/GeoPoliticaBr#Municipio">http://lodkem.ufsc.br/onto/GeoPoliticaBr#Municipio</a> >

This page shows information obtained from the SPARQL endpoint at <http://177.71.183.112:8080/openrdf-sesame/repositories/repositorio-final>.  
[As Turtle](#) | [As RDF/XML](#) | [Browse in Disco](#) | [Browse in Tabulator](#) | [Browse in OpenLink Browser](#)

**Figura 32 – Interface disponibilizada pelo Pubby Fonte: Autores**

### 3.2.5.3.5. Utilização da interface RESTful HTTP

Esta abordagem foi necessária para atingirmos um dos objetivos do projeto, que consiste no desenvolvimento de uma aplicação que utilizasse os dados já na forma de dados ligados. Para isso foi preciso encontrar uma forma de obter os dados a partir do servidor para alimentar a aplicação. O próprio Sesame, servidor de triplas RDF fornece uma interface RESTful HTTP, que atende exatamente as nossas necessidades.

Através de uma API Java desenvolvida no mesmo projeto responsável pelo Sesame é possível montar uma requisição HTTP GET, adicionar uma query SPARQL como parâmetro, obtendo como retorno os dados resultantes da execução da query sobre as triplas armazenadas no servidor.

### 3.2.6. Consumo dos dados publicados

Tendo todos os dados reunidos publicados na forma de dados ligados, o passo seguinte foi criar uma aplicação que os utilizasse. Por entender que as questões técnicas do desenvolvimento deste sistema não fazem parte do escopo do projeto, este capítulo tem com intuito abordar de maneira geral as funcionalidades que a aplicação fornece ao usuário.



A aplicação consiste em um mapa, fornecido pela API Javascript desenvolvida pela Google, centralizado no estado de Santa Catarina. Para cada município adicionamos um marcador, uma imagem que o representa, que quando clicado retorna as informações referentes àquela cidade no Sesame Server e os mostra ao usuário através de uma nota tela. O fornecimento dos dados de cada município se dá através da interface RESTful HTTP fornecida pelo Sesame, como explicado no capítulo 4.5.3.4.

Para facilitar a visualização foram adicionados dois tipos de filtro, por municípios e por total da população. O primeiro permite que o usuário escolha quais municípios, por nome, serão mostrados no mapa. Já o filtro por população mostra os municípios filtrados pelo intervalo de população escolhido.

O usuário também pode ter acesso a partir deste sistema tanto ao arquivo rdf, utilizando-o da maneira que desejar, como ao endpoint SPARQL, permitindo-o realizar consultas sobre os dados.

As imagens a seguir mostram a aplicação finalizada.



Figura 32 – Aplicação que consome os dados publicados Fonte: Autores

### Águas Mornas

▾ Geral

Código IBGE: undefined

Desmembrado de: Santo Amaro da Imperatriz

Gentílico: Agumornense

Distância da capital: 39 km

Estado: Santa Catarina

sameAs: [http://dbpedia.org/resource/Águas\\_Mornas](http://dbpedia.org/resource/Águas_Mornas)

▸ Geografia

▸ Economia

▸ População

▸ Acidentes de Trabalho



Figura 33 – Tela de detalhamento para um município Fonte: Autores

## 4. Conclusões

A finalidade deste trabalho foi publicar dados abertos na forma de dados ligados, relacionados ao domínio da geografia e estatística catarinense, mostrando todos os passos realizados durante esse processo. Ao final, desenvolver uma aplicação que utilizasse estes dados em sua forma final. Durante todo o desenvolvimento do trabalho, várias dificuldades foram encontradas, principalmente pelo fato de o tema ser um assunto relativamente novo. O que mais dificultou a execução do projeto foi a busca de dados abertos. A utilização de ferramentas que ainda se encontravam em estágio de desenvolvimento, contendo pouca documentação e apresentando instabilidade, gerando em alguns casos a perda de dados, também aumentou a complexidade do projeto.

### 4.1. Avaliação do Resultado

Durante a primeira etapa do trabalho conseguimos reunir uma quantidade muito boa de informações, sobre todo o conteúdo que foi abordado pelo projeto. Este material foi de extrema importância, pois em momentos de dúvida tínhamos onde buscar o embasamento teórico necessário. Foi neste levantamento que pudemos perceber que o movimento de publicação de dados abertos vem ganhando força nos últimos anos, principalmente no âmbito governamental com a aprovação de leis que incentivam a transparência das entidades governamentais. Também é notável o surgimento de iniciativas de publicação de dados ligados, ainda que não tão forte quando somente dados abertos.

Já a segunda consistiu em reunir os dados geográficos e estatísticos de Santa Catarina. Pode-se dizer que esta foi a parte que mais nos consumiu tempo. Apesar das iniciativas que estão surgindo, como Portal brasileiro de dados abertos e Portal da transparência do Governo Federal, que estão forçando os governos estaduais a tornarem-se cada vez mais transparentes, este processo ainda se dá de forma lenta. Outro fator que colaborou foi a necessidade da verificação de fontes dos dados encontrados, garantido que eles pudessem ser utilizados. Apesar destes contratemplos, conseguimos levantar uma boa quantidade de dados, suficiente para garantia da continuidade do trabalho.

A terceira parte consistiu na transformação dos dados encontrados, transferindo-os para uma base de dados relacional, mapeando-os para o modelo ontológico para em seguida gerar o arquivo rdf final, de acordo com as boas práticas de publicação de dados ligados. Durante

todo este processo utilizamos ferramentas para realizar tais transformações. Enquanto que a maioria já estava consolidada, algumas ainda se encontravam em fase de desenvolvimento, não apresentando uma documentação clara ou gerando erros durante a sua utilização. Em alguns casos houve a perda de dados, gerando retrabalho para completar a atividade. No final, acreditamos que de alguma forma isso foi positivo, pois pudemos testar várias ferramentas, para encontrar aquela que mais se adequava as nossas necessidades.

O quarto passo foi a publicação do arquivo gerado nos padrões de dados ligados. Para isso tivemos que pesquisar ferramentas que possibilitassem tal feito. Dentre as encontradas, foi possível fazer uma comparação, buscando a que mais atendia nossas necessidades. No final, conseguimos atingir três abordagens diferentes de publicação: arquivo estático, servidor de triplas rdf e exposição direta da base de dados. Cada uma possui vantagens e desvantagens, de acordo que a finalidade necessária. Para o nosso caso a segunda abordagem, servidor de triplas rdf, foi o que mais trouxe benefício, pois a ferramenta escolhida possuía várias funcionalidades que nos ajudaram em outras situações, como por exemplo a interface RESTful HTTP utilizada pela aplicação que consome os dados publicados.

O quinto e último esforço baseou-se no desenvolvimento de uma aplicação que consumisse os dados publicados, provando o sucesso das etapas anteriores. Conseguimos criar uma interface que permite ao usuário visualizar as informações de maneira fácil e organizada.

Analisando todo o projeto, concordamos que os objetivos propostos no início foram atingidos, apesar das dificuldades encontradas. Além disso, durante toda a fase de desenvolvimento percebemos oportunidades de melhoria, que serão explicitadas no capítulo seguinte.

## 5. Trabalhos Futuros

Com o final do desenvolvimento do trabalho, mesmo tendo atingido os objetivos propostos, percebemos que ainda existem melhorias que podem ser feitas para enriquecer o projeto. Destre elas, podemos citar:

- Aumentar o escopo dos dados, atingindo todos os municípios do Brasil;
- Captação de mais informações;
- Extensão das ontologias desenvolvidas;
- Publicação dos dados seguindo um fluxo ainda não abordado;
- Ligação com mais datasets;
- Melhorar a usabilidade da aplicação;

# Referência bibliográfica

BERNERS-LEE, Tim. BIZER, Christian. HEATH, Tom. **Linked Data - The Story So Far**. Disponível em <http://tomheath.com/slides/2009-09-london-linked-data-the-story-so-far.pdf>, 2009. Acessado em: 10/10/2011.

BIZER, Christian. HEATH, Tom. **Linked Data: Evolving the Web into a Global Data Space**. Disponível em: <http://linkeddatabook.com/book>, 2011. Acessado em: 10/10/2011.

BERNERS-LEE, Tim. **Linked Data. Design Issues**. Disponível em: <http://www.w3.org/DesignIssues/LinkedData.html>, 2006. Acessado em: 12/12/2011.

BIZER, Christian. HEATH, Tom. AYERS, Danny. RAIMOND, Yves. **Interlinking Open Data on the Web**. Disponível em: <http://sites.wiwi.fu-berlin.de/suhl/bizer/pub/LinkingOpenData.pdf>, 2007. Acessado em: 03/01/2012.

BREITMAN, Karen. **Web Semântica, a Internet do futuro**. Rio de Janeiro, 2005

W3C, **Semantic Web FAQ**. Disponível em: <http://www.w3.org/2001/sw/SW-FAQ#swonbrowser>, 2009. Acessado em: 14/11/2011.

W3C, **Resource Description Framework**. Disponível em <http://www.w3.org/RDF>, 2012. Acessado em: 20/11/2011.

BERNERS-LEE, Tim; CAILLIAU, Robert. **WorldWideWeb: Proposal for a hypertexts Project**. Disponível em: <http://www.w3.org/Proposal.html>, 1990. Acessado em: 05/03/2012.

TAUBERER, Joshua. **What is RDF**. Disponível em: <http://www.xml.com/pub/a/2001/01/24/rdf.html>, 2006. Acessado em: 14/10/2011.

W3C, **Resource Description Framework: Concepts and Abstract Syntax**. Disponível em: <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>, 2004. Acessado em: 14/10/2011.

FRANÇA, Patrícia. **Conceitos, classes e/ou universais: com o que é que se constrói uma ontologia?**. Disponível em: <http://www.linguamatica.com/index.php/linguamatica/article/view/10>, Acessado em: 02/03/2012.

GRUBER, Tom. **What is an Ontology?**. Disponível em: <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>, 1992. Acessado em: 17/11/2011.

BIANCO, Rafael. **Disponibilização de dados abertos utilizando Linked Data: Uma avaliação teórico-prática**. Florianópolis, 2011.

BERNERS-LEE, Tim. **Pre-W3C Web and Internet Background**. Disponível em: <http://www.w3.org/2004/Talks/w3c10-HowItAllStarted/?n=15>, 2004. Acessado em: 05/03/2012.

Open Government Partnership, **Participating Countries – Brazil – Efforts to Date**. Disponível em: <http://www.opengovpartnership.org/countries/brazil>, 2011. Acessado em: 05/03/2012.

Controladoria-Geral da União, **Plano de Ação do Governo Brasileiro - Parceria para Governo Aberto (OGP)**. Disponível em: <http://www.cgu.gov.br/PrevencaodaCorrupcao/AreasAtuacao/CompromissosInternacionais/Arquivos/ogp-brazil-actionplan.pdf>, 2011. Acessado em: 01/04/2012.

BERNERS-LEE, Tim. HENDLER, James. LASSILA, Ora. **The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities**. Disponível em: [http://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American\\_%20Feature%20Article\\_%20The%20Semantic%20Web\\_%20May%202001.pdf](http://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf), 2001. Acessado em: 14/11/2011.

# Apêndices

Código fonte da aplicação desenvolvida

```
package br.lodkem.ufsc.geoopenscapi.model.municipio;
```

```
import javax.xml.bind.annotation.XmlRootElement;
```

```
@XmlRootElement
```

```
public class Municipio {
```

```
    private Integer id;
```

```
    private String nome;
```

```
    private String uf;
```

```
    private Double latitude;
```

```
    private Double longitude;
```

```
    private String area;
```

```
    private Double altitude;
```

```
    private String nomeEstado;
```

```
    private String areaEstado;
```

```
    private String sameAs;
```

```
    private String populacao2010;
```



```
private String populacaoHomens;

private String populacaoMulheres;

private String repasseICMS;

private String acidenteSemCat;

private String acidenteObitos;

private String acidenteCatDoenca;

private String acidenteCatTrajeto;

private String acidenteCatTipico;

private String desmembradoDe;

private String gentilico;

private String densidadeDemografica;

private String distanciaCapital;

private String idh1991;

private String idh2000;

private String icmsArracadacao;

public Municipio() {
}

public Integer getId() {
```

```
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getUf() {
        return uf;
    }

    public void setUf(String uf) {
        this.uf = uf;
    }

    public Double getLatitude() {
        return latitude;
    }

    public void setLatitude(Double latitude) {
        this.latitude = latitude;
    }

    public Double getLongitude() {
        return longitude;
    }
}
```

```
public void setLongitude(Double longitude) {
    this.longitude = longitude;
}

public String getArea() {
    return area;
}

public void setArea(String area) {
    this.area = area;
}

public Double getAltitude() {
    return altitude;
}

public void setAltitude(Double altitude) {
    this.altitude = altitude;
}

public String getNomeEstado() {
    return nomeEstado;
}

public void setNomeEstado(String nomeEstado) {
    this.nomeEstado = nomeEstado;
}

public String getSameAs() {
    return sameAs;
}

public void setSameAs(String sameAs) {
```

```
        this.sameAs = sameAs;
    }

    public String getRepasseICMS() {
        return repasseICMS;
    }

    public void setRepasseICMS(String repasseICMS) {
        this.repasseICMS = repasseICMS;
    }

    public String getAreaEstado() {
        return areaEstado;
    }

    public void setAreaEstado(String areaEstado) {
        this.areaEstado = areaEstado;
    }

    public String getPopulacao2010() {
        return populacao2010;
    }

    public void setPopulacao2010(String populacao2010) {
        this.populacao2010 = populacao2010;
    }

    public String getPopulacaoHomens() {
        return populacaoHomens;
    }

    public void setPopulacaoHomens(String populacaoHomens) {
        this.populacaoHomens = populacaoHomens;
    }
}
```

```
public String getPopulacaoMulheres() {
    return populacaoMulheres;
}

public void setPopulacaoMulheres(String populacaoMulheres) {
    this.populacaoMulheres = populacaoMulheres;
}

public String getAcidenteSemCat() {
    return acidenteSemCat;
}

public void setAcidenteSemCat(String acidenteSemCat) {
    this.acidenteSemCat = acidenteSemCat;
}

public String getAcidenteObitos() {
    return acidenteObitos;
}

public void setAcidenteObitos(String acidenteObitos) {
    this.acidenteObitos = acidenteObitos;
}

public String getAcidenteCatDoenca() {
    return acidenteCatDoenca;
}

public void setAcidenteCatDoenca(String acidenteCatDoenca) {
    this.acidenteCatDoenca = acidenteCatDoenca;
}

public String getAcidenteCatTrajeto() {
```

```

        return acidenteCatTrajeto;
    }

    public void setAcidenteCatTrajeto(String acidenteCatTrajeto) {
        this.acidenteCatTrajeto = acidenteCatTrajeto;
    }

    public String getAcidenteCatTipico() {
        return acidenteCatTipico;
    }

    public void setAcidenteCatTipico(String acidenteCatTipico) {
        this.acidenteCatTipico = acidenteCatTipico;
    }

    public String getDesmembradoDe() {
        return desmembradoDe;
    }

    public void setDesmembradoDe(String desmembradoDe) {
        this.desmembradoDe = desmembradoDe;
    }

    public String getGentilico() {
        return gentilico;
    }

    public void setGentilico(String gentilico) {
        this.gentilico = gentilico;
    }

    public String getDensidadeDemografica() {
        return densidadeDemografica;
    }
}

```

```
public void setDensidadeDemografica(String densidadeDemografica) {  
    this.densidadeDemografica = densidadeDemografica;  
}
```

```
public String getDistanciaCapital() {  
    return distanciaCapital;  
}
```

```
public void setDistanciaCapital(String distanciaCapital) {  
    this.distanciaCapital = distanciaCapital;  
}
```

```
public String getIdh1991() {  
    return idh1991;  
}
```

```
public void setIdh1991(String idh1991) {  
    this.idh1991 = idh1991;  
}
```

```
public String getIdh2000() {  
    return idh2000;  
}
```

```
public void setIdh2000(String idh2000) {  
    this.idh2000 = idh2000;  
}
```

```
public String getIcmsArracadacao() {  
    return icmsArracadacao;  
}
```

```
public void setIcmsArracadacao(String icmsArracadacao) {
```

```

        this.icmsArracadacao = icmsArracadacao;
    }

}

package br.lodkem.ufsc.geoopensscore.service.municipio;

import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

import br.lodkem.ufsc.geoopensscapi.model.municipio.Municipio;
import br.lodkem.ufsc.geoopensscore.service.municipio.util.MunicipioSQL;

import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;

public class MunicipioService {

    private static MunicipioService municipioService;

    private NumberFormat format = NumberFormat.getNumberInstance(new Locale("pt",
"br"));

    private String ENDPOINT_URL = "";

    private MunicipioService(String url) {
        this.ENDPOINT_URL = url;
    }
}

```



```

public static MunicipioService getMunicipioServiceInstance(String url) {
    if (municipioService == null) {
        municipioService = new MunicipioService(url);
    }
    return municipioService;
}

public List<Municipio> listMunicipios(int minPop, int maxPop) {
    List<Municipio> municipios = new ArrayList<Municipio>();
    try {
        QueryExecution qe =
QueryExecutionFactory.sparqlService(ENDPOINT_URL, MunicipioSQL.getListQuery(minPop,
maxPop));

        ResultSet rs = qe.execSelect();
        while (rs.hasNext()) {
            QuerySolution solution = rs.next();
            municipios.add(getMunicipioList(solution));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return municipios;
}

public Municipio findMunicipio(int idMunicipio) {
    Municipio municipio = null;
    try {
        QueryExecution qe =
QueryExecutionFactory.sparqlService(ENDPOINT_URL,
MunicipioSQL.getFindQuery(idMunicipio));
        ResultSet rs = qe.execSelect();
        if (rs.hasNext()) {
            QuerySolution solution = rs.next();
            municipio = getMunicipioFind(solution);
        }
    }
}

```

```

        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return municipio;
}

```

```

private Municipio getMunicipioList(QuerySolution solution) {
    Municipio municipio = new Municipio();
    municipio.setId(solution.getLiteral("codMun").getInt());
    municipio.setNome(solution.getLiteral("nomeMun").getString());
    municipio.setLatitude(solution.getLiteral("lat").getDouble());
    municipio.setLongitude(solution.getLiteral("lon").getDouble());
    return municipio;
}

```

```

private Municipio getMunicipioFind(QuerySolution solution) {

    Municipio municipio = new Municipio();
    municipio.setNome(solution.getLiteral("nomeMun").getString());
    municipio.setLatitude(solution.getLiteral("lat").getDouble());
    municipio.setLongitude(solution.getLiteral("lon").getDouble());
    municipio.setArea(formatValue(solution.getLiteral("areaMun").getString()));
    municipio.setNomeEstado(solution.getLiteral("nomeEst").getString());
    municipio.setAreaEstado(formatValue(solution.getLiteral("areaEst").getString()));
    municipio.setSameAs(solution.getResource("same").getURI());

    municipio.setPopulacao2010(formatValue(solution.getLiteral("pop2010").getString()));

    municipio.setPopulacaoHomens(formatValue(solution.getLiteral("popH").getString()));

    municipio.setPopulacaoMulheres(formatValue(solution.getLiteral("popM").getString()));
    municipio.setRepasselCMS(formatValue(solution.getLiteral("icms").getString()));
}

```

```

        municipio.setAcidenteSemCat(solution.getLiteral("acSemCat").getString());
        municipio.setAcidenteObitos(solution.getLiteral("acObitos").getString());
        municipio.setAcidenteCatDoenca(solution.getLiteral("acCatDoenca").getString());
        municipio.setAcidenteCatTrajeto(solution.getLiteral("acCatTrajeto").getString());
        municipio.setAcidenteCatTipico(solution.getLiteral("acCatTipico").getString());

        municipio.setDesmembradoDe(solution.getLiteral("desmembradoDe").getString());

        municipio.setDistanciaCapital(formatValue(solution.getLiteral("distanciaCapital").getStrin
g()));

        municipio.setDensidadeDemografica(formatValue(solution.getLiteral("densidadeDemogr
afica").getString()));
        municipio.setGentilico(solution.getLiteral("gentilico").getString());
        municipio.setIdh1991(formatValue(solution.getLiteral("idh1991").getString()));
        municipio.setIdh2000(formatValue(solution.getLiteral("idh2000").getString()));

        municipio.setIcmsArrecadacao(formatValue(solution.getLiteral("icmsArrecadacao").getSt
ring()));

        return municipio;
    }

    private String formatValue(String value) {
        String retorno = null;
        try {
            retorno = format.format(new Double(value));
        } catch (Exception e) {
        }
        return retorno;
    }

    public NumberFormat getFormat() {
        return format;
    }

```

```

    }

}

package br.lodkem.ufsc.geopensscore.service.municipio.util;

public class MunicipioSQL {

    private static StringBuilder getQueryPrefix() {
        StringBuilder prefix = new StringBuilder();
        prefix.append("PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ");
        prefix.append("PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> ");
        prefix.append("PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> ");
        prefix.append("PREFIX owl: <http://www.w3.org/2002/07/owl#> ");
        prefix.append("PREFIX geo:<http://lodkem.ufsc.br/onto/GeoPoliticaBr#> ");
        prefix.append("PREFIX dados:<http://lodkem.ufsc.br/onto/DadosBr#> ");

        return prefix;
    }

    public static void main(String[] args) {
        //System.out.println(MunicipioSQL.getFindQuery(4200051));
        System.out.println(MunicipioSQL.getListQuery(462642, 515250));
    }

    public static String getListQuery(int minPop, int maxPop) {
        StringBuilder query = getQueryPrefix();
        query.append(" SELECT ?codMun ?nomeMun ?lat ?lon ");
        query.append(" WHERE { ");
        query.append(" ?municipio geo:temPontoCentralMun ?ponto ; ");
        query.append(" geo:temNomeMun ?nomeMun ; ");
        query.append(" geo:temCodIbgeMun ?codMun ; ");
        query.append(" dados:temPopulacao2010 ?pop2010 ; . ");
    }
}

```

```

        query.append(" ?ponto          geo:lat ?lat ; ");
        query.append(" geo:lon ?lon . ");
        query.append(" FILTER (?pop2010 >= " + minPop + " && ?pop2010 <= " +
maxPop + " ) . ");
        query.append(" } ORDER BY ?nomeMun ");

        return query.toString();
    }

    public static String getFindQuery(int idMunicipio) {
        StringBuilder query = getQueryPrefix();

        query.append(" SELECT ?codMun ?nomeMun ?areaMun ?same ?pop2010
?popH ?popM ?icms ?lat ?lon ?nomeEst ?areaEst ");
        query.append(" ?acSemCat ?acObitos ?acCatDoenca ?acCatTrajeto
?acCatTipico ?desmembradoDe ?densidadeDemografica ");
        query.append(" ?distanciaCapital ?gentilico ?idh1991 ?idh2000
?icmsArrecadacao ");
        query.append(" WHERE { ");
        query.append(" ?municipio   geo:temPontoCentralMun ?ponto ; ");
        query.append(" geo:pertenceEstado ?estado ; ");
        query.append(" geo:temNomeMun ?nomeMun ; ");
        query.append(" geo:temCodIbgeMun ?codMun ; ");
        query.append(" geo:temAreaMun ?areaMun ; ");
        query.append(" dados:temPopulacao2010 ?pop2010 ; ");
        query.append(" dados:temPopulacaoHomens ?popH ; ");
        query.append(" dados:temPopulacaoMulheres ?popM ; ");
        query.append(" dados:temSemCat ?acSemCat ; ");
        query.append(" dados:temObitosAcidenteTrabalho ?acObitos ; ");
        query.append(" dados:temCatDoencaProfissional ?acCatDoenca ; ");
        query.append(" dados:temCatTrajeto ?acCatTrajeto ; ");
        query.append(" dados:temCatTipico ?acCatTipico ; ");
        query.append(" dados:foiDesmembradoDe ?desmembradoDe ; ");
        query.append(" dados:temDensidadeDemografica ?densidadeDemografica ; ");

```

```

        query.append(" dados:temDistanciaCapital ?distanciaCapital ; ");
        query.append(" dados:temGentilico ?gentilico ; ");
        query.append(" dados:temIDH1991 ?idh1991 ; ");
        query.append(" dados:temIDH2000 ?idh2000 ; ");
        query.append(" dados:temArrecadacaoICMS ?icmsArrecadacao ; ");
        query.append(" dados:temRepasseICMS ?icms . ");
        query.append(" OPTIONAL { ?municipio owl:sameAs ?same } . ");
        query.append(" ?ponto          geo:lat ?lat ; ");
        query.append(" geo:lon ?lon . ");
        query.append(" ?estado      geo:temNomeEst ?nomeEst; geo:temAreaEst
?areaEst . ");
        query.append(" FILTER          (?codMun = " + idMunicipio + " . ");
        query.append(" } ORDER BY ?nomeMun ");

        return query.toString();
    }
}

```

```

package br.lodkem.ufsc.geoopenscweb.application;

```

```

import javax.ws.rs.ApplicationPath;

```

```

import com.sun.jersey.api.core.PackagesResourceConfig;

```

```

@ApplicationPath("/resources")

```

```

public class Geoopensc extends PackagesResourceConfig {

```

```

    public Geoopensc() {
        super("br.lodkem.ufsc.geoopenscweb.resources");
    }
}

```

```

}

```

```

package br.lodkem.ufsc.geoopenscweb.resources;

```

```

import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;

import br.lodkem.ufsc.geoopenscapi.model.municipio.Municipio;
import br.lodkem.ufsc.geoopensscore.service.municipio.MunicipioService;

@Path("/municipio")
public class MunicipioResource {

    @GET
    @Path("/list")
    @Produces("application/json;charset=UTF-8")
    public List<Municipio> listMunicipios(@QueryParam(value = "minPop") int limiteMinimo,
    @QueryParam(value = "maxPop") int limiteMaximo) {
        System.out.println(limiteMinimo + " " + limiteMaximo);
        return
MunicipioService.getMunicipioServiceInstance(getUrl()).listMunicipios(limiteMinimo,
limiteMaximo);
    }

    @GET
    @Path("/find")
    @Produces("application/json;charset=UTF-8")
    public Municipio findMunicipio(@QueryParam(value = "id") int idMunicipio){
        return
MunicipioService.getMunicipioServiceInstance(getUrl()).findMunicipio(idMunicipio);
    }

    private static String getUrl(){

```

```

        return Resources.getResourcesInstance().getPropertie("endpoint_url");
    }
}

package br.lodkem.ufsc.geoopenscweb.resources;

import java.io.IOException;
import java.util.Properties;

public class Resources {

    private static Resources resources;

    private Resources() {
    }

    public static Resources getResourcesInstance() {
        if (resources == null) {
            resources = new Resources();
        }
        return resources;
    }

    public String getPropertie(String token) {
        String propertie = null;
        Properties properties = new Properties();
        try {

            properties.load(Thread.currentThread().getContextClassLoader().getResource("geopen
sc.properties").openStream());

            propertie = properties.getProperty(token);
            System.out.println(propertie);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```



```
    }  
    return propertie;  
  }  
}
```

# Disponibilização na web de dados geográficos e estatísticos do Brasil na forma de dados lincados

Guilherme Nandi Tiscoski<sup>1</sup>, Jorge Renato Bornhausen de Sá<sup>1</sup>

<sup>1</sup>Departamento de Informática e Estatística - INE  
Universidade Federal de Santa Catarina (UFSC)  
Caixa Postal 476 – 88.040-900 – Florianópolis, SC – Brasil

guilhermetk@inf.ufsc.br, jorgesasa@inf.ufsc.br

**Abstract.** *The semantic web has as one of its main goals the integration of diverse data sources to enrich the information available. One of the pillars supporting this concept is called Linked Data, a set of good practice to connect and publish structured data on the web. The adoption of these practices has led to the creation of global data set, in which information from various sources and different domains are related. The aim of this paper is the presentation of the concepts applied to the publication of open data in the form of Linked Data.*

**Resumo.** *A web semântica tem como um de seus objetivos principais a integração entre diversas fontes de dados, visando enriquecer as informações disponíveis. Um dos pilares que sustenta este conceito é denominado Linked Data, um conjunto de boas práticas para conectar e publicar dados estruturados na web. A adoção destas boas práticas tem conduzido à criação de conjunto de dados global, no qual informações de diversas fontes e domínios diferentes são relacionadas. O objetivo deste artigo é a apresentação dos conceitos aplicados para a publicação de dados abertos na forma de dados lincados.*

## 1. Introdução

Dados abertos (Open Data, originalmente) são uma vertente da tecnologia da informação que vem crescendo muito na última década, principalmente nos últimos cinco anos. Seu conceito se baseia na ideia de que alguns tipos de informação devem ser disponibilizados gratuitamente, de forma que qualquer pessoa ou instituição possa ter acesso e utilizar da forma que bem entender, sem restrições de direitos autorais ou qualquer mecanismo de controle. O objetivo desse movimento é criar uma cultura coletiva de construção, agrupamento e publicação de dados.

O conceito de web semântica surgiu ao público em 2001 com a publicação de um artigo na revista Scientific America com autoria de Tim Berners-Lee, James Hendler e Ora Lassila. Seus principais objetivos são: definição de padrões de formatação para a integração e combinação de dados provenientes de diversas fontes; atribuição de significado, mostrando como os dados se relacionam com objetos do mundo real. Portanto, se um dos objetivos da web semântica é a integração entre

diversas fontes de dados, visando enriquecer as informações disponíveis, pode se dizer que Linked Data é um dos pilares que a sustenta.

O termo Linked Data representa um conjunto de boas práticas para conectar e publicar dados estruturados na Web (BERNERS-LEE et al, 2009). A adoção dessas boas práticas tem conduzido à criação de um data space global, no qual informações de diversas fontes e domínios diferentes são relacionadas.

## **2. Dados abertos**

Dados Abertos é a ideia de que certos tipos de dados devem estar disponíveis gratuitamente para que todos possam utilizar e republicar, sem restrições de copyright, patentes ou outros mecanismos de controle.

## **3. Web Semântica**

A principal idéia da web semântica é estender a web atual, mudando foco de documentos para dados, e superar as limitações que o modelo de hoje possui. Ela possui dois objetivos principais: o primeiro é definição de padrões de formatação para a integração e combinação de dados provenientes de diversas fontes: o segundo é a atribuição de significado, mostrando como os dados se relacionam com objetos do mundo real. Esses dois pontos em conjunto permitem que não só pessoas, mas também máquinas possam navegar através de bases de dados, que possuem um relacionamento entre si, estabelecido pelo significado dos seus conteúdos.

As tecnologias relacionadas com a Web Semântica atuam no plano de fundo das aplicações ao invés de causarem um impacto visual durante a navegação. Suas aplicações resultam em uma melhor experiência para o usuário que a utiliza. Já existem alguns projetos que se beneficiam com essas práticas, como o portal de finanças do Yahoo.

Web semântica é uma peça chave para a próxima geração de sistemas de informações, onde as informações recebem um significado bem definido, permitindo que pessoas e programas trabalhem em cooperação.

## **3. Linked Data**

Ao analisar a Web atual, é fácil perceber que existe uma relação entre os documentos e páginas visitadas. Essa relação é criada através da inserção de *Hyperlinks*, que direcionam os usuários às páginas ou documentos desejados. Porém, embora haja a relação entre os documentos, não existe uma relação entre as informações disponíveis. Portanto, se um dos objetivos da web semântica é a integração entre diversas fontes de dados, visando enriquecer as informações disponíveis, pode se dizer que Linked Data é um dos pilares que a sustenta.

O termo Linked Data representa um conjunto de boas práticas para conectar e publicar dados estruturados na Web (BERNERS-LEE et al, 2009). A adoção dessas boas práticas tem conduzido à criação de um data space global, no qual informações de diversas fontes e domínios diferentes são relacionadas.

A expansão de Linked Data abre portas para inúmeras novas aplicações e serviços. Já existem navegadores e motores de buscas específicos para esse padrão de dados. O tipo de aplicação mais beneficiado pela expansão da adoção de linked data são as aplicações de mashup. Diferentemente de aplicações mashups já existentes, que trabalham em cima de um conjunto de dados limitado, as aplicações que fizerem uso de dados ligados não terão um limite visível, uma vez que estarão se conectando e consumindo dados de um data space global, permitindo sempre mostrar novas informações à medida que elas são publicadas na web.

## 4. Publicação

### 4.1. Obtenção dos dados

Como fontes de dados priorizamos aquelas de origem governamental, pois garantem a disponibilização de dados atuais de forma aberta, facilitando a adoção dos princípios de disponibilização de dados abertos. Após finalizarmos as buscas chegamos ao seguinte conjunto de dados:

**Tabela 1. Dados abertos obtidos para serem publicados**

Dado	Formato	Fonte
Divisão territorial	XLS	IBGE
Censo 2010	XLS	IBGE
Acidentes de trabalho	XLS	Ministério da previdência
Arrecadação de ICMS	XLS	Secretaria de Estado da Fazenda de Santa Catarina
Repasse de ICMS	XLS	Secretaria de Estado da Fazenda de Santa Catarina
Latitude, Longitude, Área e Altitude	DBF	Departamento de Informática do SUS
Densidade, Distancia a capita, IDH 1991, IDH 2000, Gentílico, Desmembrado de	XLS	Secretaria de Estado da Fazenda de Santa Catarina

Os arquivos foram disponibilizados em vários formatos, principalmente CSV e XLS. Em virtude desse motivo tivemos que utilizar uma ferramenta ETL, neste caso o Kettle, da suite Pentaho, que nos permitiu fazer a integração de dados, tratando-os quando necessário, para em seguida inseri-los na base de dados. Para armazenar os dados na forma relacional utilizamos o banco de dados MySQL.

### 4.2. Relacionamento com outros datasets

Com todos os dados desejados já disponíveis, resolvemos identificar data sets com dados relevantes para realizar a ligação entre eles, seguindo os princípios de linked data. Escolhemos a DBpedia como data set ideal para essa ligação, pois, como observado durante a fundamentação teórica, sua ontologia possui uma quantidade enorme de informações sobre municípios.

Como a relação entre recursos ocorre através da propriedade owl:sameAs, fez-se necessária a obtenção das URIs dos municípios catarinenses presentes na wikipedia. Para esse fim, utilizamos o endpoint SPARQL da dbpedia e realizamos uma consulta que retornasse a URI dos municípios catarinenses junto de seus nomes, como visto na figura abaixo.

```

Query Text
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?uri ?label WHERE {
?uri rdf:type dbo:Place .
?uri dbo:isPartOf <http://dbpedia.org/resource/Santa_Catarina_&28state&29> .
OPTIONAL {?uri rdfs:label ?label . FILTER (lang(?label) = "pt") } .
} ORDER BY ?uri

```

**Figura 1 – Query SPARQL para obter-se a lista de URIs, atrelada ao nome, dos municípios catarinenses**

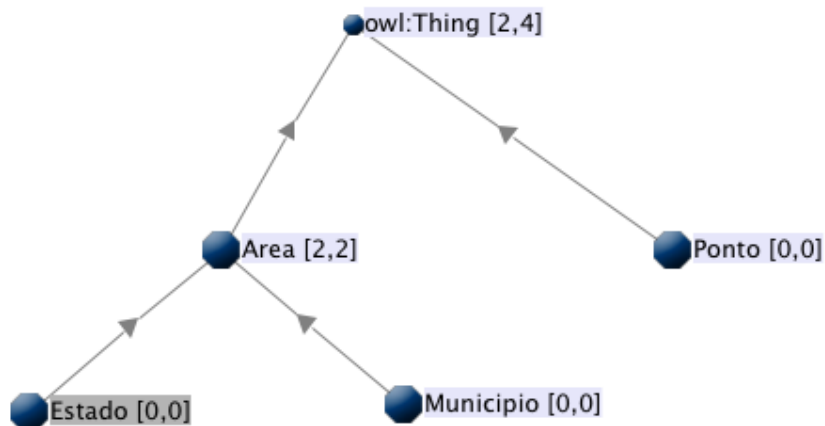
O retorno dessa consulta foi uma lista com a URI de 263 municípios catarinenses, dos 293 existentes. Para armazenar as URIs no banco, relacionando-as com as entidades já existentes, foi utilizada novamente a ferramenta Kettle e criada uma tabela adicional para sameAs.

uri	label
http://dbpedia.org/resource/%C3%81gua_Doce	"Água Doce"@pt
http://dbpedia.org/resource/%C3%81guas_Frias	"Águas Frias (Santa Catarina)"@pt
http://dbpedia.org/resource/%C3%81guas_Mornas	"Águas Mornas"@pt

**Figura 2 – Parto do resultado da pesquisa, no formato HTML**

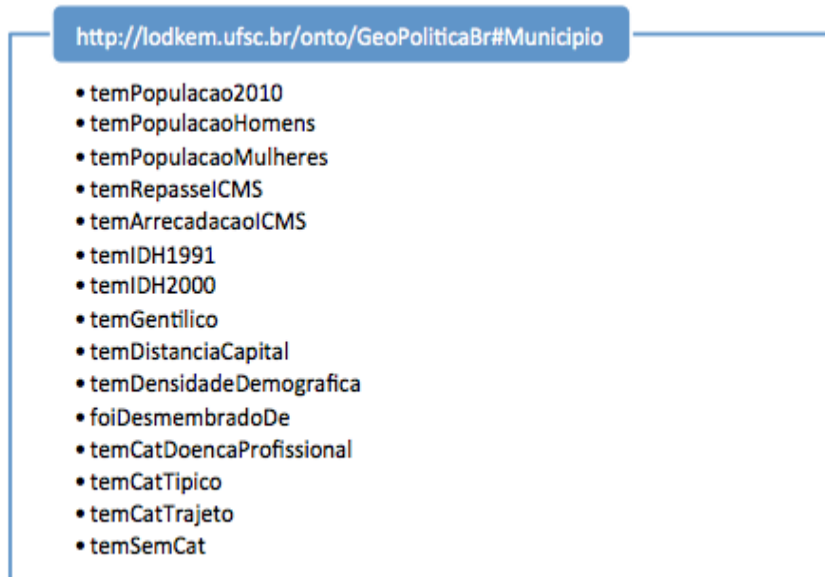
### 4.3. Geração das ontologias

A primeira ontologia gerada, denominada GeoPoliticaBr, representa a divisão territorial do brasil para estados e municípios. Ela mapeia algumas características como, latitude, longitude, area. Para este caso, usamos uma abordagem mais genérica, buscando a criação de um modelo que pudesse ser utilizado em outros trabalhos que também precisam representar estes conceitos. A figura 3 mostra a estrutura de classes da ontologia GeoPoliticaBR.



**Figura 3 – Ontologia GeoPoliticaBR**

Para representar os dados específicos do nosso projeto, foi preciso desenvolver uma nova ontologia, já que a GeoPolíticaBr foi pensada para ser reutilizada em outros trabalhos de modo que não aborda todo o nosso conjunto de dados. Criamos então a DadosBr, uma ontologia que representa somente as propriedades de um município específicas ao nosso trabalho. Tais propriedades estão representadas na figura 4.



**Figura 4 – Propriedades da ontologia DadosBR**

#### 4.4. Geração do arquivo RDF

Tendo todos os dados já reunidos em um banco de dados, o passo seguinte foi realizar o mapeamento entre o modelo relacional e o modelo ontológico, resultando em um arquivo RDF. Para isso utilizamos a ferramenta **dump-rdf**, da plataforma D2RQ.

Um pré-requisito para a utilização desta ferramenta é a existência de um arquivo que faça o mapeamento entre a base de dados relacional e as ontologias. Para

isso utilizamos uma outra ferramenta, também da plataforma D2RQ, chamada **generate-mapping**. Ela nos permitiu a geração de um arquivo de mapeamento default para nossa base de dados. Com base neste arquivo fizemos algumas alterações necessárias para atender as especificidades do nosso projeto. As principais foram :

- adicionar os namespaces das ontologias geradas por nós
- adicionar url para as classes das ontologias
- adicionar o mapeamento sameAs, responsável pela ligação com outro dataset

O resultado deste passo foi a geração do arquivo rdf com os dados da base relacional traduzido para o formato RDF/XML. Uma parte deste arquivo pode ser conferida na figura 5:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:map="file:///Users/guilhermetiscoski/Documents/d2rq-0.8/d2r-mappings/testeSaida.ttl#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:geo="http://lodkem.ufsc.br/onto/GeoPoliticaBr#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:db="file:///Users/guilhermetiscoski/Documents/d2rq-0.8/municipios-sameAs10.rdf"
  xmlns:dados="http://lodkem.ufsc.br/onto/DadosBr#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:vocab="http://www.lodkem.ufsc.br/vocab/resource/" >
  <rdf:Description rdf:about="http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#Municipio/Rio_dos_Cedros">
    <geo:pertenceEstado rdf:resource="http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#Estado/Santa_Catarina"/>
    <geo:temPontoCentralMun rdf:resource="http://www.lodkem.ufsc.br/onto/GeoPoliticaBr#Ponto/216"/>
    <dados:temPopulacaoHomens rdf:datatype="http://www.w3.org/2001/XMLSchema#int">5267</dados:temPopulacaoHomens>
    <dados:temPopulacaoMulheres rdf:datatype="http://www.w3.org/2001/XMLSchema#int">5013</dados:temPopulacaoMulheres>
    <dados:temPopulacao2010 rdf:datatype="http://www.w3.org/2001/XMLSchema#int">10200</dados:temPopulacao2010>
    <geo:temNomeMun rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Rio dos Cedros</geo:temNomeMun>
    <dados:temIDH2000 rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.817E0</dados:temIDH2000>
    <dados:temDistanciaCapital rdf:datatype="http://www.w3.org/2001/XMLSchema#double">555.654E0</dados:temDistanciaCapital>
    <dados:temGentilico rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Riocedrense</dados:temGentilico>
    <dados:temDensidadeDemografica rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.817E0</dados:temDensidadeDemografica>
    <dados:temIDH1991 rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.738E0</dados:temIDH1991>
    <geo:temAreaMun rdf:datatype="http://www.w3.org/2001/XMLSchema#double">555.654E0</geo:temAreaMun>
    <geo:temCodIbgeMun rdf:datatype="http://www.w3.org/2001/XMLSchema#int">4214706</geo:temCodIbgeMun>
    <dados:foiDesmembradoDe rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Timbó</dados:foiDesmembradoDe>
    <rdf:type rdf:resource="http://lodkem.ufsc.br/onto/GeoPoliticaBr#Municipio"/>
    <dados:temCatTrajeto rdf:datatype="http://www.w3.org/2001/XMLSchema#int">11</dados:temCatTrajeto>
    <dados:temSemCat rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</dados:temSemCat>
    <dados:temObitosAcidenteTrabalho rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</dados:temObitosAcidenteTrabalho>
    <dados:temCatDoencaProfissional rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</dados:temCatDoencaProfissional>
    <dados:temCatTipico rdf:datatype="http://www.w3.org/2001/XMLSchema#int">47</dados:temCatTipico>
    <dados:temArrecadacaoICMS rdf:datatype="http://www.w3.org/2001/XMLSchema#double">5970812.94E0</dados:temArrecadacaoICMS>
    <dados:temRepassoICMS rdf:datatype="http://www.w3.org/2001/XMLSchema#double">5122010.18E0</dados:temRepassoICMS>
    <owl:sameAs rdf:resource="http://dbpedia.org/resource/Rio_dos_Cedros"/>
  </rdf:Description>
```

Figura 5 – Parte do arquivo RDF gerado

## 4.5. Publicação dos dados

### 4.5.1. Publicação do arquivo estático

Gerar arquivos rdf estáticos e disponibilizá-los em um servidor web é a maneira mais simples e rápida de publicar dados ligados. Seguindo uma boa prática de publicação de arquivos rdf estáticos, utilizamos o formato de serialização RDF/XML, pois é o mais suportado em ferramentas que consomem dados ligados.

Tendo o arquivo rdf gerado com sucesso, como explicado no capítulo 4.4, bastou somente publicá-lo em um servidor web. Por ser somente um arquivo, escolhemos disponibilizá-lo através do Apache Tomcat, um contêiner web para aplicações que também fornece um serviço de servidor HTTP.

#### **4.5.2. Publicação dos dados a partir de um wrapper RDB2RDF**

Através do D2R Server, outro componente da plataforma D2RQ, foi possível disponibilizar todo o conteúdo da nossa base de dados, sem precisar gerar o arquivo rdf.

O D2R Server aproveita o mesmo arquivo de mapeamento utilizado na geração do arquivo rdf - capítulo 4.4 - porém ao invés da saída ser um registro rdf estático o serviço fornece uma interface amigável ao usuário. Ela consiste em páginas web pré-formatadas, que são preenchidas com os dados dinamicamente traduzidos do formato relacionar para triplas rdf convertidas em HTML para facilitar a visualização.

#### **4.5.3. Publicação dos dados a partir de um servidor RDF**

Para esta etapa, foi preciso analisar dentre as opções possíveis, qual servidor atenderia melhor as necessidades do nosso projeto. Depois de uma pesquisa, chegamos a dois candidatos: Virtuoso Universal Server e Sesame.

O Virtuoso Universal Server é uma ferramenta muito robusta, que atende aos mais diversos casos. Porém, toda essa quantidade de serviços oferecidos apresenta uma complexidade maior de implementação, tornando-se inviável para o nosso projeto.

Já o Sesame é um framework que além de outras funcionalidades, fornece um servidor para armazenar triplas RDF. Sua complexidade de implementação é baixa, pois roda em um contêiner web como o Apache Tomcat, que já vinha sendo usado no projeto. Apesar de não oferecer uma interface Linked Data, é possível contornar esta problema utilizando o Pubby, uma ferramenta que fornece uma interface Linked data para servidores que não a possuam.

Por atender as nossas necessidades alido a uma baixa complexidade de implementação, escolhemos o Sesame para ser nosso servidor RDF.

## **5. Consumo dos dados**

Tendo todos os dados reunidos publicados na forma de dados lincados, o passo seguinte foi criar uma aplicação que os utilizasse.

A aplicação consiste em um mapa, fornecido pela API Javascript desenvolvida pela Google, centralizado no estado de Santa Catarina. Para cada município adicionamos um marcador, uma imagem que o representa, que quando clicado retorna as informações referentes àquela cidade no Sesame Server e os mostra ao usuário através de uma nova tela.

Para facilitar a visualização foram adicionados dois tipos de filtro, por municípios e por total da população. O primeiro permite que o usuário escolha quais municípios, por nome, serão mostrados no mapa. Já o filtro por população mostra os municípios filtrados pelo intervalo de população escolhido.

Uma parte da aplicação pode ser visualizada nas figuras 6 e 7.



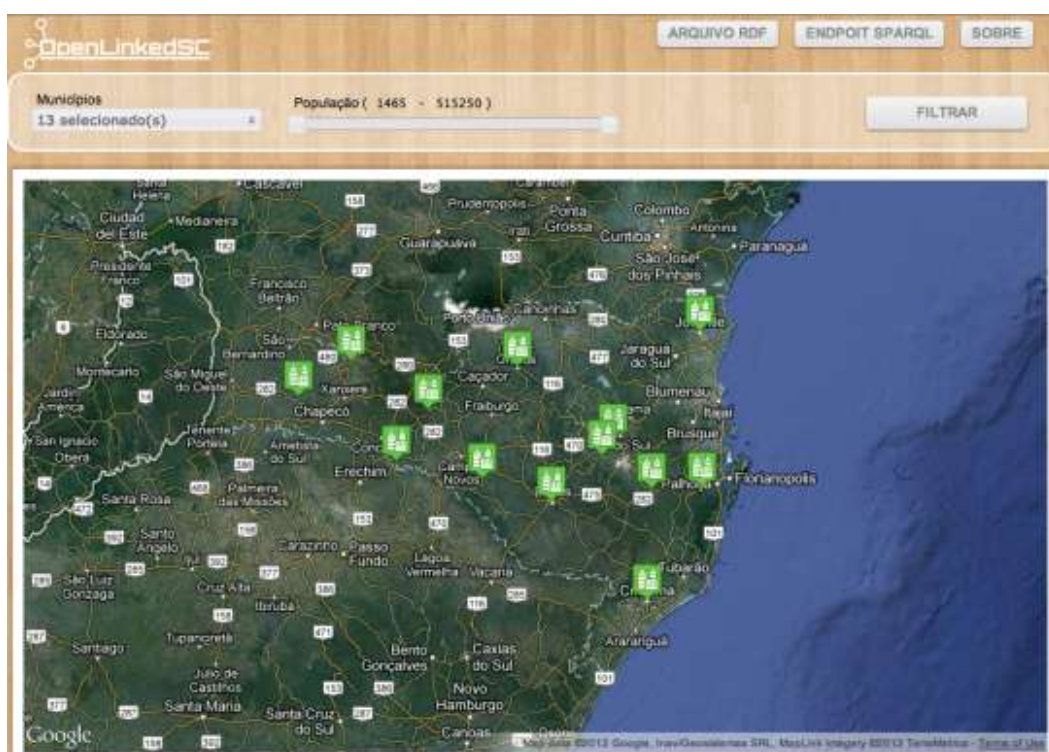


Figura 6 – Aplicação que consome os dados publicados



Figura 7 – Detalhamento de um município selecionado

## 6. Conclusões

A finalidade deste trabalho foi publicar dados abertos na forma de dados ligados, relacionados ao domínio da geografia e estatística catarinense, mostrando todos os passos realizados durante esse processo. Ao final, desenvolver uma aplicação que utilizasse estes dados em sua forma final.

Durante todo o desenvolvimento do trabalho, várias dificuldades foram encontradas, principalmente pelo fato de o tema ser um assunto relativamente novo. O que mais dificultou a execução do projeto foi a busca de dados abertos. A utilização de ferramentas que ainda se encontravam em estágio de desenvolvimento, contendo pouca documentação e apresentando instabilidade, gerando em alguns casos a perda de dados, também aumentou a complexidade.

Analisando todo o projeto, concordamos que os objetivos propostos no início foram atingidos, apesar das dificuldades encontradas. Além disso, durante toda a fase de desenvolvimento percebemos oportunidades de melhoria.

## 7. Referências

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3), 1–22. IEEE. Retrieved from <http://tomheath.com/slides/2009-09-london-linked-data-the-story-so-far.pdf>

Heath, T., & Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space* (1st ed., p. 137). New York, New York, USA: Morgan & Claypool Publishers. Retrieved from <http://linkeddatabook.com/book>  
<http://uploading.com/files/7186ecce/1608454304Linked.rar>

Berners-Lee, T. (2006). Linked Data. *W3C Design Issues*. Retrieved from <http://www.w3.org/DesignIssues/LinkedData.html>

Bizer, C., Heath, T., Ayers, D., & Raimond, Y. (2007). Interlinking Open Data on the Web. *4th European Semantic Web Conference (ESWC2007)*. Retrieved from <http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkingOpenData.pdf>  
<http://www4.wiwiss.fu-berlin.de/is-group/page/publications/Publication25>

Breitman, K. K. (2005). *WEB SEMÂNTICA A INTERNET DO FUTURO*. (K. K. Breitman, Ed.) *Rio de Janeiro LTC* (p. 199). LTC. Retrieved from <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Web+Semantica+A+Internet+do+Futuro#0>