O que são dados abertos conectados? Qual o papel deles na construção da Web do futuro (i.e. Web dos dados)? E como podem ajudar a resolver problemas econômicos, sociais e de gestão? Conheça um pouco mais sobre o conceito de dados abertos conectados e as vantagens de criar um ecossistema para estruturação, disponibilização e consumo de dados no formato aberto. Saiba como utilizar os princípios conhecidos como "Sistema de 5 Estrelas" para classificar o grau de abertura dos dados disponíveis na Web.

Aprenda também um pouco sobre a área de Web Semântica, ontologias, modelagem de dados e padrões de compartilhamento de informação. Saiba quais são os métodos e padrões atualmente utilizados para o compartilhamento de metadados que dão significados aos dados da Web. E veja as tendências mais recentes no desenvolvimento de aplicações inteligentes que utilizam a semântica dos dados para geração de ferramentas que atendem às necessidades do mercado e da sociedade no século 21.

Cursos e certificações referentes ao conteúdo do livro são promovidos pelo Centro de Estudos sobre Tecnologias na Web (Ceweb.br). Estes cursos estão disponíveis em *ceweb.br/cursos*.

Espera-se que ao final da leitura deste livro o leitor:

- Tenha uma visão geral sobre dados abertos conectados e suas potenciais aplicações.
- Tenha também adquirido conhecimentos para modelar, publicar e consumir dados abertos.
- Entenda o processo de disponibilização de dados de maneira estruturada usando ontologias.
- Saiba mais sobre as diversas tecnologias da Web Semântica e suas potencialidades.
- Utilize as tecnologias semânticas tanto para disseminação dos dados quanto para criação de aplicativos simples.

Ig Ibert Bittencourt

Doutor em Ciência da Computação pela UFCG e pós-doutor pela Unicamp, é professor do Instituto de Computação da Universidade Federal de Alagoas e bolsista de Produtividade (DT) do CNPq. É vice-presidente da Comissão Especial de Informática na Educação da SBC, representante consultivo da UFAL no W3C e cofundador do Núcleo de Excelência em Tecnologias Sociais (NEES). É um dos fundadores de duas startups premiadas por seu caráter inovador, sendo uma na área de Tecnologias na Educação (MeuTutor) e outra na área de Tecnologias Semânticas (LinKn). Tem trabalhado com pesquisa, desenvolvimento e inovação nas áreas de Tecnologias Educacionais, Dados Abertos Conectados e Empreendedorismo Social.

Fique conectado:

twitter.com/novateceditora

facebook.com/novatec

www.novatec.com.br

ISBN 978-85-7522-449-6

DADOS NA WEB

DADOS ABERTOS CONECTADOS

novatec

DADOS ABERTOS CONECTADOS

Seiji Isotani
Ig Ibert Bittencourt

conteúdo oficial nic.br conteúdo oficial

novatec

ORGANIZAÇÃO ceweb.br

Seiji Isotani

Professor do Departamento de Sistemas de Computação da Universidade de São Paulo. Fez doutorado em Engenharia de Ontologias e Web Semântica na Osaka University, no Japão, e realizou seu pós-doutorado na Carnegie Mellon University, nos Estados Unidos. Publicou mais de uma centena de artigos científicos nas áreas de ontologias, dados abertos conectados e tecnologias educacionais. Junto ao W3C, atua em projetos relacionados à criação de um ecossistema para produção e consumo de dados abertos. É cofundador do Laboratório de Computação Aplicada à Educação (CAEd) e das empresas MeuTutor e Linkn, que trabalham no desenvolvimento de tecnologias semânticas.

DADOS ABERTOS CONECTADOS

Licença

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Isotani, Seiji
Dados abertos conectados / Seiji Isotani, Ig Ibert Bittencourt. -- São Paulo : Novatec Editora, 2015.

ISBN 978-85-7522-449-6

1. Acesso a dados públicos 2. Serviços de informação on-line 3. Tecnologia 4. Web semântica 5. Web sites - Desenvolvimento I. Bittencourt, Ig Ibert. II. Título.

15-07274 CDD-004.678

Índices para catálogo sistemático:

1. Web semântica : Acesso a dados abertos : Ciências da computação 004.678

MP20150817

DADOS ABERTOS CONECTADOS

Seiji Isotani

Ig Ibert Bittencourt

ceweb.br

Novatec

Esta é uma publicação do:

Núcleo de Informação e Coordenação do Ponto Br – NIC.br

Diretor Presidente: Demi Getschko

Diretor Administrativo: Ricardo Narchi

Diretor de Serviços: Frederico Neves

Diretor de Projetos Especiais e de Desenvolvimento: Milton Kaoru Kashiwakura

Diretor de Assessoria às Atividades do CGI.br: Hartmut Richard Glaser

Organização: Centro de Estudos sobre Tecnologias Web – Ceweb.br

Autores: Seiji Isotani e Ig Ibert Bittencourt

Coordenação Executiva e Editorial: Caroline Burle dos Santos Guimarães e Vagner Diniz

Revisão: Luisa Caliri

Designer da capa: Maricy Rabelo

Ilustrações: Ricardo Hurmus

Editora: Novatec

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Email: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

Sobre o CGI.br

O Comitê Gestor da Internet no Brasil tem a atribuição de estabelecer diretrizes estratégicas relacionadas ao uso e desenvolvimento da Internet no Brasil e diretrizes para a execução do registro de Nomes de Domínio, alocação de Endereço IP (Internet Protocol) e administração pertinente ao Domínio de Primeiro Nível ".br". Também promove estudos e recomenda procedimentos para a segurança da Internet e propõe programas de pesquisa e desenvolvimento que permitam a manutenção do nível de qualidade técnica e inovação no uso da Internet.

Sobre o NIC.br

O Núcleo de Informação e Coordenação do Ponto BR é uma entidade civil, sem fins lucrativos, que implementa as decisões e projetos do CGI.br. São atividades permanentes do NIC.br coordenar o registro de nomes de domínio (Registro.br), estudar, responder e tratar incidentes de segurança no Brasil (CERT.br), estudar e pesquisar tecnologias de redes e operações (Ceptro.br), produzir indicadores sobre as tecnologias da informação e da comunicação (Cetic.br), fomentar e impulsionar a evolução da Web no Brasil (Ceweb.br) e abrigar o escritório do W3C no Brasil (*http://www.w3c.br/*).

Sobre o Ceweb.br

O Centro de Estudos sobre Tecnologias Web (Ceweb.br) tem como missão disseminar e promover o uso de tecnologias abertas na Web, fomentar e impulsionar a sua evolução no Brasil por meio de estudos, pesquisas e experimentações de novas tecnologias. No escopo de atividades desenvolvidas pelo Centro, destacam-se o estímulo às discussões sobre o ecossistema da Web e a preparação de subsídios técnicos à elaboração de políticas públicas que fomentem esse ecossistema como meio de inovação social e prestação de serviços.

Livro em formato digital

Em *http://ceweb.br/publicacao/livro-dados-abertos/* estão disponíveis as versões desse livro, com todas as figuras coloridas, nos formatos HTML e PDF.

Sumário

Dedicamos este livro as nossas esposas, Cristiana e Monike, pelo apoio e carinho incondicionais demonstrados durante todas as viagens, reuniões, discussões e os períodos de isolamento que culminaram na escrita do primeiro livro de dados abertos conectados do Brasil.

Agradecimentos

Os autores deste livro gostariam de agradecer a todos os alunos, pesquisadores e professores que fazem (ou fizeram) parte do NEES – Núcleo de Excelência em Tecnologias Sociais – e do CAED – Laboratório de Computação Aplicada à Educação e Tecnologia Social Avançada –, que contribuíram direta e indiretamente com os estudos e as pesquisas em dados abertos conectados. A sinergia entre os laboratórios e o espírito questionador, inovador e de liderança de seus membros foi peça-chave para consolidação dos conhecimentos apresentados neste livro.

Estendemos nossos agradecimentos também ao W3C Brasil, NIC.br e CEWEB.br, pelo apoio inestimável tanto para criar um ecossistema de produção e consumo de dados abertos de qualidade no Brasil quanto para a geração de recursos de ensino e treinamento fundamentais para capacitação de profissionais nesta área. Em especial, agradecemos a Caroline Burle e Vagner Diniz pelo apoio e pela dedicação despendida, fundamentais para a concepção deste livro.

Sobre os autores

Seiji Isotani é pesquisador, inventor, empreendedor e professor do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP). Tem bacharelado e mestrado em Ciência da Computação pela Universidade de São Paulo (IME-USP). Concluiu seu doutorado na área de Engenharia de Ontologias e Web Semântica na Universidade de Osaka, no Japão. Realizou seu pós-doutorado na Universidade Carnegie Mellon, nos Estados Unidos, onde foi contratado e permaneceu no quadro docente (Faculty) até 2011. Tem experiência nas áreas de Ensino, Ciência da Computação, Desenvolvimento de Software e Transferência de Tecnologia, com ênfase em sistemas colaborativos e sistemas educacionais semânticos. Junto ao W3C, tem trabalhado em projetos de pesquisa e extensão relacionados à criação de um ecossistema para produção e consumo de dados abertos. É fundador e atual co-coordenador do Laboratório de Computação Aplicada à Educação e Tecnologia Social Avançada (CAEd), realizando pesquisas em ontologias, open linked data, web semântica, sistemas tutores inteligentes, aprendizagem colaborativa e mineração de dados educacionais. Também é cofundador da empresa MeuTutor, que desenvolve tecnologias semânticas para a Educação e ganhou diversos prêmios nacionais e internacionais.

Ig Ibert Bittencourt é doutor em Ciência da Computação pela UFCG (2009) e pós-doutor pela Unicamp (2013). É professor do Instituto de Computação da Universidade Federal de Alagoas e bolsista de Produtividade em Desenvolvimento Tecnológico e Extensão Inovadora do CNPq. Também é vice-presidente da Comissão Especial de Informática na Educação da Sociedade Brasileira de Computação, representante consultivo da UFAL no W3C e membro da SBC, ACM e W3C. É um dos fundadores e diretor do Núcleo de Excelência em Tecnologias Sociais (NEES), que objetiva a formação de líderes inovadores com visão social. Fundou duas startups premiadas por seu caráter inovador, sendo uma na área de Tecnologias na Educação (MeuTutor) e outra na área de Tecnologias Semânticas (LinKn). Atualmente tem trabalhado com pesquisa, desenvolvimento e inovação nas áreas de Tecnologias Educacionais, Dados Abertos Conectados e Empreendedorismo Social.

Prefácio

Caroline Burle dos Santos Guimarães
Vagner Diniz

> "*Dados são a alma da tomada de decisões e a matéria-prima para a prestação de contas. É quase impossível a concepção, o acompanhamento e a avaliação de políticas eficazes sem dados de alta qualidade que fornecem as informações corretas sobre as coisas certas no momento certo.*"
>
> *A World that Counts: Mobilising the Data Revolution for Sustainable Development*, Organização das Nações Unidas, 2014

O mundo está repleto de dados. Dados são criados toda vez que alguém utiliza seu celular, onde quer que esteja, que produtos com seus códigos de barras são fabricados, despachados, armazenados e vendidos, e que veículos com GPS vão e vêm pelas estradas, circulam pela rede e são passíveis de serem analisados, processados e transformados em informações de valor. Apesar da enxurrada de informações, de muitos dados sobre pessoas, governos, empresas, pouco ainda sabemos como fazer para que dados resolvam nossos problemas do cotidiano.

As tecnologias abertas têm se revelado instrumentos alavancadores de inovação. A possibilidade de inovação incremental a partir de processos colaborativos de uso da tecnologia permite mudanças disruptivas como a nova economia compartilhada, aplicativos sociais e reaproveitamento de códigos e dados. Os textos e os cursos abertos, como recursos educacionais abertos, geram capacitação e formam

pessoas aptas a utilizar essas tecnologias abertas, que podem produzir novos negócios e novas soluções para a sociedade.

Acreditamos também que as tecnologias abertas para a disponibilização de dados têm um potencial enorme de prover maior transparência e melhor participação dos cidadãos nas soluções dos seus problemas.

Com base nessas premissas, a equipe do Centro de Estudos sobre Tecnologias Web (Ceweb.br), do NIC.br, organizou o curso *online* avançado sobre publicação de dados em formato aberto[1], cujo conteúdo produzido pelos professores Seiji Isotani e Ig Ibert Bittencourt transformou-se neste livro. O conceito Dados Abertos Conectados, do inglês "Linked Data", foi criado por Tim Berners-Lee pela necessidade de padronizar a conexão entre dados na Web. Compreende-se que o uso dos padrões criados pelos Grupos de Trabalho do W3C e o trabalho da comunidade de desenvolvedores, de gestores governamentais e da sociedade interessada no desenvolvimento Web são essenciais para que se alcance efetivamente dados abertos e conectados.

Dados abertos de alta qualidade são dados publicados e distribuídos na Internet, compartilhados em formato aberto para que possam ser lidos por qualquer pessoa e por máquinas, permitindo o cruzamento com outros dados de diferentes fontes, para serem livremente reutilizados pela sociedade. Dados abertos governamentais são dados produzidos pelos governos, que devem ser colocados à disposição de qualquer cidadão e para qualquer fim (W3C BRASIL, 2011, p. 4). As definições de Dados Abertos e Dados Abertos Conectados são tidas como base para compreender os conceitos a serem tratados neste livro.

O primeiro capítulo – "Visão Holística: Da Produção ao Consumo de Dados Abertos" – apresenta o conceito de Dados Abertos e enfatiza a importância das três normas para publicação de dados em formato aberto: os dados precisam ter disponibilidade e acesso,

1 O curso online de Dados Abertos conectados está disponível em: <*http://ceweb.br/cursos/*>.

reúso e redistribuição e participação universal. Mostra os benefícios de abertura e publicação de dados e discorre sobre a criação do portal americano de dados abertos (data.gov.us) e da Parceria para Governo Aberto (*Open Government Partnership – OGP)*. Vale ressaltar que, sendo o Brasil um dos fundadores da OGP, ele incentivou a publicação de dados abertos no âmbito mundial.

A Lei de Acesso à Informação (LAI), que entrou em vigor no Brasil em maio de 2012, é citada pelos autores como arcabouço jurídico para a obrigação governamental de disponibilizar dados abertos no país. O ciclo de vida dos dados abertos também é analisado nesse capítulo, assim como a grande produção de dados – o *big data*. A Web Semântica é a visão do W3C sobre a Web de dados conectados, tópico relevante para se trabalhar dados de forma inteligente e automática.

Os autores explicam a transição da Web de documentos para a Web de dados e que Dados Conectados referem-se a "um conjunto de boas práticas para publicação e conexão de dados estruturados na Web, usando padrões internacionais recomendados pelo W3C". Ao final do capítulo inicial, descrevem o processo para publicação de dados abertos, detalhando-os nos capítulos seguintes.

O capítulo 2 – "Estruturação de Dados e Dados Abertos Conectados" – mostra que são essenciais para obter dados conectados os padrões de representação com o modelo RDF e a criação de triplas. Apresenta a importância do esquema de distribuição das cinco estrelas, proposto por Tim Berners-Lee, e o significado da classificação para atingir o ponto máximo, a quinta estrela. Os autores explicam sobre a necessidade de os dados serem disponibilizados sob licenças que permitam o seu reúso.

Esse capítulo também mostra como a DBPedia ajuda a tornar possível a Web de Dados e, por meio de exemplos, demonstra alguns casos reais como o data.gov.uk, da BBC, o caso de Nova York e o caso brasileiro da Globo.com. Todos chegaram às cinco estrelas dos dados abertos.

O terceiro capítulo – "Ontologias e Representação de Conhecimento" – explica a obrigação de usar ontologias para chegar aos Dados Abertos Conectados. Conceitua a composição de uma ontologia, os tipos que existem e a linguagem de ontologias da Web – OWL, em inglês *Ontology Web Language*. Esse padrão pode ser dividido em duas camadas: uma para descrever a sintaxe e outra para a semântica. Enfatiza-se, ainda, que toda ontologia criada em OWL 2 tem uma estrutura sintática obrigatória, baseada em RDF/XML.

O capítulo seguinte – "Engenharia de Ontologias" – detalha as metodologias de desenvolvimento de ontologias, abordando a complexidade na sua criação e as ferramentas existentes para criá-las. Descreve-se o ciclo de vida de uma ontologia, que começa na especificação, segue para a conceitualização, é formalizada, em seguida é implementada para, posteriormente, existir um cuidado com a sua manutenção. Esse capítulo ainda mostra como criar uma ontologia, com o exemplo da Ontologia de Vinho.

O último capítulo – "Desenvolvimento de Aplicações Semânticas" – disserta sobre os padrões de desenvolvimento, especialmente o desenvolvimento orientado a triplas RDF e o desenvolvimento orientado a objetos. Também são citadas ferramentas para desenvolvimento de aplicações semânticas, plataformas para publicação de dados e frameworks para manipulação de RDF, bancos de dados RDF e sistemas de mapeamento objeto-ontologia.

Este livro mostra a importância dos Dados Abertos conectados, que podem contribuir para o uso da Web como principal meio para inovação social. A amálgama dos dados na Web ubíqua pode facilitar a vida dos usuários, e, assim como a Web revolucionou o consumo de documentos, os Dados Abertos Conectados podem revolucionar o acesso aos dados e a maneira como estes são usados.

CAPÍTULO 1

Visão Holística: Da Produção ao Consumo de Dados Abertos

1.1 Introdução

Saber trabalhar com grandes quantidades de dados procedentes de diversas localidades e com diferentes formatos é uma das habilidades mais desejadas na última década (Davenport et al., 2012). Isso ocorre devido ao crescimento exponencial dos dados gerados pela sociedade e à necessidade de minerar informações obtidas por meio da análise das conexões semânticas entre conceitos e relações presentes nestes dados (Isotani et al., 2009; Bittencourt et al., 2008). Para exemplificar essa problemática, um estudo desenvolvido pela *IDC Digital Universe* fez uma medição dos dados criados replicados e consumidos em 2011 e estimou que 1,8 trilhão de gigabytes de dados foi produzido durante o ano (EMC, 2012). Contudo, grande parte destes dados não está disponível ao público. Estes dados tampouco estão estruturados para facilitar sua compreensão mesmo por aqueles que podem acessá-los e manipulá-los. Como resultado, a extração de informações e a produção de conhecimentos que poderiam ser úteis para a sociedade não acontecem com a agilidade e a eficácia necessárias para lidar com as questões sociais e econômicas do século 21.

Para modificar esta situação, diversas empresas, governos e institutos de pesquisa têm realizado esforços para disponibilizar dados e produzir tecnologias *web* que permitam criar um *ecossistema de produção e consumo de dados* com o objetivo de agilizar a descoberta de novos conhecimentos e agregar valor a qualquer informação disponibilizada livremente na Internet.

Neste contexto, o objetivo principal deste texto é fomentar e incentivar a qualificação de profissionais por meio da capacitação teórica, técnica e tecnológica que apoiam a criação e a manutenção deste ecossistema. Em particular, focaremos métodos e ferramentas para modelar e estruturar os dados de maneira adequada (por exemplo, por meio de ontologias) para que estes possam ser utilizados e reutilizados por programas de computador. Além disso, pretende-se apresentar técnicas de desenvolvimento de software utilizando tecnologias avançadas provindas da área de *Web Semântica* (Berners-Lee et al., 2001) e *Linked Data* (Berners-Lee, 2006) para fazer uso efetivo dos dados publicados na Web.

1.1.1 Motivação

Antes de introduzir os conceitos teóricos e técnicos, gostaríamos de iniciar o texto com um exemplo que incentive o leitor a pensar sobre as necessidades e potencialidades de produzir e disponibilizar dados livremente na Web.

Propomos pensar um pouco sobre a gestão de cidades, estados e países. Para manter os vários serviços públicos oferecidos pelo governo, existem diversas informações que precisam ser compartilhadas, integradas e gerenciadas. Por exemplo, o serviço de transporte público urbano abrange frotas de ônibus, trens ou metrô, que são concessões municipais, estaduais ou federais. Além disso, existem também as frotas intermunicipais e interestaduais que conectam diferentes municípios e estados. Neste contexto, para utilizar este serviço e verificar se ele está sendo oferecido com qualidade e

eficiência desejadas, são necessários produção e consumo de dados que normalmente são armazenados em diferentes bases de dados com informações sobre os diversos tipos de transporte em operação.

Imagine que os dados sobre os transportes públicos estejam disponíveis apenas para os técnicos e funcionários autorizados. Isso significa que estes dados não podem ser acessados pelos cidadãos e, portanto, não há como obter informações sobre os horários, as rotas e os destinos dos transportes públicos, inviabilizando o planejamento das pessoas que precisam utilizar este meio de locomoção.

Contudo, se os dados sobre os transportes públicos estivessem disponíveis livremente na Web em formato aberto[1], um cidadão poderia ter acesso às informações contidas nestes dados e utilizá-las a seu favor. Por exemplo, poderia planejar uma rota simples da sua casa ao trabalho utilizando diferentes meios de transporte; poderia também comparar o custo-benefício de diferentes rotas e tipos de transporte. Da mesma forma, um funcionário de um município também usufruiria de benefícios, pois poderia facilmente acessar dados de outros municípios e do estado para realizar suas atividades. Por exemplo, poderia adequar os horários dos transportes locais, alocar mais veículos em horários com maior demanda e também comparar os dados locais com os de municípios similares para analisar a eficiência do serviço.

Apesar dos benefícios dos dados disponibilizados em formato aberto apresentados no parágrafo anterior, há um grande problema quando surge a necessidade de associar diversos dados ou quando a quantidade de dados é excessiva para interpretação humana. Por exemplo, se o cidadão quiser verificar qual a média de pessoas que pegam um determinado tipo de transporte em um determinado horário para evitar horários de pico em rotas com conexões intermunicipais ou interestaduais, ele será obrigado a verificar cada uma das bases de dados dos diferentes municípios e/ou estados, para então

1 É importante frisar que, apesar de os dados poderem ser abertos em qualquer formato, desde que tenham licenças abertas, veremos a seguir que é recomendável que os dados sejam abertos pelo menos em formato CSV, porém, idealmente, em formato RDF.

analisar e planejar manualmente cada uma das conexões e cada um dos horários, de forma a criar o melhor trajeto para atender as suas necessidades. Este problema pode ser ainda mais complexo para os funcionários que gerenciam o transporte público, pois, para obter as informações sobre os diversos serviços oferecidos pelos municípios, eles teriam que manualmente acessar centenas de bases de dados, que possivelmente contêm informações diferentes, para, assim, extrair e compilar as informações desejadas.

Nesse cenário, caso os dados estivessem estruturados e conectados de forma que um computador pudesse processá-los, então as tarefas apresentadas poderiam ser automatizadas (Barros et al., 2011). Um aplicativo computacional poderia acessar os dados das diversas bases de municípios, dos estados e da federação para planejar trajetos em qualquer região do país. Seria possível criar também um aplicativo para analisar o transporte público de centenas de municípios para realizar diversas comparações e análises estatísticas que são fundamentais para obter uma visão geral sobre a cobertura e a qualidade do serviço realizado. Assim, em vez de demorar dias, tarefas complexas e trabalhosas poderiam ser realizadas em poucos minutos.

Para que isso ocorra é necessário lidar com diversos desafios, que podem ser resumidos em como gerenciar, coletar, modelar, padronizar e consumir dados adequadamente. Assim, nesta primeira parte do texto, apresentaremos os fundamentos teóricos necessários para lidar com estes desafios e para ter uma visão holística e conceitual sobre como criar um ambiente propício para produção e consumo de dados.

1.2 Dados Abertos

Esta seção tem por objetivo fazer uma revisão sobre dados abertos, abordando os conceitos de dados abertos e dados abertos governamentais, sua importância, seus princípios e, finalmente, um panorama sobre dados abertos no Brasil e no mundo.

É importante frisar que esta seção não objetiva estressar a temática de Dados Abertos, tendo em vista que tal conhecimento está disponibilizado no curso *Publicação de Dados em Formato Aberto* (W3C Brasil, 2013), disponível no site da *Escola de Políticas Públicas*.

Segundo a *Open Definition* (Open Definition, 2014), dados abertos são dados que podem ser livremente utilizados, reutilizados e redistribuídos por qualquer pessoa – sujeitos, no máximo, à exigência de atribuição *à fonte original e ao compartilhamento pelas mesmas* licenças em que as informações foram apresentadas. Ou seja, a abertura de dados está interessada em evitar um mecanismo de controle e restrições sobre os dados que forem publicados, permitindo que tanto pessoas físicas quanto jurídicas possam explorar estes dados de forma livre.

Por esta perspectiva, a definição do termo *dados abertos* carrega três normas fundamentais (Open Knowledge Foundation, 2010):

- **Disponibilidade e acesso:** os dados devem estar disponíveis como um todo e sob custo não maior que um custo razoável de reprodução, e preferencialmente devem ser possíveis de ser baixados pela Internet. Os dados devem também estar disponíveis de uma forma conveniente e modificável.
- **Reúso e redistribuição:** os dados devem ser fornecidos sob termos que permitam a reutilização e a redistribuição, inclusive a combinação com outros conjuntos de dados.
- **Participação universal:** todos devem ser capazes de usar, reutilizar e redistribuir – não deve haver discriminação contra áreas de atuação ou contra pessoas ou grupos. Por exemplo, restrições de uso "não comercial" que impediriam o uso "comercial", ou restrições de uso para certos fins (ex.: somente educativos) excluem determinados dados do conceito de "abertos".

A partir do momento que há um movimento de abrir dados, em que as três normas fundamentais supracitadas são respeitadas, é possível que diferentes organizações e sistemas possam trabalhar de

forma colaborativa. Isso ocorre devido à capacidade dessas organizações e desses sistemas de interoperar os dados que foram abertos, ampliando assim a comunicação e potencializando o desenvolvimento eficiente de sistemas complexos. Para tal, os dados devem ser acessíveis, legíveis por máquinas, ter formato aberto e informação produzida por todos e para todos.

Apesar de o movimento de dados abertos ter sido impulsionado mais recentemente, o termo "Dados Abertos" surgiu em 1995 em um documento de uma agência científica americana, abordando a divulgação de dados ambientais e geofísicos (Chignard, 2013). No documento, os autores promoveram um completo e aberto intercâmbio de informação científica entre diferentes países como um pré-requisito para a análise e a compreensão de fenômenos naturais globais. É importante frisar que o princípio de bens comuns aplicado ao conhecimento já foi teorizado por Robert King Merton, em 1942, quando sua teoria mostrou os benefícios de dados científicos abertos.

Mais recentemente, os dados abertos têm sido estimulados por movimentos globais, como a campanha feita pela Iniciativa Internacional de Transparência em Programas de Assistência (do inglês, *International Aid Transparency Initiative*) sobre a transparência nos registros dos gastos.

Outro importante acontecimento para o movimento de dados abertos aconteceu em dezembro de 2007, quando pensadores e ativistas da Internet se reuniram para definir o conceito de *Dados Abertos Públicos (ou Governamentais)* (Chignard, 2013). A ideia básica desenvolvida foi que os dados governamentais são propriedades comuns, da mesma forma que as ideias científicas. A filosofia por trás do conceito de dados governamentais abertos é inspirada no conceito de *código aberto* (do inglês, *open source*), fundamentada por três pilares conceituais: abertura, participação e colaboração.

Essa visão de dados abertos governamentais foi fortalecida em 2008, após um memorando do presidente Barack Obama sobre Transparência e Dados Governamentais (Obama, 2008) e pela

criação do *Data.gov*, com o objetivo de disponibilizar um portal de dados abertos no qual os dados do governo norte-americano poderiam ser acessados na Internet por qualquer cidadão.

Nessa perspectiva, o governo brasileiro foi um dos fundadores da *Open Government Partnership*, criada em 2011, que conta atualmente com a participação de 65 países. O governo brasileiro criou também o dados.gov.br, portal que disponibiliza dados governamentais seguindo os princípios de dados abertos. Os dados abertos governamentais fazem parte da política de acesso à informação do governo federal (INDA); ambos foram criados em 2012[2]. A Administração Pública Federal, por meio da Lei de Acesso à Informação[3], em seu artigo 8º, reconheceu a necessidade de disponibilizar dados governamentais em formato aberto.

1.3 Da Grande Produção de Dados aos Dados Conectados

Em um estudo realizado em 2003 por pesquisadores da Escola de Gerenciamento de Informação e Sistemas da Universidade de Berkeley, estimou-se que a humanidade tenha acumulado 12 exabytes (12×10^{18}) de dados até o momento anterior aos computadores virarem commodities, na década de 90. Entretanto este mesmo estudo mostrou que somente no ano de 2002 a humanidade produziu mais de cinco exabytes em mídias de armazenamento óptico, magnético, filme e impressão. Esta produção é equivalente à produção de 37 mil novas bibliotecas do tamanho da Biblioteca do Congresso Americano. Destes cinco exabytes produzidos, 92% foram armazenados em mídias magnéticas, a maioria em discos rígidos, o que aponta para uma incrível democratização da informação (Floridi, 2010).

A Figura 1.1 apresenta um ciclo de vida típico da informação. Observe a temporalidade presente no ciclo da informação e como esta é consumida para a geração de novos dados que serão consumidos

2 Disponível em: <*http://www.governoeletronico.gov.br/biblioteca/arquivos/instrucao-normativa-da-infraestrutura-nacional-de-dados-abertos-2013-inda*>.

3 Lei nº 12.527 de 18 de novembro de 2011, disponível em: <*http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm*>.

infinitamente. É importante frisar que a geração de novos dados baseada em dados anteriormente consumidos é inerente à sociedade e mostra a importância que a estruturação e a conexão de dados têm para simplificar e facilitar a recuperação da informação e a produção de novos conhecimentos.

Figura 1.1 – Ciclo de Vida da Informação. Fonte: adaptado de Floridi (2010).

De fato, a informação exerce papel fundamental no desenvolvimento da sociedade pós-moderna. Ao observar os membros do G7 – Canadá, França, Alemanha, Itália, Japão, Reino Unido e Estados Unidos –, vemos que todos eles podem ser qualificados como sociedades da informação, pois pelo menos 70% dos seus PIBs dependem de bens intangíveis (bens relacionados à informação) (Floridi, 2010). Ou seja, o funcionamento e o crescimento dessas nações dependem da constante geração e do consumo de dados de forma massiva.

Em um estudo mais recente, foi destacado que, de 2006 a 2010, o número de dados digitais gerados cresceu de 166 exabytes para 988 exabytes. Esta grande quantidade de dados que tem sido gerada já alcançou a casa dos *zettabytes* (1.000 exabytes ou 10^{24}) em 2014 (Floridi, 2010). Como descrito anteriormente, mais de 90% desses dados estão armazenados em discos rígidos, e centenas de milhões de computadores, ininterruptamente, estão processando esses dados em busca de informação útil e relevante e, às vezes, nova. Estima-se que, em 2020, o volume de dados chegará a 40 zettabytes, como mostra a Figura 1.2.

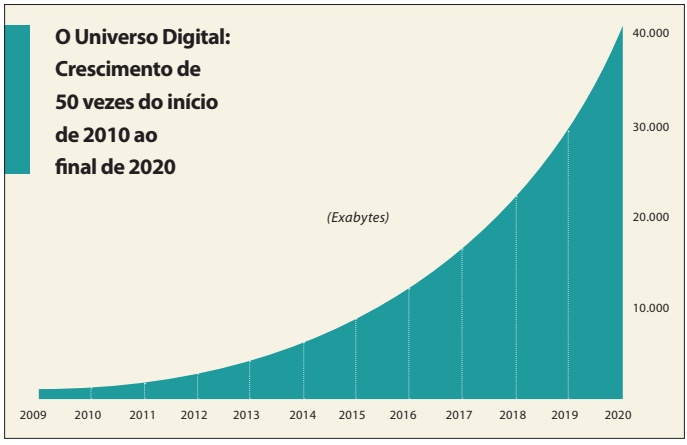

Figura 1.2 – Estimativa de crescimento de dados digitais de 2010 a 2020. Fonte: adaptado de EMC (2012).

Este incrível crescimento de dados faz com que seja fundamental uma cuidadosa análise desses dados e a compreensão sobre quais tipos de dados têm sido gerados nesta sociedade da informação. Os dados digitais estão, de alguma forma, estruturados e têm sido gerados com o intuito de prover informações úteis e gerar novos conhecimentos (vide Figura 1.3).

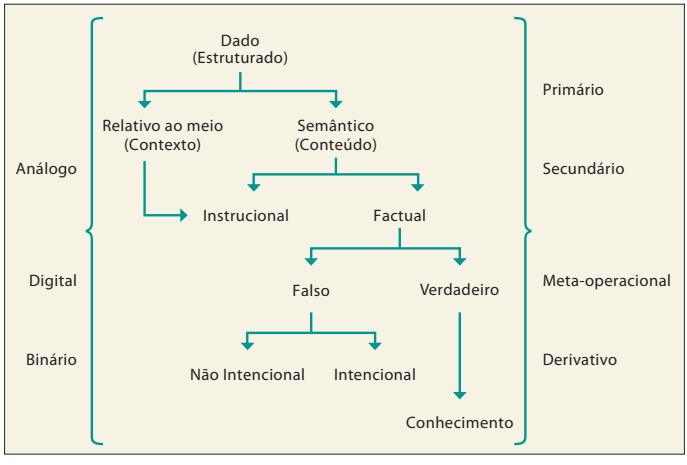

Figura 1.3 – Dados Binários, Digitais e Análogos. Fonte: adaptado de Floridi (2010).

É importante frisar que muitos desses dados podem ser descobertos e acessados tanto por seres humanos quanto por máquinas, por meio da *World Wide Web*. Em 1989, o físico inglês Sir Timothy John Berners-Lee[4], no CERN[5], inventou a WWW[6] (World Wide Web) a partir da proposição de três tecnologias fundamentais: o HTML (Hypertext Markup Language), o servidor HTTP (Hypertext Transfer Protocol) e o URI (Unified Resource Identifier). Tais invenções foram motivadas pela necessidade de facilitar o compartilhamento de documentos entre os pesquisadores e visitantes do CERN. Com isso, tanto o lançamento do browser *X Windows Mosaic 1.0* quanto a primeira conferência sobre a WWW estimularam o desenvolvimento da Web. Em maio de 1994, houve a primeira conferência internacional sobre a WWW, em Genebra[7]. Esta conferência simboliza a grande popularização da Web, pois nela foi anunciado o consórcio que cuida dos padrões e das tecnologias relacionados ao desenvolvimento da Web, o W3C (*World Wide Web Consortium*).

Ainda nesta conferência, Tim Berners-Lee proferiu uma palestra sobre a necessidade de semântica na Web. Segundo Tim Berners-Lee, a forma que os documentos *web* estavam estruturados (por meio de nós e links) fazia com que apenas seres humanos pudessem entender o significado contido neles. Isso impedia que máquinas pudessem acessar e obter significado dos documentos. A Figura 1.4 ilustra o relacionamento entre páginas sem semântica explícita.

De forma mais detalhada, os documentos utilizam a linguagem HTML, que é uma linguagem que faz apresentação de hipertextos. Os hipertextos são documentos que contêm links e nós (ou pontos de conexão) com outros documentos, permitindo assim a navegação entre si. Entretanto os links que existem para relacionar os documentos

4 Também conhecido por Tim Berners-Lee ou TBL.
5 Do francês Conseil Européen pour la Recherche Nucléaire (ou Centro Europeu de Pesquisas Nucleares).
6 É importante frisar que o WWW foi inicialmente motivado pelo ENQUIRE, também desenvolvido por TIM Berners-Lee.
7 First International Conference on World Wide Web.

não tinham nenhum tipo de característica que diferenciasse um do outro, de tal forma que não é possível para as máquinas distinguir o significado entre uma relação e outra.

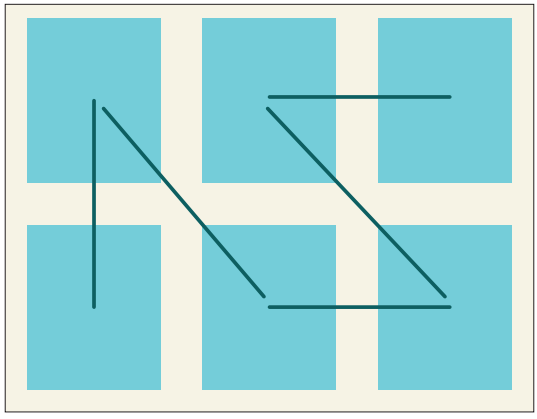

Figura 1.4 – Páginas web e relacionamentos. Fonte: adaptado de Berners-Lee (1994).

É importante frisar que tais problemas ficam mais evidentes nos dias de hoje, pois se calculam dezenas de bilhões de páginas *web* disponíveis e mais de um zettabyte de dados. Esta quantidade de documentos torna praticamente impossível o acesso e a busca por informação de forma eficiente e consistente para os seres humanos, fazendo com que haja a necessidade de entidades de softwares buscarem informações e processarem atividades para os humanos (Bittencourt et al., 2008). Com isso, a semântica acessível por máquina é potencializada por meio da especificação de documentos *web* em uma linguagem que permita que os links sejam criados com valor em seu relacionamento. Isso faz com que os recursos tenham uma semântica associada, permitindo a execução automática de atividades como compra de produtos personalizados, negociação por pacotes turísticos, agendamento de consultas, entre outras. A Figura 1.5 descreve os documentos e os relacionamentos com os recursos.

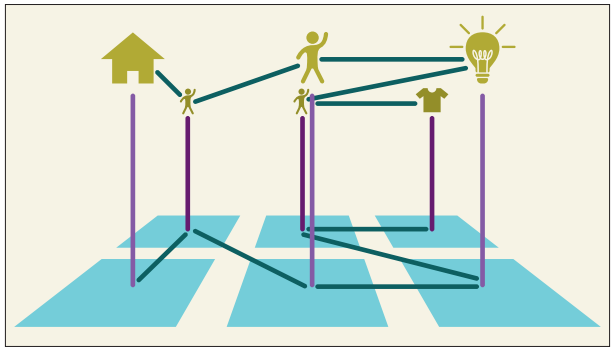

Figura 1.5 – Documentos web e relacionamentos com outros recursos. Fonte: adaptado de Berners-Lee (1994).

Passados cinco anos desde sua palestra na conferência WWW, Tim Berners-Lee publicou um livro intitulado *Weaving the Web*, no qual o termo *Web Semântica* apareceu pela primeira vez. Finalmente, em 2001, o artigo publicado na revista *Scientific American* marcou o início da pesquisa relacionada à *Web Semântica*. Tal artigo foi intitulado (e traduzido para o português) de "Web Semântica: Um novo formato de conteúdo para a Web que tem significado para computadores vai iniciar uma revolução de novas oportunidades[8]". Neste artigo, Tim Berners Lee aborda características da Web Semântica, propondo as camadas da *Web Semântica* e descrevendo como poderia funcionar.

De forma mais detalhada, a Web Semântica busca utilizar recursos provenientes da Inteligência Artificial (como agentes inteligentes e representação de conhecimento), Engenharia de Software (como frameworks e plataformas), Computação Distribuída (como *web services*), entre outros, para executar atividades na Web que antes só eram possíveis por agentes humanos.

Ou seja, a *Web Semântica* estende a *web* clássica, provendo uma estrutura semântica para páginas *web*, a qual permite que tanto agentes humanos quanto agentes de software possam entender o conteúdo presente em páginas *web*. Dessa forma, a *Web Semântica* provê um

8 Do inglês "The Semantic Web: a new form of Web content that is meaningful to computer will unleash a revolution of new possibilities" (Berners-Lee et al., 2001).

ambiente em que agentes de software podem navegar através de páginas *web* e executar tarefas sofisticadas. Em outras palavras, a *Web Semântica* é necessária para expressar informações de forma precisa, podendo tais informações ser interpretadas por máquinas e, dessa forma, permitir que agentes de software possam processar, compartilhar, reusar, além de poder entender os termos que estão sendo descritos pelos dados (Devedzic, 2006; Isotani et al., 2009).

Um cenário clássico, ilustrado no artigo de 2001 sobre a *Web Semântica*, descreve a marcação de uma consulta médica. Nele, basicamente, Lucy precisava marcar uma consulta para a sua mãe e levá-la ao médico. Houve a necessidade de localizar médicos qualificados e com o mesmo plano de saúde que a mãe de Lucy. Ela precisava estar disponível no horário da consulta da mãe e não poderia perder muito tempo com o deslocamento, ou seja, a consulta deveria ser em um local próximo à casa de sua mãe. Logo, Lucy precisaria configurar seu agente para que ele pudesse recuperar o tratamento médico, serviços disponíveis e consultórios próximos à casa de sua mãe e finalmente, marcar a consulta que melhor se encaixasse com as exigências de Lucy. A Figura 1.6 ilustra alguns dos recursos que podem ser utilizados em um cenário como este. Nela aborda-se a integração de ontologias, agentes inteligentes e serviços semânticos para alcançar o objetivo apresentado neste cenário.

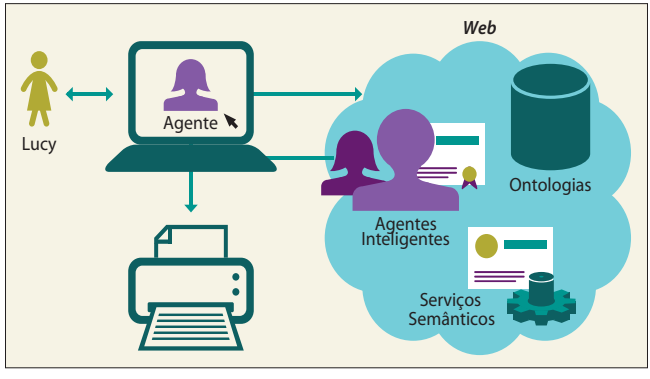

Figura 1.6 – Tecnologias da Web Semântica para processar uma consulta médica. Fonte: adaptado de Bittencourt (2009).

Objetivando tornar a visão da *Web Semântica* mais factível em termos de implementação, Tim Berners-Lee propôs um modelo em camadas, conhecido como "bolo de noiva" ou "pirâmide da Web Semântica", descrevendo os recursos e as linguagens para a Web Semântica, tal como mostrado na Figura 1.7.

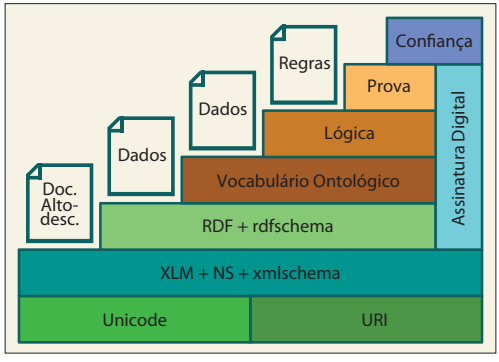

Figura 1.7 – Camadas da Web Semântica inicialmente proposta. Fonte: adaptado de Berners-Lee (2000).

Após a publicação do artigo sobre Web Semântica na *Scientific American*, em 2001, a área tem se desenvolvido bastante, o tema tem evoluído rapidamente e seus resultados são divulgados em diversas conferências, periódicos, livros, grupos de pesquisa, entre outros, tanto no Brasil quanto em outros países. Ocorreram vários avanços desde então em busca do desenvolvimento das camadas propostas por Tim Berners-Lee (Dubost and Herman, 2008). Entretanto, apesar de diversos trabalhos atuais ainda utilizarem esta perspectiva das camadas da Web Semântica, em 2007 foi feita uma reavaliação das camadas propostas, descrevendo as tecnologias que foram recomendadas pelo W3C. A Figura 1.8 apresenta a nova perspectiva das camadas da Web Semântica.

Uma visão ainda mais atualizada sobre a pilha de tecnologias que envolve a Web Semântica é apresentada na Figura 1.9, mostrando toda a complexidade e as tecnologias envolvidas com a área.

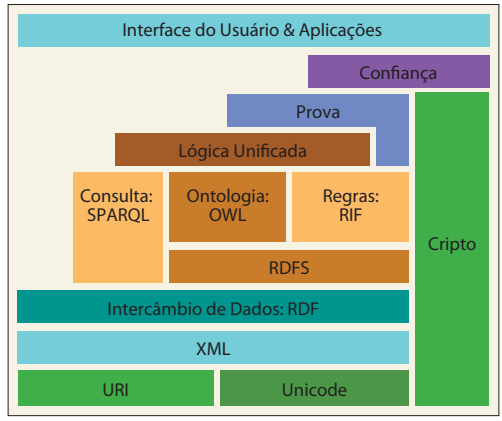

Figura 1.8 – Camadas da Web Semântica Revisitada. Fonte: adaptado de Berners-Lee (2006).

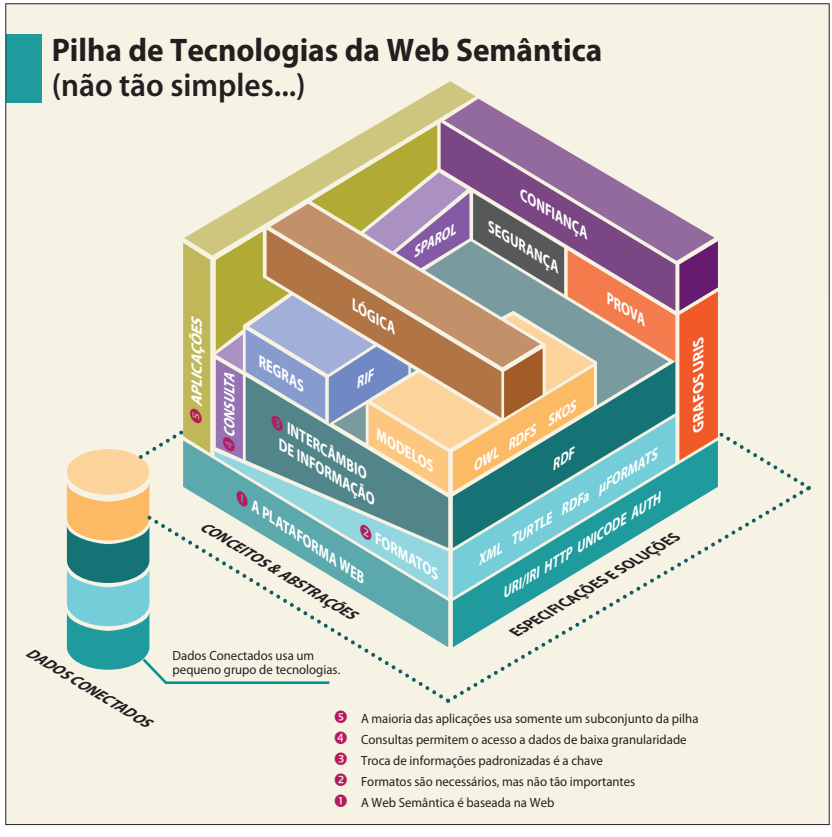

Figura 1.9 – Estado Atual da Web Semântica. Fonte: adaptado de Nowack (2009).

Como parte do desenvolvimento da Web Semântica, surgiu o conceito de *Linked Data*.

1.4 Dados Conectados e Dados Abertos Conectados

Antes de apresentarmos os conceitos relacionados, é importante salientar que o termo "Dados Conectados" é utilizado pelos autores como a tradução mais adequada para fazer referência ao conceito de *Linked Data*. Apesar de estarmos usando o termo Dados Conectados, outros pesquisadores fazem menção ao mesmo conceito usando os termos dados interligados ou dados ligados. Vários dicionários traduzem o termo *Linked* como ligado, conectado, associado, entre outros. Nós acreditamos que o termo ligado não é o mais adequado para transmitir o significado atribuído ao termo "*linked*" dentro do nosso contexto. Ao observar o termo "Ligar", há muitos cenários que não estão adequadamente aplicados a *Linked Data*. No entanto isso não acontece ao considerarmos a palavra "conectar". Em português, não haverá dúvidas sobre a semântica (ou o significado) esperada com o termo "conectar". Outro ponto fundamental para a escolha está nos princípios de *Linked Data*, principalmente ao se observar os dois últimos, em que fica claro que o objetivo proposto foi literalmente conectar dados na Web. Com relação ao termo interligado, existe um problema, visto que a tradução inversa deste termo para o inglês é *interlinked*. Levando essas informações em consideração, resolvemos traduzir *Linked Data* como Dados Conectados.

O Conceito de Dados Conectados (do inglês, Linked Data[9]) pode ser definido como um conjunto de boas práticas para publicar e conectar conjuntos de dados estruturados na Web, com o intuito de criar uma "Web de Dados" (Bizer et al., 2006). Estas práticas são fundamentadas em tecnologias *web*, como HTTP (Hypertext Transfer Protocol) e URI (Uniform Resource Identifier), com o objetivo de permitir a

9 A partir deste momento iremos utilizar apenas o termo em português, Dados Conectados.

leitura dos dados conectados, de forma automática, por agentes de software. A Web de Dados cria inúmeras oportunidades para a integração semântica dos próprios dados, motivando o desenvolvimento de novos tipos de aplicação e de ferramenta, como navegadores e motores de busca. Ao se observar as camadas da Web Semântica, os Dados Conectados podem ser considerados como descrito na Figura 1.10. Observemos também que, na Figura 1.9, os Dados Conectados também estão presentes, porém representando uma pequena parcela das tecnologias que compõem a Web Semântica.

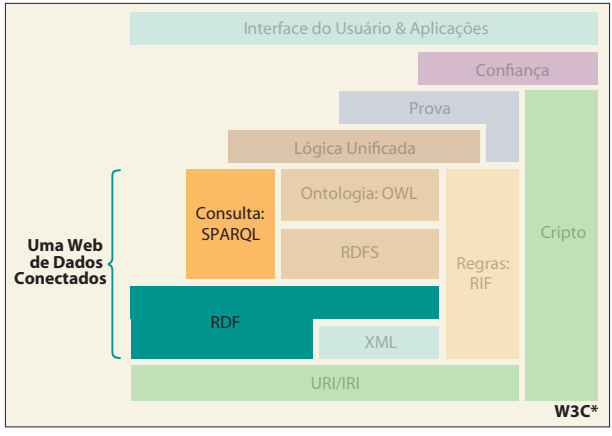

Figura 1.10 – A visão de Dados Conectados segundo as camadas da Web Semântica. Fonte: adaptado de Gandon et al. (2013).

Para um melhor entendimento sobre a Web de Dados, pode-se estabelecer um paralelo entre a Web de Documentos (a Web atual) e a Web de Dados. A primeira faz uso do padrão HTML para acessar dados, enquanto na segunda os dados são acessados a partir do padrão RDF (*Resource Description Framework)*. Na Web de Documentos, hiperlinks são usados para navegar entre as páginas, enquanto na Web de Dados os links RDF são usados para acessar dados de diversas fontes.

A Web de Documentos é baseada em um conjunto de padrões, incluindo: um mecanismo de identificação global e único, os URIs (*Uniform Resource Identifier*); um mecanismo de acesso universal, o

HTTP; e um formato-padrão para representação de conteúdo, o HTML. De modo semelhante, a Web de Dados tem por base alguns padrões[10], como: o mesmo mecanismo de identificação e acesso universal usado na Web de documentos (os URIs e o HTTP); um modelo-padrão para representação de dados, o RDF; e uma linguagem de consulta para acesso aos dados, a linguagem SPARQL.

Na Figura 1.11, podemos visualizar no diagrama apresentado a relação que existe entre Web Semântica, Dados Conectados, RDF e os diversos formatos de dados estruturados.

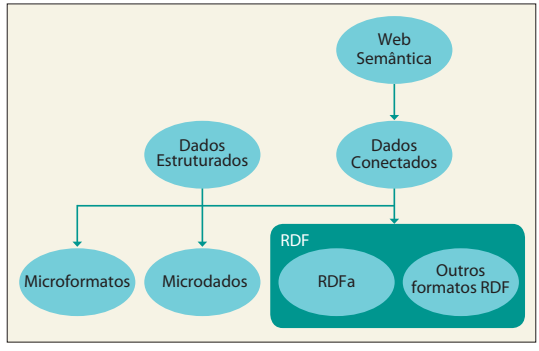

Figura 1.11 – Ecossistema de Dados Estruturados. Fonte: adaptado de Wood et al. (2014).

Desta forma, os padrões de Dados Conectados permitem que qualquer pessoa publique os dados de uma maneira que possam ser lidos por pessoas e processados por máquinas. Isso é possível porque os dados que antes estavam "escondidos" na Web de Documentos estão agora acessíveis graças à utilização dos padrões supracitados para a conexão de dados. Esta conexão de dados permite que todos (homens e máquinas) possam trabalhar conjuntamente de forma mais eficiente (como no desenvolvimento de aplicações para os cidadãos com o objetivo de melhorar o transporte público[11]). Os cenários de utilização como o motivado na Seção 1.1 apresentam o incrível potencial que os Dados Conectados podem proporcionar, desde

10 Todos estes conceitos e padrões serão detalhados em capítulos posteriores.
11 Para mais detalhes, vide Seção 1.1.

negócios até a melhoria do sistema público. Dados Conectados que podem se integrar com os outros dados e consequentemente formar novos conhecimentos demonstram a importância de explorar esta área. Outro exemplo: uma das maiores empresas de comércio eletrônico, a Best Buy, conseguiu melhorar entre 15% e 30% o número de clicks via buscador Google para o seu site por meio da utilização do formato de serialização de Dados Conectados, RDFa.

A área de Dados Conectados surgiu em 2006, com a publicação, também por Tim Berners-Lee, do documento *Design Issues* (Berners-Lee, 2006), com uma subseção de Web Semântica exclusiva para Dados Conectados. Em menos de uma década, as seguintes empresas fizeram algumas realizações (Wood et al., 2014): i) a Google anunciou a utilização do formato de serialização JSON-LD[12] para o Gmail; ii) a IBM anunciou que o Banco de Dados DB2 se tornaria um servidor de Dados Conectados; iii) Facebook expôs os Dados Conectados via Graph API; iv) a BBC usou Dados Conectados para gerar páginas de três de seus produtos; v) o governo britânico disponibilizou várias de suas fontes de dados em formato RDF (Data.gov, 2014).

O termo "Dados Conectados" se refere a um conjunto de boas práticas para publicação e conexão de dados estruturados na Web, usando padrões internacionais recomendados pelo W3C. É importante frisar que "Dados Conectados" não necessariamente precisam ser abertos. Por exemplo, uma entidade privada pode conectar dados, mas não necessariamente deixá-los abertos, como é o caso da *Open Corporates*[13], que tem a maior base conectada de corporações do mundo. Outro exemplo é querer disponibilizar dados de forma conectada por meio de *Capability URLs* (Tennison, 2014) com o objetivo de "esconder" dados que sejam de acesso privado. Um novo conceito que vem surgindo, e que já tem uma agenda de pesquisa, é chamado de *Dados Fechados Conectados*[14] (Cobden et al., 2011).

12 Recomendação do W3C para serialização de Dados Conectados que é baseada no JSON (JSON-LD será discutido em mais detalhes nos capítulos subsequentes).
13 Opencorporates.com
14 Do inglês, *Linked Closed Data*.

Apesar de iniciativas em Dados Conectados de forma fechada, muitas iniciativas (principalmente governamentais) estão se preocupando com a conexão e a publicação dos dados de forma aberta. Esta visão fica bastante clara ao se observar os princípios propostos por Tim Berners-Lee. Estes princípios são conhecidos como "Sistema de 5 Estrelas" (vide Figura 1.12), um sistema que classifica por meio de estrelas o grau de abertura dos dados. Quanto mais aberto, maior o número de estrelas para os dados e mais facilidade para ser enriquecido (conectado).

Figura 1.12 – Caneca das 5 Estrelas de Dados Conectados. Fonte: adaptado de CafePress (2011).

As 5 estrelas para Dados Abertos são:

1. Disponível na Internet (em qualquer formato; por exemplo, PDF), desde que com licença aberta, para que seja considerado Dado Aberto.
2. Disponível na Internet de maneira estruturada (em um arquivo Excel com extensão XLS).
3. Disponível na Internet, de maneira estruturada e em formato não proprietário (CSV em vez de Excel).

4. Seguindo todas as regras anteriores, mas dentro dos padrões estabelecidos pelo W3C (RDF e SPARQL): usar URL para identificar coisas e propriedades, de forma que as pessoas possam direcionar para suas publicações.
5. Todas as regras anteriores, mais: conectar seus dados a outros dados, de forma a fornecer um contexto.

Como dito anteriormente, é aconselhável que os dados sejam abertos considerando no mínimo 3 estrelas, porém estamos interessados neste documento em dados abertos a partir de 4 estrelas. Na Tabela 1.1, replicamos uma relação de benefícios da publicação de dados seguindo a classificação das 5 estrelas tanto para quem publica quanto para quem os consome (W3C Brasil, 2013):

Tabela 1.1 – Benefícios da publicação de dados (W3C Brasil, 2013)

Estrelas	Quem consome	Quem publica
★	☐ Ver os dados ☐ Imprimi-los ☐ Guardá-los (no disco rígido ou em um pen-drive, por exemplo) ☐ Modificar os dados como queira ☐ Acessar o dado de qualquer sistema ☐ Compartilhar o dado com qualquer pessoa	☐ É simples de publicar ☐ Não precisa explicar repetitivamente que as pessoas podem fazer uso dos dados
★★	☐ Os mesmos benefícios de quem usa 1 estrela ☐ Usar softwares proprietários para processar, agregar, calcular e visualizar os dados ☐ Exportá-los em qualquer formato estruturado	☐ É fácil publicar
★★★	☐ Os mesmos benefícios de quem usa 2 estrelas ☐ Manipular os dados da forma que lhe agrada, sem estar refém de algum software em particular	☐ É ainda mais fácil de publicar Obs.: Podem ser necessários conversores ou plugins para exportar os dados do formato proprietário

Estrelas	Quem consome	Quem publica
★★★★	☐ Os mesmos benefícios de quem usa 3 estrelas ☐ Fazer marcações ☐ Reutilizar parte dos dados ☐ Reutilizar ferramentas e bibliotecas de dados existentes, mesmo que elas entendam apenas parte dos padrões usados por quem publicou ☐ Combinar os dados com outros	☐ Há controle dos itens dos dados e pode melhorar seu acesso ☐ Outros publicadores podem conectar seus dados, promovendo-os às 5 estrelas
★★★★★	☐ Descobrir mais dados vinculados enquanto consome dados ☐ Aprender sobre a classificação das 5 estrelas	☐ Torna o dado mais fácil de ser descoberto ☐ Aumenta o valor do dado ☐ A organização ganha os mesmos benefícios com a vinculação de dados que os consumidores

Podemos nos perguntar: a aplicação destes princípios está realmente sendo feita pelas instituições e estes padrões são realmente aplicados? Será que as pessoas estão de fato conectando os dados aos dados de outras pessoas?

Além dos exemplos de grandes corporações e governos (como Google, Facebook, Best Buy, governo britânico, entre outros) que já citamos aqui, podemos apresentar também o *The Linked Open Data (LOD) Project*, um projeto da comunidade de Web Semântica iniciado em 2007 por um Grupo de Interesse[15] do W3C. O objetivo do projeto foi fazer com que os dados fossem livremente disponibilizados para todos. A Figura 1.13 apresenta o último grafo gerado pelo LOD Project.

Cada círculo (nó) representa um vocabulário criado em RDF e cada seta (arco) representa uma conexão entre os vocabulários. Observe que o nó central na *cloud* apresentada equivale à DBPedia, que extraiu todos os dados da Wikipédia e disponibilizou em formato RDF para que qualquer um pudesse se conectar às suas aplicações.

15 *W3C Semantic Web Education and Outreach (SWEO) Interest Group*.

Uma empresa que fez uso do DBpedia foi a BBC. As cores dos nós representam áreas que classificam os dados (por exemplo, publicações, ciências da vida, geográficas, governo, mídias).

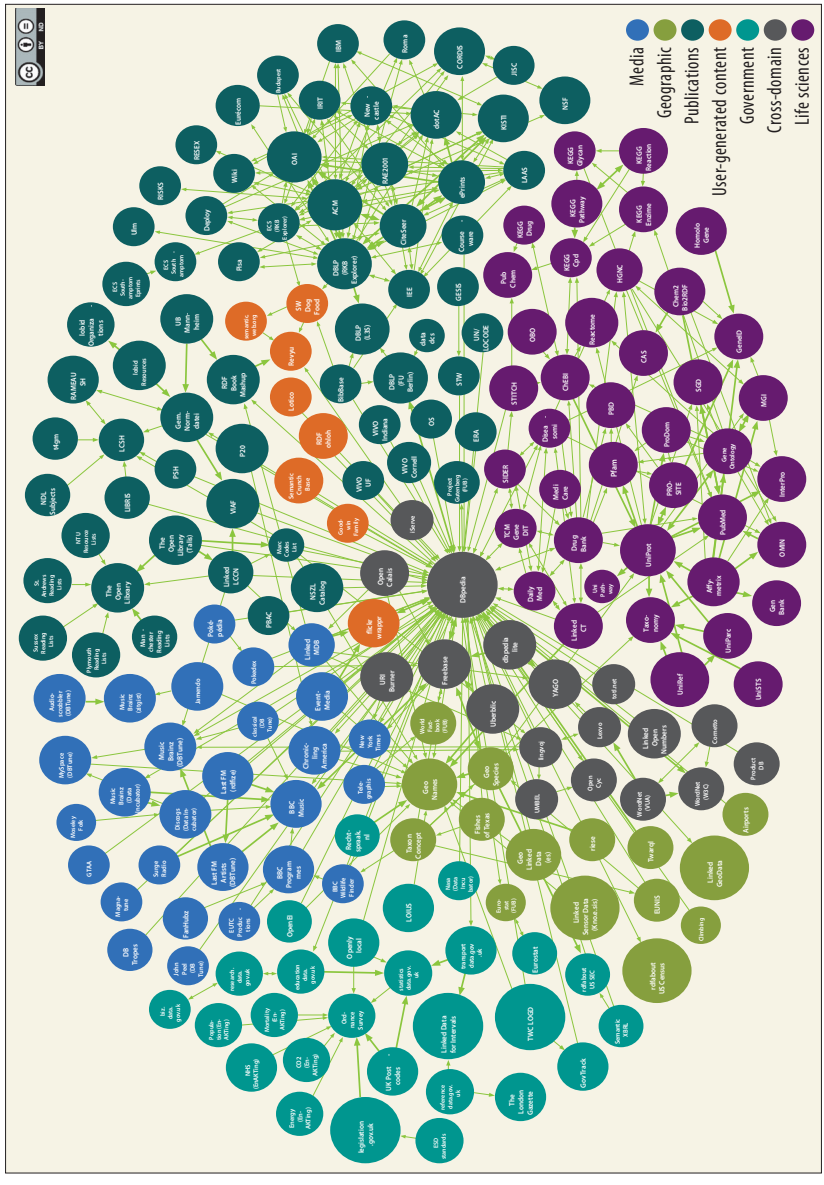

Figura 1.13 – Cloud do Projeto de Dados Conectados. Fonte: adaptado de LOD Project (2011).

Apesar de alguns desenvolvedores terem considerado a especificação de dados que utiliza o padrão RDF pouco atraente e complexa, veremos aqui que podemos fazer isso de forma sistemática. O Grupo de Trabalho do W3C de Dados Abertos Governamentais[16] definiu e publicou um conjunto de boas práticas para a publicação de dados conectados. Só para citar um exemplo (pois não é interesse deste capítulo estressar este tema), apresentamos a Figura 1.14, que descreve uma série de etapas para a geração, a publicação e a exploração de dados abertos governamentais.

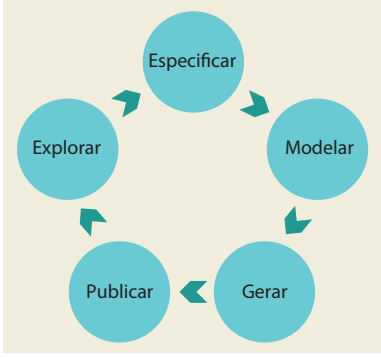

Figura 1.14 – Processo para publicação de dados abertos (Villazón-Terrazas et al., 2011).

No total foi definido um conjunto de dez boas práticas para a publicação de dados abertos conectados, sendo elas (Hyland et al., 2014):

1. **Preparar os stakeholders**: esta etapa é voltada para a formação dos usuários que irão criar e manter os Dados Abertos Conectados.
2. **Selecionar o conjunto (fonte) de dados**: etapa dedicada a definir que se pretende abrir e conectar a outros dados e disponibilizá-los para reúso.
3. **Modelar os dados**: com os stakeholders capacitados e o conjunto de dados definido, começa-se a etapa de modelagem dos Dados Conectados. Ou seja, como iremos representar os dados e como eles se relacionam com outros dados e de forma independente de aplicação.

16 Do inglês, Government Linked Data Working Group (*http://www.w3.org/2011/gld/*).

4. **Especificar a licença**: nesta etapa, a equipe/organização, responsável pelos dados que serão abertos, deve definir e especificar a licença que será usada.
5. **Nomear bons URIs**: esta etapa é o núcleo dos Dados Abertos Conectados, e a definição e o uso de boas práticas para URIs é fundamental.
6. **Usar vocabulários-padrões**: uma das melhores formas de conectar dados é por meio do reúso de vocabulários conhecidos (por exemplo, Recomendações do W3C).
7. **Converter os dados**: uma vez que a estratégia de modelagem e as boas práticas para URIs foram definidas e os vocabulários a serem reusados foram identificados, vem a etapa de converter os dados da fonte original para representação adequada aos Dados Conectados.
8. **Prover acesso aos dados**: esta etapa define quais serão as formas de acesso que seres humanos e máquinas terão aos dados.
9. **Anunciar novo conjunto de Dados Conectados**: de nada adianta conectar os dados e não anunciar para a sociedade que eles foram disponibilizados. Desta forma, esta etapa cumpre este papel de divulgação do novo *Dataset* publicado.
10. **Reconhecer a função social**: esta etapa é definida para que o responsável por publicação dos dados cumpra a função de manter os dados publicados ao longo do tempo.

Como observamos, o processo de abertura de dados envolve várias etapas e bastante critério ao publicar e anunciar para a sociedade. Abordaremos neste documento cada uma destas etapas supracitadas e mostraremos como publicar e desenvolver estas aplicações. Ou seja, estamos interessados tanto em quem publica os dados quanto em quem consome os dados.

1.5 Considerações Finais

O principal objetivo deste capítulo foi oferecer ao leitor uma visão geral sobre dados abertos, considerando todo o seu ciclo de vida. Foi também intenção deste capítulo dissertar sobre como o conceito de dados abertos pode ser ampliado e potencializado para Dados Conectados. Esperamos que as seguintes mensagens tenham sido passadas:

- Compreensão sobre a importância de dados abertos e abertura de dados.
- Entendimento sobre a integração do conceito de dados abertos com Dados Conectados.
- Aprendizado sobre os benefícios de abertura e publicação de dados.
- Compreensão sobre o ciclo de vida para publicação de dados.

CAPÍTULO 2

Estruturação de Dados e Dados Abertos Conectados

2.1 Introdução

Como descrevemos no Capítulo 1, dados abertos são dados que podem ser livremente utilizados, reutilizados e redistribuídos por qualquer pessoa – sujeitos, no máximo, à exigência de citar a fonte original e compartilhar com as mesmas licenças em que as informações foram inicialmente apresentadas. Vimos também no capítulo anterior que a disponibilização destes dados é classificada de acordo com o "Sistema de 5 Estrelas": quanto maior o número de estrelas, melhores são a visibilidade, a integração e a usabilidade destes dados tanto por pessoas quanto por máquinas.

Neste capítulo, estamos interessados em aprofundar os conhecimentos sobre os dados abertos estruturados e conectados (ou *Linked Data*) seguindo o "Sistema de 5 Estrelas". Queremos, assim, apresentar ao leitor alguns conhecimentos técnicos para utilizar modelos e padrões que viabilizam o uso dos dados estruturados disponíveis na Web para busca e navegação de forma (semi) automática com o apoio de programas de computador. Esses modelos e padrões para disponibilização dos Dados Conectados são a base para construir a

Web de Dados e, por consequência, prover um dos "pilares" necessários para consolidação da Web Semântica.

A *Web de Dados* tem como objetivo fazer um contraponto a *Web de Documentos* (vide Capítulo 1). Na Web de Documentos, os recursos como páginas *web*, figuras e vídeos estão conectados por meio de URIs[1]. Essa conexão é fundamental para fazer associações entre diferentes recursos disponíveis na Web e para consumo massivo da informação destes recursos por pessoas (agentes humanos). Assim, podemos dizer que na Web de Documentos são os recursos, e não as informações contidas nestes recursos, que estão conectados. Porém, para que um programa de computador possa utilizar as informações contidas na Web, é necessário que estejam disponíveis para ele conhecimentos adicionais que caracterizem os recursos, como eles se relacionam e quais dados estão contidos neles. Neste contexto, surge a Web de dados, na qual os recursos estão conectados também por URIs, porém informações adicionais são disponibilizadas para permitir que as máquinas possam compreender melhor os dados contidos nestes recursos e o "significado" de determinada relação entre dois ou mais recursos. Por exemplo, na Figura 2.1 (inspirada no trabalho de [Chen, 2007]), temos em 2.1(a) a Web de Documentos na qual o relacionamento entre páginas é feito por meio de *links* (i.e., URIs) entre documentos. Já em 2.1(b) temos a Web de dados que apresenta links entre páginas e informações adicionais que deixam explícita a relação entre os dados contidos nos documentos. Neste pequeno grafo de (b), os dados mais importantes desta página são apresentados como vértices (e.g., *Brasília*) enquanto o significado das relações entre estes dados *é apresentado* nas arestas (e.g., *Capital-do*).

As informações adicionais que descrevem os dados contidos nos documentos são chamadas de *metadados*, ou seja, dados sobre dados. Os metadados em conjunto com métodos de modelagem conceitual da informação, que serão apresentados no próximo capítulo, são essenciais para publicação de dados conectados. Ao longo deste capítulo,

1 Sigla em inglês (Uniform Resource Identifier) para identificador universal de recursos, os quais são referenciados na linguagem HTML pela tag `<a href = "endereço URI">`.

apresentaremos, por meio de exemplos, os conceitos técnicos necessários para criar e trabalhar com metadados e a Web de dados. Estes conceitos permitirão que o leitor tenha a capacidade de produzir dados abertos de alta qualidade utilizando uma das tecnologias recomendadas pelo W3C conhecida como RDF[2] (*Resource Description Framework*).

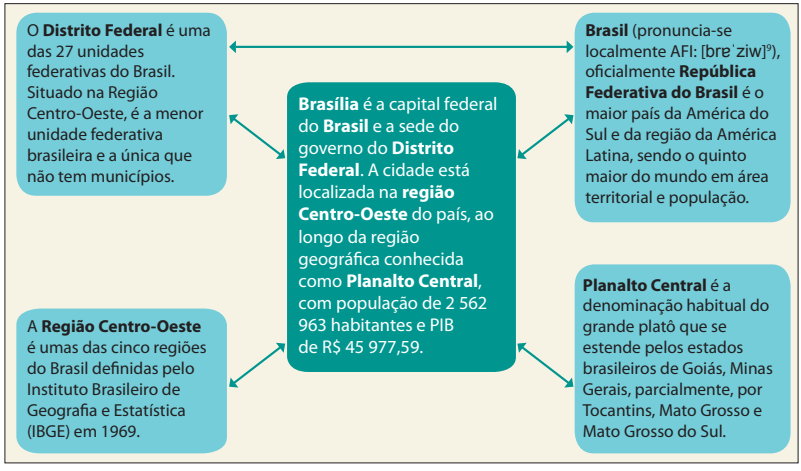

Figura 2.1 (a) – Apresenta-se um conjunto de páginas web, no qual os documentos estão conectados (web de documentos). Fonte: autores.

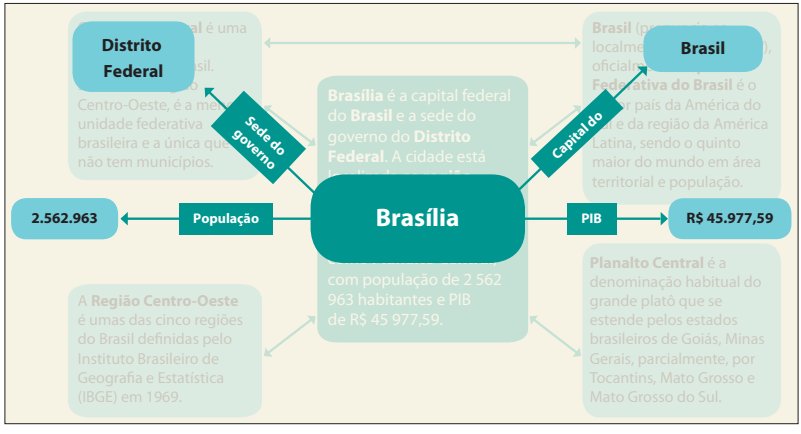

Figura 2.1 (b) – As mesmas páginas web estão conectadas a mais informações adicionais sobre os dados disponíveis nas páginas, permitindo que as máquinas possam compreender e processar melhor as relações entre as páginas e as informações contidas nelas. Fonte: autores.

2 *http://www.w3.org/2011/rdf-wg/wiki/Main_Page*

2.2 Estruturação de Dados e Dados Abertos Conectados

Os Dados Conectados, como apresentados na seção anterior, fazem da Web um banco de dados global que podemos chamar de *web* de dados. A partir deste banco de dados global, os desenvolvedores podem criar e executar consultas que buscam dados de diferentes fontes e combiná-las para prover algum tipo de informação útil. Este tipo de consulta é impossível (ou inviável) quando se pensa em tecnologias de gerenciamento de bancos de dados tradicionais (Wood et al., 2014).

Ao observar a relação entre a estruturação de dados e os Dados Conectados (não necessariamente Dados Abertos Conectados), conforme mostramos na Figura 1.10 (vide Capítulo 1), podemos compreender que um dos problemas para a sustentabilidade da Web de Dados são os Dados Estruturados. Frisamos, contudo, que neste capítulo focaremos somente os conceitos relacionados à estruturação dos dados da Web, que permitem a representação de dados conectados abertos, conforme mostra o bloco à direita da Figura 1.10.

Nesse contexto, podemos destacar diversos aspectos a partir deste ecossistema de Dados Conectados e estruturados:

- É possível utilizar diferentes formatos de estruturação de dados na Web.
- Os Dados Conectados têm uma estrutura de dados baseada em RDF.
- Existem diferentes formatos RDF para estruturação de dados.
- Os Dados Conectados podem ser considerados um subconjunto da Web Semântica.

É importante salientar que não há como ter Dados Conectados sem fazer a utilização do modelo de dados RDF[3]. No entanto, como vimos no Capítulo 1, há uma série de exigências que precisam ser

3 Veremos RDF detalhadamente neste capítulo.

atendidas para que os dados sejam estruturados no modelo RDF. No que tange aos Dados Abertos Conectados, as 5 estrelas para Dados Abertos Conectados mostram que quanto mais aumentamos o número de estrelas, melhores serão os dados abertos. Entretanto só podemos efetivamente considerar que os dados publicados são do tipo "Dados Conectados" quando estes recebem pelo menos 4 estrelas. Assim, na seção seguinte, daremos uma explicação mais detalhada sobre o sistema 5 estrelas, por meio de exemplos.

2.2.1 Padrões de Representação

A preparação dos dados para que passem de Dados Abertos para Dados Abertos Conectados pode ser feita por meio de uma série de transformações intermediárias. Conforme vimos no Capítulo 1, a Figura 2.2 apresenta o esquema de distribuição 5 estrelas para Dados Abertos, proposto por Tim Berners-Lee, em 2010 (Berners-Lee, 2010).

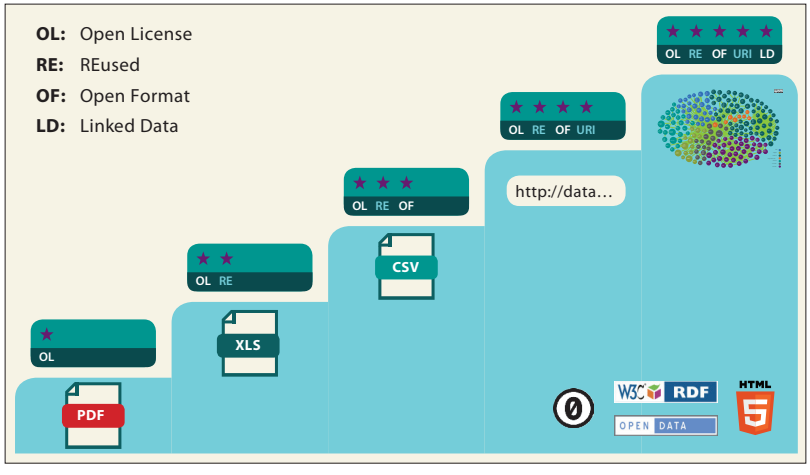

Figura 2.2 – Esquema de publicação de dados abertos de acordo com as 5 estrelas para Dados Abertos. Fonte: adaptado de Berners-Lee (2010).

Para exemplificar, vejamos o cenário disponível em (5 Start Open Data, 2012). Como poderíamos abrir os dados sobre a temperatura da cidade irlandesa Galway, conforme Tabela 2.1.

Tabela 2.1 – Exemplo para demonstrar esquema de 5 estrelas dos dados abertos

Day	Lowest Temperature (°C)
Saturday, 13 November 2010	2
Sunday, 14 November 2010	4
Monday, 15 November 2010	7

A abertura pode ocorrer de acordo com o esquema das 5 estrelas. De acordo com as estrelas, a única exigência para que o dado seja considerado aberto é que os dados sejam disponibilizados com licenças abertas. Mesmo um documento sendo publicado em PDF ou PNG, se ele utiliza uma licença aberta, então pode ser considerado um dado aberto.

Como exemplo para dados abertos com apenas 1 estrela, veja o URI a seguir. Este URI abre um arquivo PDF contendo dados de temperatura em Galway, na Irlanda. Ou seja, só há a necessidade de disponibilizar o identificador para os recursos em formato PDF. *http://5stardata.info/gtd-1.pdf*.

É importante enfatizar que há diferentes licenças sendo utilizadas pela comunidade de Dados Abertos. Como exemplos de licenças, citamos as licenças da *Open Data Commons* (Open Data Commons, 2014), como:

- *Public Domain Dedication and License (PDDL)*[4]
- *Attribution License (ODC-By)*[5]
- *Open Database License (ODC-ODbL)*[6]

Apesar de o dado do exemplo apresentado ter sido publicado com licença aberta, o dado está "trancado" em um documento PDF. Ou seja, ele não pode ser manipulado e reutilizado facilmente em outros contextos e outras situações. Isso faz com que o acesso ao dado dentro do documento fique bastante difícil, tornando complicado

4 Disponível em: <*http://opendatacommons.org/licenses/pddl/*>
5 Disponível em: <*http://opendatacommons.org/licenses/by/*>
6 Disponível em: <*http://opendatacommons.org/licenses/odbl/*>

o seu uso e acesso por máquinas (e até por pessoas, dependendo da complexidade e da quantidade dos dados disponibilizados).

A segunda estrela considera que, além de o dado estar sob licença aberta, também deverá utilizar um formato de dado estruturado. No entanto, para que o dado tenha 2 estrelas, pode-se disponibilizá-lo utilizando formatos proprietários para estruturação dos dados. Por exemplo, são considerados dados de 2 estrelas aqueles publicados com licença aberta e disponibilizados no formato de uma planilha Excel em XLS, formato proprietário cuja detentora de seus direitos é a Microsoft. No URI a seguir, temos o exemplo de dados abertos publicados com 2 estrelas para o nosso exemplo sobre a disponibilização de dados da temperatura de Galway: *http://5stardata.info/gtd-2.xls*.

Diferentemente da abertura dos dados com 1 estrela, o dado disponível em formato XLS pode ser acessado por máquina de forma menos complicada. Isso permite que os dados sejam utilizados mais facilmente tanto por pessoas, que podem utilizar, adaptar e integrar estes dados com outros dados disponíveis, quanto por máquinas que podem ler estes dados e manipulá-los mais livremente. Apesar de o dado em formato XLS facilitar o consumo dos dados, existem ainda dois problemas identificados: 1) os dados continuam trancados em um documento; 2) o acesso ao dado dentro do documento XLS necessita de software proprietário.

A terceira estrela dos dados abertos é alcançada quando se utiliza formatos abertos e não proprietários, como CSV (do inglês, *Comma-Separated Values*, significando valores separados por vírgula). Por meio da publicação de dados em formato não proprietário, a manipulação dos dados fica muito mais simplificada e sem restrições dos softwares proprietários. O exemplo da temperatura de Galway publicado em formato 3 estrelas está disponível neste link: *http://5stardata.info/gtd-3.csv*.

Os dados no documento CSV ficam disponíveis conforme demonstrado na Tabela 2.2.

Tabela 2.2 – Dados de temperatura em três dias diferentes

"Day",	"Lowest Temperature (°C)"
"Saturday, 13 November 2010",	2
"Sunday, 14 November 2010",	4
"Monday, 15 November 2010",	7

Como podemos observar, o arquivo CSV faz o armazenamento de dados tabulares em um formato *plain-text, ou* sequência de caracteres. Este tipo de documento nada mais é que um formato de dados delimitador, no qual as colunas de uma tabela são separadas pelo caractere *vírgula*, enquanto os registros são separados por *novas linhas* do arquivo. A IETF[7], órgão responsável pelo desenvolvimento e pela melhoria da Internet, disponibiliza o RFC 4180[8], uma especificação para a descrição de dados tabulares em formato CSV.

A publicação em formato CSV apresenta um avanço no esquema de distribuição de 5 Estrelas. Contudo, o formato CSV não tem a riqueza para representar detalhes importantes e não permite que desenvolvedores possam criar programas de computador mais inteligentes que extraiam múltiplos catálogos de dados, manipule-os, visualize-os e os combine de maneira flexível. Apesar disso, a grande maioria dos dados disponíveis atualmente nos diversos bancos de dados públicos (por ex., *http://data.gov.br/*) está nesse formato. Em virtude desse grande número de dados abertos em formato CSV, o W3C criou um grupo de trabalho (*CSV on the Web*[9]) com o objetivo de definir um vocabulário para metadados em CSV e métodos padronizados para:

(a) encontrar os metadados via protocolo HTTP;

(b) encontrar os dados descritos pelos metadados por meio de uma API; e

(c) mecanismos de mapeamento de CSV para outros formatos como RDF e JSON.

7 Do inglês, Internet Engineering Task Force.
8 Disponível em: <*http://tools.ietf.org/html/rfc4180*>.
9 Charter disponível em: <*http://www.w3.org/2013/05/lcsv-charter*>.

A quarta estrela está relacionada com a utilização de URIs para identificar os recursos na Web. A grande vantagem da utilização de URIs é que eles podem ser encontrados e consumidos por outras pessoas e máquinas de forma muito mais simplificada do que a disponibilização em CSV. Ou seja, o URI nos permite compartilhar cada dado que estamos publicando. Há diversos formatos para representação destes dados (por ex., Atom e RSS), mas o tipo de formato em que estamos interessados é o RDF. O link com o exemplo da temperatura contendo 4 estrelas está disponível em: *http://5stardata.info/gtd-4.html*.

Após ver este documento, você pode estar se perguntando: não é apenas mais uma página em HTML? Qual a novidade desta página? Por que a extensão do documento está em HTML e não em RDF como aconteceu com os outros formatos? Isto acontece porque documentos em formato RDF podem ser serializados e disponibilizados de diferentes formas. Uma delas é o RDFa (*Resource Description File in Attributes*), que adiciona uma série de atributos às *tags* HTML. Desta forma, em vez de disponibilizar um arquivo com a extensão RDF, pode-se disponibilizar um arquivo HTML com vários atributos de um documento RDF. Neste contexto, as informações sobre os dados ficam separadas das informações sobre como apresentar os dados. O código HTML contendo os atributos RDF são apresentados a seguir.

Fazendo uma leitura do trecho de código com RDFa, disponível em *http://www.w3.org/2013/05/lcsv-charter*, representando um dado aberto com 4 estrelas, observamos na linha 6 que há três atributos sendo utilizados na tag `<tr>`, sendo eles `rel`, `resource` e `class` (este atributo global não está trazendo nenhuma informação adicional para a descrição das temperaturas)[10].

```
01 <table border="1px">
02   <tbody><tr>
03     <th>Day</th>
04     <th>Lowest Temperature (°C)</th>
05   </tr>
```

10 Uma lista de atributos utilizados em RDFa está disponível em *http://www.w3.org/TR/2013/REC-rdfa-core-20130822/#s_syntax*.

```
<tr rel="meteo:forecast" resource="#forecast20101113" class="highlight">
   <td>
      <div about="#forecast20101113" class="highlight">
         <span property="meto:predicted" content="2010-11-13T00:00:00Z"
         datatype="xsd:dateTime" style="border: 1px dotted red;">
            Saturday, 13 November 2010</span>
      </div>
   </td>
   <td rel="meteo:temperature" class="highlight">
      <div about="#temp20101113" class="highlight">
         <span property="meto:celsius" datatype="xsd:decimal"
            style="border: 1px dotted red;">2</span>
      </div>
   </td>
</tr>
<tr rel="meteo:forecast" resource="#forecast20101114" class="highlight">
   <td>
      <div about="#forecast20101114" class="highlight">
         <span property="meto:predicted" content="2010-11-14T00:00:00Z"
            datatype="xsd:dateTime" style="border: 1px dotted red;">
               Sunday, 14 November 2010</span>
      </div>
   </td>
   <td rel="meteo:temperature" class="highlight">
      <div about="#temp20101114" class="highlight">
         <span property="meto:celsius" datatype="xsd:decimal"
            style="border: 1px dotted red;">4</span>
      </div>
   </td>
</tr>
<tr rel="meteo:forecast" resource="#forecast20101115" class="highlight">
   <td>
      <div about="#forecast20101115" class="highlight">
         <span property="meto:predicted" content="2010-11-15T00:00:00Z"
            datatype="xsd:dateTime" style="border: 1px dotted red;">
               Monday, 15 November 2010</span>
```

```
42          </div>
43       </td>
44       <td rel="meteo:temperature" class="highlight">
45          <div about="#temp20101115" class="highlight">
46             <span property="meto:celsius" datatype="xsd:decimal"
47                style="border: 1px dotted red;">7</span>
48          </div>
49       </td>
50    </tr>
51 </tbody></table>
```

Observando assim os dois atributos da tag (`<tr rel="meteo:forecast" resource="#forecast20101113">`), temos:

- `<tr>` é uma tag HTML que representa uma linha da tabela.
- `rel` é um atributo que define um relacionamento entre dois recursos. Neste exemplo, `rel` está sendo utilizado para indicar que esta linha tem um relacionamento com `meteo:forecast`. A utilização deste atributo não apresenta novas informações para apresentar um recurso no navegador, por exemplo, formatação de interface, no entanto fornece informações adicionais para máquinas poderem identificar e manipular um recurso.
- `meteo:forecast` é um *alias*, nome curto ou pseudônimo, para descrever um recurso disponível no endereço *http://purl.org/ns/meteo#forecast*. Este *alias* é criado para que, em vez de toda vez repetirmos o endereço *web* completo para cada identificador de documentos, seja possível definir um nome simples que reduza a complexidade do código e facilite a leitura. Neste caso em particular, o nome `meteo` é a abreviação do termo em inglês *meteorology* utilizado pelos criadores do documento. Se você clicar com o botão direito no link *http://5stardata.info/gtd-4.html* e pedir para exibir o código-fonte[11], encontrará a tag HTML com a identificação

11 Código-fonte disponível em view-source: <*http://5stardata.info/gtd-4.html*>.

de `meteo`, conforme a última linha do trecho de código a seguir (disponível em *http://purl.org/ns/meteo#forecast*).

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:xsd ="http://www.w3.org/2001/XMLSchema#"
xmlns:rdsf="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:meteo="http://purl.org/ns/meteo#">
```

- `resource` é utilizado para identificar o recurso do relacionamento. É importante frisar que o relacionamento definido por meio deste atributo não fornecerá um link apresentado na página, mas sim um objeto que poderá ser utilizado por máquinas, neste caso `#forecast20101113`.
- `#forecast20101113` é o objeto no qual identifica um determinado recurso. Ou seja, por meio de consultas a este recurso, os valores das colunas *Day* e *Lowest Temperature* ˚C identificadas pelas tag `<td>` podem ser buscados.

É importante frisarmos que esta estrutura de descrição está disponível em todo o documento do exemplo anterior, no qual outros atributos e recursos podem ser identificados, como `about`, `meteo:temperature`, `xsd:dateTime`, `meteo:celsius`, e assim por diante.

Toda esta descrição teve como objetivo apresentar ao leitor como podemos descrever e publicar nossos dados com 4 estrelas. Por meio deste tipo de publicação, os dados mais importantes do documento terão um URI que poderá ser utilizado por buscadores com diferentes propósitos. Por meio da disponibilização deste URI, é possível:

a) conectar e combinar estes dados com outros dados;
b) reusar estes dados em outros contextos;
c) melhorar a busca e a compreensão dos dados apresentados;
d) possibilitar inferência a partir de dados parciais;
e) permitir navegação entre documentos, entre outras atividades.

Como acabamos de descrever, esta abordagem permite que os dados possam ser conectados e combinados com outros dados para melhorar ainda mais sua identificação e compreensão. Fazer este tipo de conexão equivale a disponibilizar os seus dados com 5 estrelas. A única diferença dos dados abertos com 4 e 5 estrelas é que neste último os seus dados estarão conectados com dados de outras fontes. Segundo o mesmo exemplo, os dados de temperatura da cidade irlandesa Galway com 5 estrelas podem ser identificados neste link: *http://5stardata.info/gtd-5.html*.

Um tipo de dado que poderia ser útil a apresentar seriam informações sobre a cidade de Galway, na Irlanda, ou até mesmo sobre o país. Observemos a seguir um trecho do código HTML em 5 estrelas. Nesse trecho de código não há nenhum tipo de atributo adicional com relação ao código com 4 estrelas. Porém, ao observar os valores que estão sendo descritos nos atributos, podemos ver que há diferença.

```
<span rel="owl:sameAs" resource="http://dbpedia.org/resource/Galway"
   class="highlight"></span>
<table border="1px">
   <tbody><tr>
      <th>Day</th>
      <th>
         <div about="#temp" class="highlight">
            Lowest
            <a rel="rdfs:seeAlso" href+"http://en.wikipedia.org/wiki/Temperature"
               resource="dbpedia.org/resource/Temperature"
               class="highlight">Temperature</a>
               (<span rel="owl:sameAs"
                  resource="http://dpedia.org/resource/Celsius"
                  class="highlight">°C</span>)
         </div>
      </th>
   </tr>
```

Cada um deles é apresentado a seguir:

- `owl:sameAs` é uma propriedade pertencente à linguagem OWL (do inglês, *Web Ontology Language*[12]), que descreve que um recurso é igual a outro recurso. Neste caso, o código informa que `meteo:Place#Galway` (identificado pela linha `<div id="data" about="#Galway" typeof="meteo:Place">`) é o mesmo que Galway identificado no link *http://dbpedia.org/resource/Galway*.
- *http://dbpedia.org/resource/Galway* é o recurso que foi identificado pela propriedade `owl:sameAs`.
- `rdfs:seeAlso` é uma propriedade do RDF-S (*Resource Description Framework Schema*) que tem por objetivo fornecer informação adicional ao que está sendo descrito no documento. Neste exemplo, a propriedade `rdfs:seeAlso` contém o valor *http://en.wikipedia.org/wiki/Temperature*, indicando, assim, que informações adicionais sobre o conceito de temperatura podem ser encontradas na Wikipédia por meio deste link.
- *http://dbpedia.org/resource/Celsius* está sendo utilizado neste documento da mesma forma que *http://dbpedia.org/resource/Galway.* Ou seja, está sendo utilizado para dizer que "°C" é a mesma coisa que *http://dbpedia.org/resource/Celsius.*

Você pode estar se perguntando qual o real benefício que esta descrição adicional trouxe para o documento. Caso você queira acessar os links da DBPedia para Galway, verá que muitos dados adicionais são fornecidos sobre a cidade de Galway que antes não poderiam ser acessados automaticamente. Por meio desta identificação, é possível que máquinas possam, por exemplo, apresentar uma descrição da cidade, ou fotos turísticas da cidade, ou até mesmo disponibilizar um mapa com a localização exata da cidade.

Gostaríamos de salientar ao leitor que os exemplos apresentados no formato 4 e 5 estrelas requerem conhecimentos um pouco mais avançados sobre publicação de páginas Web, como HTML e também RDF. Não se preocupe caso não tenha acompanhado todas as

12 Veremos em detalhes no próximo capítulo.

minúcias do código, principalmente com relação ao RDF, pois essa foi sua primeira exposição a este conteúdo. Apesar de termos apresentado nesta subseção algumas das propriedades presentes no modelo RDF e também termos apresentado o formato de serialização RDFa, o modelo RDF apresenta muito mais riqueza, e vale a pena conhecê-lo em mais detalhes. É isso que faremos na próxima subseção.

2.3 Representação de Dados Conectados com o Modelo RDF

O RDF (*Resource Description Framework*) equivale a uma linguagem de representação de informação na Web, permitindo que recursos possam ser descritos formalmente e sejam acessíveis por máquinas. Segundo (Shadbolt et al., 2006), o objetivo do RDF é prover uma representação minimalista do conhecimento da Web.

A versão RDF 1.1 foi publicada em fevereiro de 2014 e estendeu a versão 1.0 proposta em 2004. Esta nova versão já está disponível (Schreiber et al., 2014). Nesta seção, abordaremos algumas das características presentes no RDF 1.1, como o modelo RDF, o seu vocabulário e suas formas de serialização. Esse conhecimento é importante para compreender a riqueza de expressividade que o RDF proporciona para descrever e representar a informação disponível na Web. Esse framework também é fundamental para entender os mecanismos necessários para criação de dados conectados de qualidade.

2.3.1 Modelo RDF

O primeiro aspecto a destacar do modelo RDF é que ele foi projetado para descrever recursos na Web. No entanto é importante lembrar que a Web é um espaço de informação no qual os itens de interesse precisam ser identificados. Assim, cada recurso tem um identificador único e global (chamado de URI) para que possa ser identificado na WWW. Na Figura 2.3 apresentamos um exemplo

de como o recurso *Oaxaca Weather Report* é identificado pelo URI *http://weather.example.com/oaxaca*. Para entender o exemplo, é preciso entender a arquitetura da Web composta de três bases fundamentais, como descritas a seguir (Jacobs et al., 2004):

1. **Recurso único**: como explicado anteriormente, os identificadores são utilizados para identificar os recursos. No exemplo a seguir, o identificador é o URI, que provê uma maneira simples e única de identificar recursos na Web, sendo caracterizado por três aspectos:

   a) **Uniformidade**: utilização de recursos tanto no mesmo contexto quanto em contextos diferenciados.

   b) **Recurso**: qualquer coisa que possa ser identificada por um URI, como um vídeo, uma imagem, um serviço, um documento, entre outra coisas.

   c) **Identificador**: informação requerida para identificar e diferenciar um determinado recurso de qualquer outro.

   Além disso, um URI[13] pode ser classificado como:

   i) URL (Uniform Resource Locator), que basicamente define um localizador/endereço para um determinado recurso a partir de um protocolo existente; e

   ii) URN (Unified Resource Name), que representa um nome para um determinado recurso, garantindo unicidade e persistência de forma global mesmo quando o recurso não está disponível. Finalmente, destacamos também o IRI (International Resource Identifier), que é uma generalização do URI. Diferente do URI, que é baseado nos caracteres ASCII[14], o IRI amplia o número de caracteres, permitindo que caracteres chineses (kanji) e japoneses (hiragana e katakana), bem como os caracteres Cirílicos e Coreanos, possam ser utilizados.

13 Para mais detalhes, vide *http://www.w3.org/2001/tag/doc/identify*.
14 Do inglês, American Standard Code for Information Interchange (ASCII)

2. **Interação**: a Web apresenta uma arquitetura cliente-servidor. A comunicação na Web acontece por meio de protocolos-padrões que permitem a troca de mensagens entre um Servidor *web* que implementa este protocolo e um Browser Cliente que envia a solicitação para o servidor. O protocolo-padrão utilizado na Web é o HTTP (*Hypertext Transfer Protocol*) e, como o próprio acrônimo diz, é um protocolo de transferência de documentos hipertextos (ex.: HTML). Por exemplo, ao utilizarmos um browser (navegador), podemos digitar o URI *http://weather.example.com*, apresentado na Figura 2.3; com isso o navegador envia uma requisição do tipo HTTP GET para o Servidor *web* e este retorna uma mensagem contendo a representação do recurso.

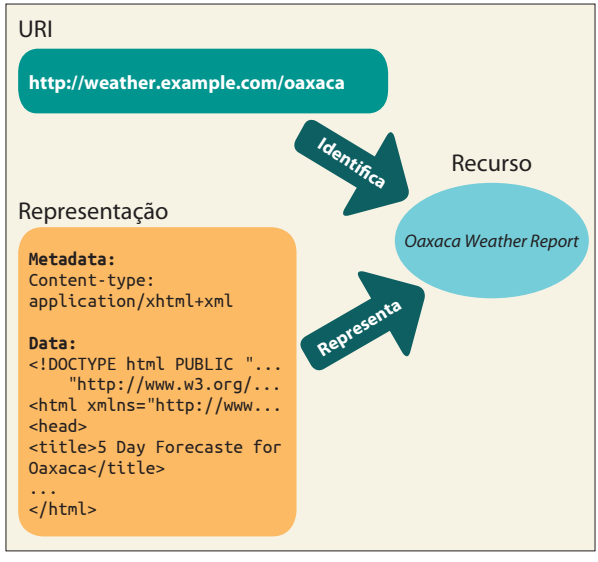

Figura 2.3 – Três bases arquiteturais da Web (Identificação, Interação e Formato). Fonte: adaptado de Jacobs (2004).

3. **Formatos**: na comunicação cliente-servidor, o servidor irá retornar para o cliente (navegador) uma representação em um determinado formato. Cada formato de representação retornado para o cliente conterá informação de Metadados e Dados.

Os metadados são os campos de cabeçalhos utilizados para identificar o formato de representação. No exemplo da Figura 2.3, o formato de representação é XHTML, que é identificado pelo `Content-Type: application/xhtml+xml`. Veremos mais a seguir que em RDF o valor do `Content-Type` será diferente, pois no modelo RDF temos diferentes formas de representação/serialização, como por exemplo Turtle (que contém `Content-Type: text/turtle`) ou JSON-LD (que contém `Content-Type: application/ld+json`).

Com este esclarecimento feito acerca das bases fundamentais da arquitetura da Web, apresentamos a Figura 2.4, que caracteriza o RDF. Podemos observar que um recurso pode ser descrito por uma coleção de propriedades denominada *descrição RDF*. Cada propriedade dentro da descrição RDF contém um tipo (*property type*) e um valor (*value*). Assim, qualquer recurso pode ser descrito com RDF e ser identificado por um URI. Inversamente, podemos pensar que um URI tem uma descrição em RDF que caracteriza um recurso disponível na Web (Iannella, 1998). A relação RDF com URI é um conceito-chave para referenciar e descrever um recurso de forma única e não ambígua.

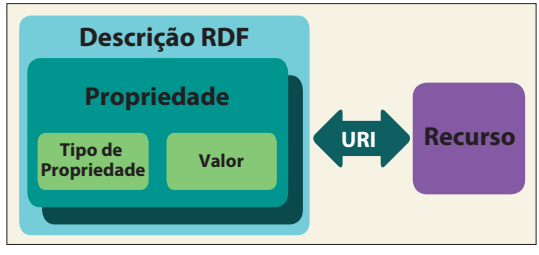

Figura 2.4 – Representação abstrata do RDF. Fonte: adaptado de Bittencourt (2009).

Além de auxiliar na descrição dos recursos disponíveis na Web de maneira mais precisa (descrição sintática), o RDF também oferece a possibilidade de descrever a relação e os significados entre diversos recursos (descrição semântica). Assim, na próxima seção, apresentaremos o conceito de triplas RDF, que permite explicitar relações entre recursos.

2.3.1.1 Triplas

Como destacamos anteriormente, o modelo RDF permite a descrição dos recursos. Para descrever a relação entre recursos o RDF oferece uma estrutura de triplas do tipo <*sujeito*> <*predicado*> <*objeto*> (Cyganiak et al., 2014). O conjunto destas estruturas em tripla é chamado de Grafo RDF. Este grafo RDF pode ser visualizado na Figura 2.5.

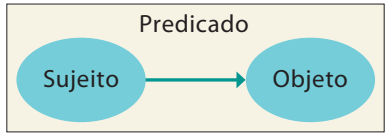

Figura 2.5 – Estrutura de triplas presente no RDF. Fonte: adaptado de Cyganiak (2014).

Essa figura apresenta um grafo dirigido e acíclico, no qual cada tripla é representada como *nó-arco-nó*. Desta forma, um grafo RDF normalmente contém múltiplas triplas que irão compor um documento RDF. Como exemplo de múltiplas triplas que são disponibilizadas em um documento RDF, temos a Figura 2.6.

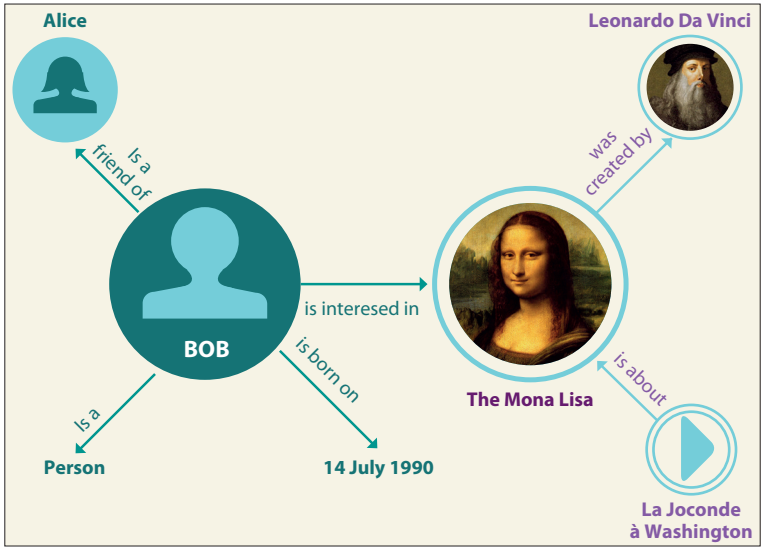

Figura 2.6 – Cenário descrevendo um conjunto de triplas e seus relacionamentos. Fonte: adaptado de Schreiber et al. (2014).

Neste exemplo existem seis triplas que representam a relação entre conceitos/recursos (i.e., vértices do grafo como *Bob*, *Person* etc). Na relação <*sujeito*> <*predicado*> <*objeto*>, o sujeito e o objeto representam dois recursos que são relacionados por um predicado. Por exemplo, <*Bob*> <*é amigo da*> <*Alice*>, <*Bob*> representa um sujeito, que é um recurso, <*Alice*> representa o objeto que é outro recurso e <*é amigo da*> representa uma propriedade que relaciona <*Bob*> à <*Alice*>. Da mesma forma <*A Mona Lisa*> <*foi criada por*> <*Leonardo Da Vinci*> representa a relação entre o sujeito <*A Mona Lisa*> e seu criador, o objeto <*Leonardo Da Vinci*>, por meio da relação/propriedade <*foi criada por*>. A riqueza desta representação permite que um computador possa interpretar os dados representados por estas triplas. Contudo, para que isso seja completamente realizado, precisamos garantir que cada um dos elementos do grafo seja representado e referenciado de maneira única. E isso pode ser realizado utilizando os IRI.

2.3.1.2 IRIs

O Identificador de Recurso Internacional (ou IRI) é o responsável por identificar de *maneira única* um recurso na Web. Como um RDF é um framework, no qual todo recurso preferencialmente deve ter um URI/IRI, então o IRI pode representar um <*sujeito*>, <*predicado*> ou <*objeto*>. Por exemplo, na Figura 2.6 a tripla <*Bob*> <*é amigo da*> <*Alice*> poderia ser representado utilizando IRIs como:

- Sujeito: <*http://example.com/Bob*>
- Predicado: <*http://example.com/is_a_friend_of*>
- Objeto: <*http://example.com/Alice*>

É importante destacarmos que, da mesma forma como apresentado na seção anterior, os dados aqui apresentados podem ser disponibilizados com 5 estrelas, fazendo com que determinado dado seja conectado com outro dado. Ou seja, poderíamos reutilizar padrões e definições previamente desenvolvidos e consolidados. Assim, no

exemplo, poderíamos fazer uso do vocabulário *FOAF (Friend of a Friend)* [*http://xmlns.com/foaf/spec/*], que tem por objetivo descrever relacionamento entre pessoas e informações na Web. Neste caso, o exemplo <*Bob*> <*é amigo da*> <*Alice*> poderia ser descrito da seguinte forma:

- Sujeito: <*http://example.com/Bob*>
- Predicado: <*http://xmlns.com/foaf/0.1/knows*>
- Objeto: <*http://example.com/Alice*>

O mesmo poderia ocorrer com os diversos recursos da Figura 2.6, como apresentado na Tabela 2.3. A título de exemplo, de acordo com a tabela, podemos ver o reúso de três vocabulários, sendo eles presentes na FOAF, DBPedia e a própria especificação do RDF.

Tabela 2.3 – Recursos em um documento RDF e recursos candidatos ao reúso

Recurso da Figura 2.6	IRI (que podem ser reusados)
<Person>	*http://xmlns.com/foaf/0.1/Person*
<is a>	*http://www.w3.org/1999/02/22-rdf-syntax-ns#type*
<The Mona Lisa>	*http://dbpedia.org/page/Mona_Lisa*
<Leonardo da Vinci>	*http://dbpedia.org/page/Leonardo_da_Vinci*
<is about>	*http://www.w3.org/1999/02/22-rdf-syntax-ns#about*

Outro vocabulário que poderia ser utilizado é o Dublin Core[15], que tem por objetivo descrever metadados, sendo bastante empregado para publicações científicas. Por exemplo, o recurso <*foi criada por*> está sendo utilizado para indicar que a obra *Mona Lisa* foi criada por *Leonardo da Vinci*. Neste contexto, o vocabulário Dublin Core contém o termo <*http://purl.org/dc/terms/created*>, que poderia ser utilizado para indicar que Leonardo da Vinci criou Mona Lisa (relação inversa a inicialmente proposta, mas que tem o mesmo significado). Neste caso, toda a tripla seria construída por meio de IRIs que seriam reusadas. Essa reutilização dá maior robustez aos dados publicados

15 Disponível em: <*dublincore.org/documents/2012/06/14/dcmi-terms/*>.

na Web, reduzindo problemas de ambiguidade e de interpretação errônea.

Com as informações apresentadas nesta seção, conseguimos utilizar IRI para descrever recursos em triplas RDF, contudo existem recursos/informações que não são associados com uma IRI. Neste caso o RDF permite utilizar literais e tipos de dados para descrever uma informação.

2.3.1.3 Literais e Tipos de Dados

Os literais (usados em conjunto com os tipos de dados) podem ser compreendidos como todos os valores identificados em um grafo RDF que não tem um IRI associado. De acordo com a Figura 2.6, podemos identificar um literal que é a data "*14 July 1990*". Este tipo de literal é um tipo de dado conhecido como *date*. Podemos também associar a expressão "*La Joconde à Washington*" a um literal que representa um texto, e seu tipo de dado é conhecido como *string*.

Reforçamos novamente aqui que uma das vantagens incorporadas com o IRI é a ampliação dos caracteres além da tabela ASCII. Ou seja, podemos então definir que "*La Joconde à Washington*" é um literal relacionado a um vídeo sobre a Mona Lisa descrito utilizando-se o idioma Francês (i.e., "fr"), enquanto ワシントンでモナリザ é um literal relacionado ao mesmo vídeo, porém utilizando o idioma japonês (ex.: "jp"). É importante frisar que os literais estão associados a tipos de dados. Como o RDF foi construído como uma camada acima do XML (vide Figura 1.7 ou Figura 1.8), todos os tipos de dados disponíveis em XML podem ser reusados em RDF. A lista de tipos de dados disponíveis em XML é apresentada na Figura 2.7. No RDF, um novo tipo de dado foi adicionado para descrever HTML, que é o `rdf:HTML`.

No entanto, não podemos esquecer que os literais só podem ser aplicados a objetos, ou seja, nunca podem descrever sujeitos ou predicados. Veremos em capítulos subsequentes que uma das recomendações para Dados Abertos Conectados é utilizar URIs para

nomear as coisas (Berners-Lee, 2006) em vez de aplicar Literais. Este tipo de visão é uma boa prática recomendada para Dados Abertos que estendeu a ideia de RDF, qualificando ainda mais os dados que estão sendo representados.

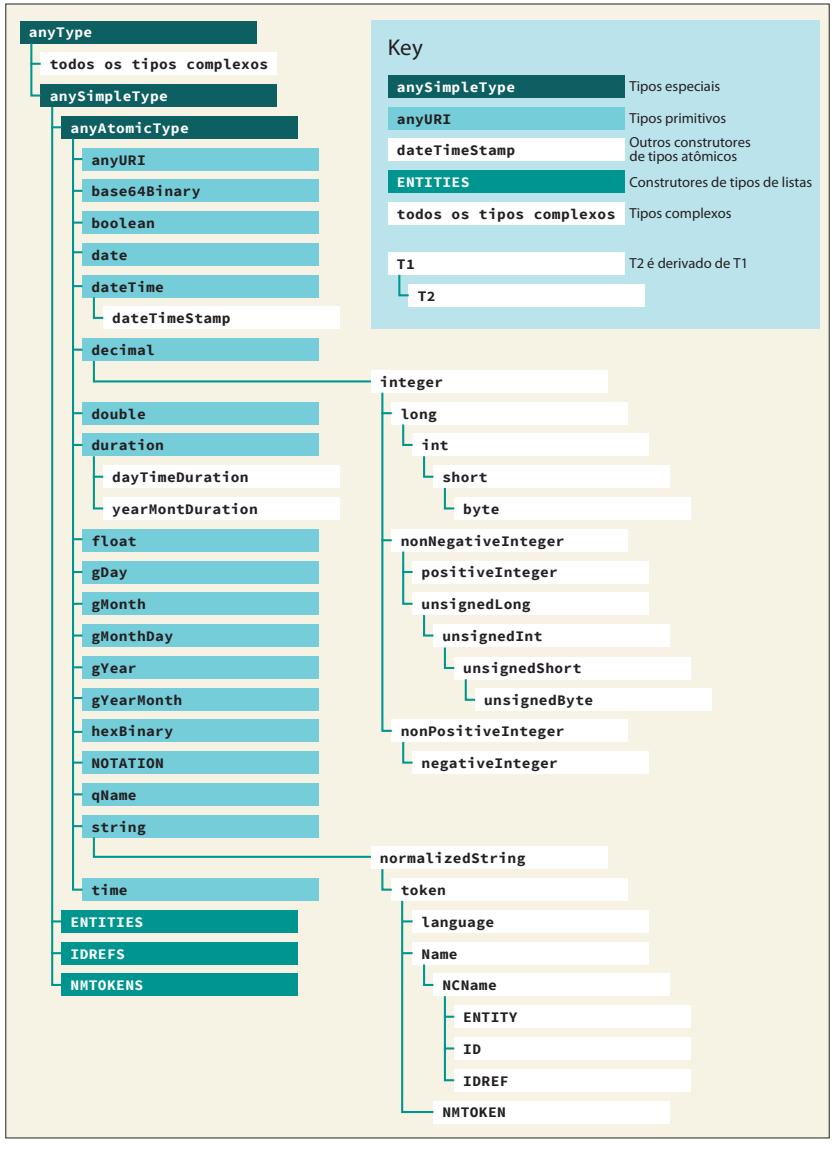

Figura 2.7 – Diagrama apresentando os tipos de dados XML e seus derivativos. Fonte: adaptado de Peterson et al. (2012).

Como resultado do reúso de diversos vocabulários, bem como dos literais, temos como Grafo RDF resultante da Figura 2.6 o grafo apresentado na Figura 2.8.

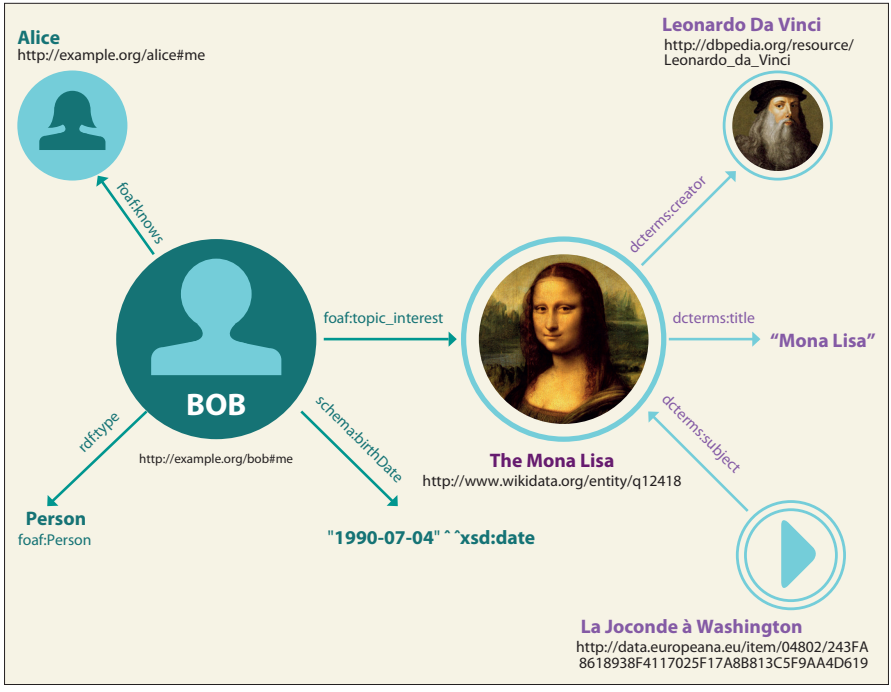

Figura 2.8 – Grafo resultando após reúso de vocabulários. Fonte: adaptado de Schreiber et al. (2014).

Observemos que em "La Joconde à Washington" foi reutilizado um recurso do catálogo de dados publicado pela Europeana[16], enquanto o literal de data permaneceu. Além disso, o literal de data para representar a data de nascimento de Bob apareceu após a data "`^^xsd:date`". O delimitador "`^^`" caracteriza que está sendo descrito um literal, enquanto o `xsd` é o alias (pseudônimo) criado, que faz referência ao *XML Schema* (vide Figura 2.4). Finalmente, "`date`" equivale ao tipo de dado que está sendo definido. Outro vocabulário reusado foi o Schema.org, que provê uma série de esquemas e tem sido usado por *engines* de buscas de empresas como Google, Microsoft, Yahoo! e outros.

16 Disponível em: <*http://www.europeana.eu/*>.

2.3.1.4 Múltiplos Grafos

Outro conceito importante que foi adicionado à nova versão do RDF é o conceito de Múltiplos Grafos. Ao se criar um documento RDF, pode-se adicionar outros grafos que se conectam ao grafo original, proporcionando assim catálogos de dados (do inglês, *Datasets*) conectados. O mais importante é que os múltiplos grafos podem ser acessados por um único IRI.

Ao observarmos o exemplo da Figura 2.6, referente à pessoa Bob, podemos ver que, pela Figura 2.8, todos os dados relacionados à Mona Lisa estão sendo reutilizados. Isso quer dizer que o Grafo da Mona Lisa poderia ser acessado de maneira independente por um único IRI, como o disponível em Wikidata, com o IRI *https://www.wikidata.org/wiki/Special:EntityData/Q12418*. Com isso, teríamos dois grafos conectados, sendo um para descrever Bob (*http:// example.org/bob*) e outro para Mona Lisa, como apresentado na Figura 2.9.

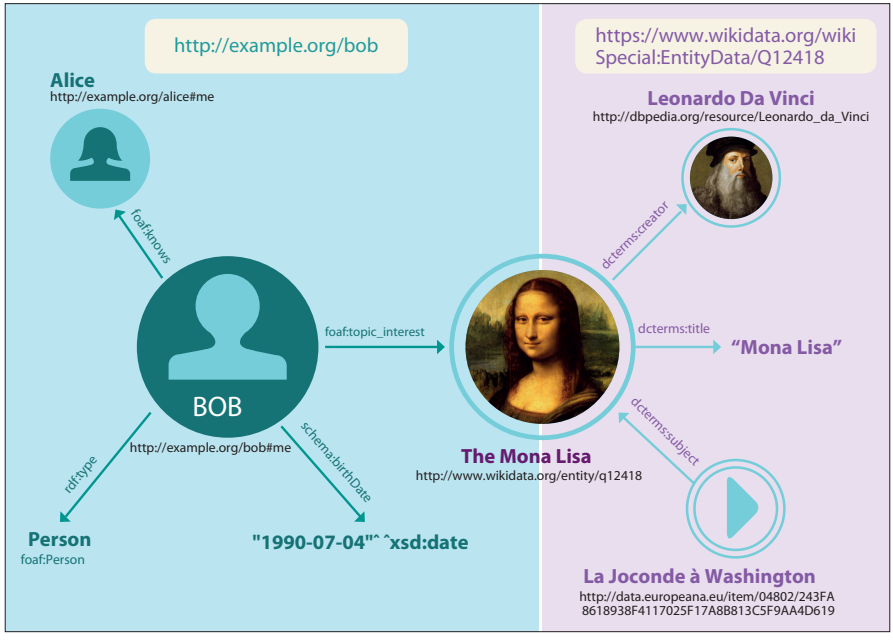

Figura 2.9 – Exemplo conectando dois grafos RDF. Fonte: adaptado de Schreiber et al. (2014).

A partir do momento que podemos conectar dois grafos RDF, podemos fazer a conexão de múltiplos grafos e termos a atribuição de um único IRI. Isso facilita a reutilização de grandes quantidades de dados presentes em grafos RDF sem a necessidade de um usuário ter o conhecimento completo deles. Além disso, facilita a extensão de grafos de maneira mais simples. Por exemplo, o grafo da Figura 2.9 poderia ser facilmente estendido e incrementado adicionando qual a licença aplicada aos dados de Bob e quem publicou os dados, como apresentado na Figura 2.10.

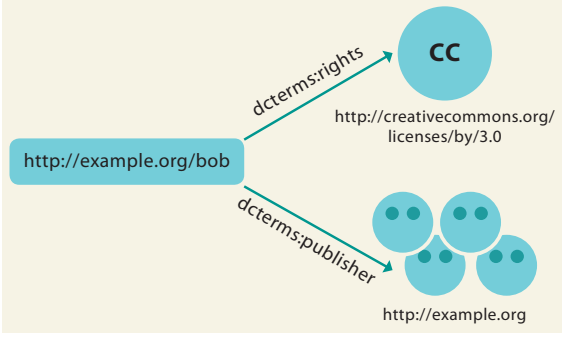

Figura 2.10 – Novos grafos relacionados ao IRI que descreve Bob. Fonte: adaptado de Schreiber et al. (2014).

De acordo com as Figuras 2.9 e 2.10, podemos observar três grafos associados, sendo um que descreve a pessoa Bob, outro para descrever Mona Lisa, outro para informar a licença relacionada à pessoa Bob e quem publicou. O grafo base[17] equivale ao primeiro grafo que conecta com os grafos subsequentes, sendo neste caso <*http://example.org/bob*>. Desta forma, pelo IRI <*http://example.org/bob*>, podemos acessar os múltiplos grafos supracitados. O código resultante deste exemplo é apresentado em seguida utilizando a linguagem TriG[18] descrevendo múltiplos grafos para o exemplo das figuras 2.9 e 2.10 (adaptado de Schreiber et al., 2014).

17 O termo correto do inglês é *Graph Name*. Para mais detalhes, vide *http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#dfn-graph-name*.

18 TriG é uma linguagem recomendada pelo W3C em fevereiro de 2014 e foi criada com o propósito de descrever RDF Datasets. Para mais detalhes, vide *http://www.w3.org/TR/2014/REC-trig-20140225/*.

```
01 BASE <http://example.org/>
02 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
03 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
04 PREFIX schema: <http://schema.org>
05 PREFIX dcterms: <http://purls.org/dc/terms/>
06 PREFIX wd: <http://www.wikidata.org/entity/>
07
08 GRAPH: <http://example.org/bob>
09   {
10      <bob#me>
11       a foaf:Person ;
12         foaf:knows <alice#me> ;
13         schema:birthDate "1990-07-04"^^xsd:date ;
14         foaf:topic_interest wd:Q12418 .
15   }
16
17 GRAPH: <https://www.wikidata.org/wiki/Special:EntityData/Q12418>
18   {
19      wd:Q12418
20         dcterms:title "Mona Lisa" ;
21         dcterms:creator <http: dbpedia.org/resource/Leonardo_da_Vinci> .
22
23         <http://data.europeana.eu/item/04802/
            243FA8618938F4117025F17A8b813C5F9AA4D619>
24            dcterms:subject wd:Q12418 .
25   }
26
27 <http://example.org/bob>
28         dcterms:publisher <http://example.org> ;
29         dcterms:rights <http://creativecommons.org/licences/by/3.0/> .
```

2.3.2 Esquema RDF (RDF-S)

Esta subseção tem por objetivo apresentar o esquema RDF, que é uma extensão do RDF conhecida como *RDF Schema* ou RDF-S, que possibilita descrição semântica. O RDF-S é um vocabulário para modelagem de dados que amplia a expressividade do RDF para prover mecanismos de descrição de taxonomias entre recursos

e suas propriedades. Ou seja, o RDF-S permite descrever grupos de recursos (também conhecidos como classes) e suas relações utilizando o conceito de triplas apresentado na seção anterior. A Figura 2.11 apresenta os principais elementos do RDF-S. Apesar de a Figura 2.11 ter sido extraída da especificação do RDF 1.0, que foi recomendação do W3C em março de 2000 (Brickley et al., 2000), os elementos descritos na figura também estão presentes na versão 1.1 de fevereiro de 2014 (Brickley et al., 2014).

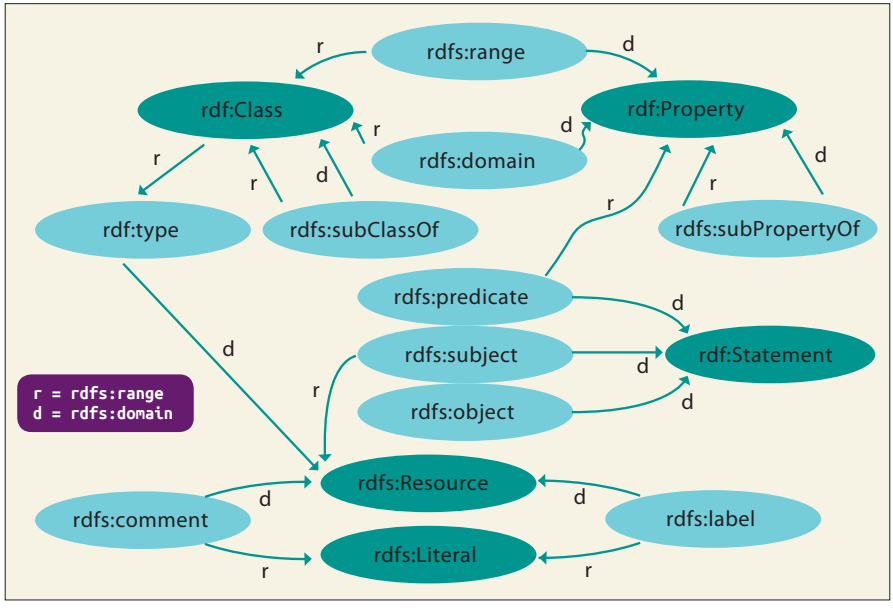

Figura 2.11 – Diagrama do RDF-S. Fonte: adaptado de Brickley et al. (2000).

Assim como o *XML Schema* é baseado no XML, o mesmo ocorre com *o RDF Schema*, que é baseado no RDF. Isso implica que o RDF-S tem um IRI para cada recurso e que também apresenta uma estrutura de triplas <*sujeito*> <*predicado*> <*objeto*>. Gostaríamos de destacar dois conceitos básicos presentes em RDF-S:

- **Classe**: este conceito é utilizado para descrever recursos em um documento RDF. Quando temos interesse em descrever recursos, podemos simplesmente utilizar o `rdfs:Resource`, que é uma

instância de `rdfs:Class`. Outros dois tipos de classe importantes são `rdfs:Literal` e `rdfs:DateType` para definir tipos primitivos e derivativos. Toda classe RDF tem um IRI associado, e podemos definir propriedades para as classes RDF.

- **Propriedade**: o papel de uma propriedade é definir relações entre sujeitos e objetos. Um tipo de propriedade importante equivale ao `rdf:type`. Ao dizermos que uma <*Classe A*> é uma instância da <*Classe B*>, usamos o `rdf:type`. Ou seja, <*Classe A*> `rdf:type` <*Classe B*>. Por exemplo, na Figura 2.9, para indicar que Bob é do tipo Pessoa, do vocabulário FOAF, fazemos:

  ```
  Bob  rdf:type  foaf:Person
  ```

  Outros dois tipos de propriedade bastante usados são `rdfs:domain` (responsável por indicar o sujeito da relação) e `rdfs:range` (responsável por indicar o objeto da relação). Quando dizemos <*Propriedade P1*> `rdfs:domain` <*Classe C1*>, quer dizer que o sujeito da relação é a classe C. O mesmo ocorre para indicar o objeto, como <*Propriedade P2*> `rdfs:range` <*Classe C2*>. Por exemplo, na Figura 2.9, na qual temos a relação *Bob* `foaf:knows` *Alice*, estamos abstratamente dizendo que uma pessoa conhece outra pessoa. Neste caso, temos que a propriedade `foaf:knows` contém tanto o domínio (i.e., `rdfs:domain`) quanto a imagem (i.e., `rdfs:range`) como `foaf:Person`.

Ressaltamos a importância do RDF-S que é usado para representação e modelagem de conceitos. No próximo capítulo iremos tratar sobre modelagem utilizando os diversos conceitos já apresentados até o momento.

2.3.3 Formatos de Serialização em RDF 1.1

Até o momento, abordamos características importantes do Modelo RDF, bem como conceitos fundamentais do RDF Schema, que determina a semântica por trás do Modelo RDF. Outro aspecto fundamental para

RDF são as formas que eles podem ser apresentados ou serializados. Esta é de fato uma grande diferença presente no RDF 1.0 e em sua evolução para o RDF 1.1, pois o número de formatos de serialização foi bastante ampliado, como pode ser visto na Figura 2.12 (Wood, 2014).

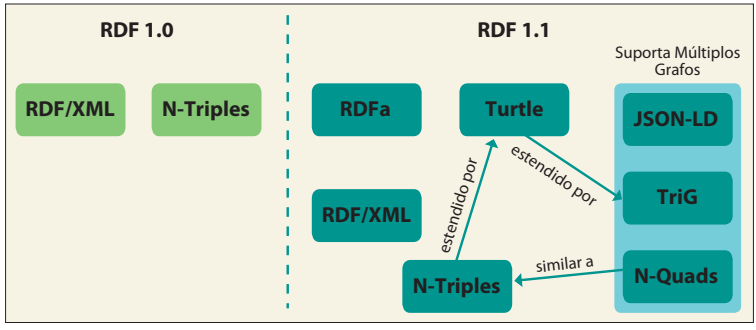

Figura 2.12 – Formatos de Serialização disponíveis nas versões 1.0 e 1.1 do RDF. Fonte: adaptado de Wood (2014).

Apesar de termos abordado o RDFa e o Turtle em seções anteriores, nas próximas subseções iremos brevemente abordar cada um dos formatos de serialização.

2.3.3.1 RDFa

O RDFa (*Resource Description Framework in Attributes*) tem por objetivo embutir código RDF em estruturas HTML e XML (Herman, 2015). Isto é feito por meio da inclusão de significado via atributos dos elementos. É por esta razão que RDFa é considerada o RDF em Atributos. A grande vantagem de utilização do RDFa é que máquinas de buscas podem melhorar seus resultados aumentando a precisão sobre o real significado de determinado documento. Ou seja, as máquinas de buscas podem agregar os dados de um documento com dados de outro documento, enriquecendo assim os resultados de buscas. No trecho de código a seguir em RDFa (adaptado de Schreiber et al., 2014) podemos ver um código HTML com atributos RDF embutidos que representa o grafo apresentado na Figura 2.9.

```
<body prefix="foaf: http://xmlns.com/fcaf/0.1/
             schema: http://schema.org/
             dcterms: http://purl.org/dc/terms/">
    <div resource="http:example.org/bob#me" typeof="foaf:Person">
      <p>
        Bob knows <a property="foaf:knows"
                     href="http://example.org/alice#me">Alice</a>
        and was born on the <time property="schema:birthDate">1990-07-04</time>.
      </p>
      <p>
        Bob is interested in <span property="foaf:topic_interest"
        resource="http://www.wikidata.org/entity/Q12418">the Mona Lisa</span>.
      </p>
    </div>
    <div resource="http://www.wikidata.org/entity/Q12418">
      <p>
        The <span property="dcterms:title">Mona Lisa</span> was painted by
        <a property="dcterms:creator" href="http://dbpedia.org/resource/
         Leonardo_da_Vinci">Leonardo da Vinci</a>
        and is the subject of the video
        <a href="http://data.europeana.eu/item/04802/
                        243FA8618938F4117025F17A8B813C5F9AA4D619">
         'La Joconde à Washington'</a>
      </p>
    </div>
    <div resource="http://data.europeana.eu/item/04802/
                          243FA8618938F4117025F17A8B813C5F9AA4D619">
      <link property="dcterms:subject" href="http://www.wikidata.org/
                                                 entity/Q12418"/>
    </div>
  </body>
```

Podemos observar que no código HTML aparecem quatro atributos com o objetivo de descrever código RDF, sendo eles:

- `prefix`, que tem por objetivo descrever os vocabulários que estão sendo reusados no documento HTML. Neste caso, temos o reúso de três vocabulários bastante conhecidos por profissionais da área (FOAF, Schema.org e Dublin Core).

- `resource`, cujo objetivo é descrever um determinado recurso, da mesma forma que foi explicado no RDF Esquema, para a classe `rdfs:Resource`. Por exemplo, na linha 4, temos a descrição do recurso *http://example.org/bob#me*.
- `property`, cujo objetivo é descrever uma propriedade. Semelhante ao atributo `resource`, o atributo `property` é similar à propriedade `rdf:Property`. Como explicamos anteriormente (vide Subseção 1.3.2), uma propriedade tem o objetivo de relacionar dois elementos, ou seja, relacionar um sujeito a um objeto. Isso quer dizer que uma propriedade irá relacionar dois recursos. Podemos observar que, como consequência disso, todas as propriedades apresentadas no código aparecem dentro da estrutura de um elemento `<div>`. Isso quer dizer, por exemplo, que das linhas 4 a 13 há três triplas relacionadas a Bob.

```
http://example.org/bob#me  foaf:knows           Alice
http://example.org/bob#me  schema:birthDate     1990-07-04
http://example.org/bob#me  foaf:topic_interest  the Mona Lisa
```

- `typeof`, que equivale ao atributo para representar o elemento `rdf:type` (i.e., tem o mesmo objetivo do `rdf:type`). Por exemplo, na linha 4, temos:

```
http://example.org/bob#me    rdf:type    foaf:Person
```

2.3.3.2 RDF/XML

RDF/XML (Gandon; Schreiber, 2014) foi a primeira serialização feita para RDF e pode ser claramente reconhecida pelo "bolo de noiva" (i.e., arquitetura em camadas) apresentado na Figura 1.7. Este formato tem uma sintaxe XML para os grafos RDF. O trecho de código a seguir apresenta o mesmo código RDFa, porém em RDF/XML.

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <rdf:RDF
03         xmlns:dcterms:"http://purl.org/dc/terms/"
04         xmlns:dfoaf:"http://xmlns.com/foaf/0.1/"
```

```
05          xmlns:rdf:"http://www.w3.org/1999/02/22-rdf-syntax-ns#"
06          xmlns:schema:"http://schema.org/">
07       <rdf:Description rdf:about="http://example.org/bob#me">
08          <rdf:type rdf:resource="http:xmlns.com/foaf/0.1/Person"/>
09          <schema:birthDate rdf:datatype="http:www.w3.org/2001/
             XMLSchema#date">1990-07-04<schema:birthDate>
10          <foaf:knows rdf:resource="http://example.org/alice#me"/>
11          <foaf:topic_interest
             rdf:resource="http://www/wikidata.org/entity/Q12418">
12       </rdf:Description>
13       <rdf:Description rdf:about="http://www/wikidata.org/entity/Q12418">
14          <dcterms:title>Mona Lisa</dcterms:title>
15          <dcterms:creator
             rdf:resource="http://dbpedia.org/resource/Leonardo_da_Vinci"/>
16       </rdf:Description>
17       <rdf:Description rdf:about="http://data.europeana.eu/item/04802/
          243FA861893F4117025F117025F17A8B813C5F9AA4D619">
18          <dcterms:subject rdf:resource="http://www.wikidata.org/entity/Q12418"/>
19       </rdf:Description>
20 </rdf:RDF>
```

Podemos observar que a primeira linha do código já apresenta o elemento `<?xml>`. Diferente do RDFa, que tem todos os seus elementos dentro da estrutura do HTML, o RDF/XML tem toda a sua estrutura dentro do XML e da tag `<rdf:RDF>`. Já o elemento `rdf:Description` é utilizado para identificar sujeitos. Ao observarmos o código, o primeiro sujeito apresentado (vide Linha 7) contém quatro subelementos. Isso quer dizer que cada subelemento apresentado entre as linhas 8 e 11 representam uma tripla combinada com o sujeito descrito na linha 7. Outra característica importante que apresentamos no RDF/XML é que o sujeito, apesar de ser identificado pelo elemento `rdf:Description`, é descrito pelo `rdf:about`.

Diferentemente do código em RDFa, os tipos de dados reusados do esquema XML foram explicitamente definidos para identificar a semântica do `schema:birthDate`.

2.3.3.3 JSON-LD

JSON-LD (Sporny et al., 2014) é um dos mais recentes formatos de serialização e surgiu como uma extensão da proposta do JSON, com o objetivo de transformar código JSON para RDF com o mínimo de esforço possível. Este formato é bastante intuitivo para programadores já familiarizados com a sintaxe JSON. O trecho de código a seguir apresenta um código em JSON-LD (adaptado de Schreiber et al., 2014).

```
01 {
02      "@context": "example-context.json",
03      "@id": "http://example.org/bob#me",
04      "@type": "Person",
05      "birthdate": "1990-07-04",
06      "knows": "http://example.org/alice#me",
07      "interest": {
08         "@id": "http://www.wikidata.org/entity/Q12418",
09         "title": "Mona Lisa",
10         "subject_of": "http://data.europeana/item/04802/
                                243FA8618938F41177025F17A8B813C5F9AA4D619",
11         "creator": "http://dbpedia.org/resource/Leonardo_da_Vinci"
12      }
13 }
```

Podemos observar que o código descreve o recurso Bob, identificado pelo IRI *http://example.org/bob#me*, por meio da chave `@id`, descrita na linha 3, e o tipo de recurso sendo descrito na linha 4. As linhas 5, 6 e 7 definem objetos com os quais o sujeito Bob se relaciona. Da mesma forma que ocorre com os outros formatos de serialização, podemos identificar as triplas no JSON-LD, como exemplificadas a seguir:

```
http://example.org/bob#me  birthDate  1990-07-04
http://example.org/bob#me  knows      http://example.org/alice#me
http://example.org/bob#me  interest   http://www.wikidata.org/entity/Q12418
```

Dois pontos que devemos destacar desse código JSON-LD são:

i) Na linha 7, podemos identificar outro recurso por meio do qual o sujeito Bob se relaciona. Esse recurso equivale a outro grafo contendo uma estrutura própria e sendo identificado pelo IRI descrito na linha 8.

ii) Não é possível identificar o esquema e a semântica dos recursos. Isso acontece porque o contexto no qual os recursos estão inseridos fica em outro documento. Este contexto é identificado pelo objeto `@context`, descrito na linha 2. Logo, o código que descreve o contexto é apresentado a seguir (adaptado de Schreiber et al., 2014).

```
01 {
02      "@context": {
03        "foaf": "http://xmls.com/foaf/0.1/",
04        "Person": "foaf:Person"
05        "interest": "foaf:topic_interest",
06        "knows": {
07           "@id": "foaf:knows",
08           "@type": "@id"
09       },
10        "birthdate": {
11           "@id": "http://schema.org/birthDate",
12           "@type": "http://www.w3.org/2001/XMLSchema#date"
13        },
14        "dcterms": "http://purl.org/dc/terms/",
15        "title": "dcterms:title",
16        "creator": {
17           "@id": "dcterms:creator",
18           "@type": "@id"
19        },
20        "subject_of": {
21           "@reverse": "dcterms:subject",
22           "@type": "@id"
23        }
24      }
25 }
```

É por meio deste contexto que é possível fazer o mapeamento do código JSON-LD, apresentado no início desta subseção, com grafo apresentado na Figura 2.9.

2.3.3.4 N-Triples

N-Triples faz parte da família de formatos Turtle. N-Triples é o formato de serialização mais simples e intuitivo que existe (Carothers et al., 2014). Como a principal característica do RDF é ter uma estrutura em triplas <*sujeito*> <*predicado*> <*objeto*>, N-Triples simplesmente estrutura o código em forma de triplas (vide trecho de código a seguir). Isso quer dizer que cada linha de um código N-Triples equivale a uma tripla. Por exemplo, o código a seguir tem sete linhas, representando assim sete triplas (adaptado de Schreiber et al., 2014).

```
01 <http://example.org/bob#me>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://xmlns.com/foaf/0.1/Person> .
02 <http://example.org/bob#me>
   <http://xmlns.com/foaf/0.1/knows>
   <http://example.org/alice#me> .
03 <http://example.org/bob#me>
   <http://schema.org/birthDate> "1990-07-04"^^
   <http://www.w3.org/2001/XMLSchema#date> .
04 <http://example.org/bob#me>
   <http://xmlns.com/foaf/0.1/topic_interest>
   <http://www.wikidata.org/entity/Q12418> .
05 <http://www.wikidata.org/entity/Q12418>
   <http:purl.org/dc/terms/title> "Mona Lisa" .
06 <http://www.wikidata.org/entity/Q12418>
   <http:purl.org/dc/terms/creator>
   <http://dbpedia.org/resource/Leonardo_da_Vinci> .
07 <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619>
   <http:purl.org/dc/terms/subject>
```

Podemos ver que ao final de cada linha há um ponto (".") sendo usado exatamente para identificar o fim de uma linha. Outros aspectos a observarmos é que todos os IRIs são absolutos, ou seja, têm a identificação do recurso completo. Devido a sua simplicidade, não é possível definir prefixos e usar IRIs relativos[19].

2.3.3.5 Turtle

Com o objetivo de ampliar as possibilidades de descrição de um documento N-Triples, o formato Turtle foi criado, podendo assim descrever prefixos e IRIs relativos na estrutura do documento (Prud'hommeaux et al., 2014). Da mesma forma que N-Triples, o formato Turtle é bastante simples e ainda mais fácil de ler. O código a seguir apresenta a estrutura em Turtle (adaptado de Schreiber et al., 2014).

```
01 BASE <http://example.org/>
02 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
03 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
04 PREFIX schema: <http://schema.org/>
05 PREFIX dcterms: <http://purl.org/dc/terms/>
06 PREFIX wd: <http://www.wikidata.org/entity/>
07
08 <bob#me>
09      a foaf:Person ;
10      foaf:knows <alice#me> ;
11      schema:birthDate "1990-07-04"^^xsd:date ;
12      foaf:topic_interest wd:Q12418
13
14 wd:Q12418
15      dcterms:title "Mona Lisa" ;
16      dcterms:creator <http://dbpedia.org/reource/Leornardo_da_Vinci> .
17
18 <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4d619>
19      dcterms:subject wd:Q12418 .
```

19 IRIs relativos definem um recurso dentro de um context. Por exemplo, o IRI relativo de *http://example.org/bob#me é bob#me*.

As seis primeiras linhas do código mostram os IRIs que podem ser definidos como prefixos e IRI base do documento, característica esta não permitida no N-Triples. Podemos ainda observar que as linhas 8, 14 e 18 apresentam sujeitos com seus predicados e objetos logo abaixo deles. Este tipo de organização e endentação torna bastante intuitiva a leitura do documento, facilitando assim a identificação das triplas RDF.

Objetivando ainda mais a simplificação da leitura de documentos RDF, a linha 9 apresenta mais uma característica do formato Turtle. A primeira tripla descrita para Bob é `bob#me a foaf:Person`. O elemento que é responsável por relacionar o sujeito ao predicado deste exemplo é o token "`a`". Este token "`a`" tem a mesma semântica da propriedade `rdf:type` e é usado para dizer que `bob#me` é do tipo `foaf:Person`. Você pode estar se perguntando por que o "a". A letra "a" representa um artigo da língua inglesa, sendo traduzido para português como "um" ou "uma". Ou seja, ao lermos a tripla descrita anteriormente na linha inglesa seria *Bob is a person* (no português: *Bob é uma pessoa*). Logo, o token "`a`" foi adicionado para tornar o formato Turtle ainda mais intuitivo para o ser humano.

2.3.3.6 TriG

O formato TriG é uma extensão do formato Turtle, ou seja, herda a mesma simplicidade e facilidade de leitura (Carothers et al., 2014). Este formato foi criado para representar múltiplos grafos que foram adicionados a partir da especificação 1.1 do RDF. Um exemplo de código TriG para representação de múltiplos grafos foi apresentado na Seção 2.3.1.4.

Observemos que a única diferença entre o código Turtle e o código TriG é que os sujeitos descritos são encapsulados dentro de uma palavra-chave chamada `GRAPH`. Isso quer dizer que o conjunto de triplas dentro do elemento `GRAPH` representa um catálogo de dados (ou *dataset*). No código da Seção 2.3.1.4, são descritos três grafos, sendo dois grafos com a identificação (*named graph*) e um sem (aplicando *blank nodes*).

Na estrutura da linguagem[20], é obrigatória a descrição de pelo menos dois grafos, sendo um necessariamente descrito como *blank node*, ou seja, sem IRI para identificar o grafo. É importante destacar que um documento Turtle também é considerado um documento TriG.

2.3.3.7 N-Quads

Finalmente, o formato N-Quads é utilizado para permitir o intercâmbio de catálogo de dados (Carothers, 2014). Como você pode ver no código a seguir em N-Quads (adaptado de Schreiber et al., 2014), ele é uma extensão da N-Triples e apresenta em cada linha a identificação de uma tripla.

```
01 <http://example.org/bob#me>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://xmlns.com/foaf/0.1/Person> .
02 <http://example.org/bob#me>
   <http://xmlns.com/foaf/0.1/knows>
   <http://example.org/alice#me> .
03 <http://example.org/bob#me>
   <http://schema.org/birthDate> "1990-07-04"^^
   <http://www.w3.org/2001/XMLSchema#date> .
04 <http://example.org/bob#me>
   <http://xmlns.com/foaf/0.1/topic_interest>
   <http://www.wikidata.org/entity/Q12418> .
05 <http://www.wikidata.org/entity/Q12418>
   <http:purl.org/dc/terms/title> "Mona Lisa".
06 <http://www.wikidata.org/entity/Q12418>
   <http:purl.org/dc/terms/creator>
   <http://dbpedia.org/resource/Leonardo_da_Vinci>.
07 <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619>
   <http:purl.org/dc/terms/subject> .
08 <http://example.org/bob#me>
   <http:purl.org/dc/terms/publisher>
   <http://example.org> .
09 <http://example.org/bob#me>
   <http:purl.org/dc/terms/rights>
   <http://creativecommons.org/licences/by/3.0/> .
```

20 Alguns consideram TriG uma linguagem.

Na Tabela 2.4 apresentamos um resumo dos formatos de serialização, bem como a estrutura do código para cada formato e o propósito de uso de cada um deles.

Tabela 2.4 – Formatos de serialização e suas características de uso

Serialização RDF	Tipo de código	Quando usar
RDFa	Código RDF embutido em HTML	SEO (Search Engine Optimization)
RDF/XML	Código RDF com estrutura em XML	Aplicações que usam estruturas em XML
JSON-LD	Código RDF com estrutura JSON	Aplicações que usam JSON
N-Triples	Código RDF com estrutura de triplas	Processamento e intercâmbio de Big Data em RDF
Turtle	Código RDF para facilitar a leitura humana	Processamento e intercâmbio de Big Data em RDF
TriG	Código com estrutura Turtle	Representação de múltiplos grafos
N-Quads	Código RDF com estrutura de triplas	Processamento e intercâmbio de grandes catálogos de dados

2.4 Exemplos de Sucesso de Dados Conectados

Nesta seção, iremos apresentar cinco casos de sucesso de utilização de Dados Abertos Conectados, sendo três deles relacionados a aplicações empresariais e dois a aplicações governamentais.

2.4.1 Tornando a Web de Dados Possível: DBPedia

Apesar de este exemplo ser um caso mais técnico, consideramos de fundamental importância dissertarmos sobre ele, dado o impacto da DBPedia na evolução da Web em direção a Web de Dados.

Este caso de sucesso é sem dúvida o responsável pela ampliação na força-tarefa que estabeleceu as bases para o advento da Web de Dados. A DBPedia tem uma estrutura em RDF/OWL que equivale ao nó central da Web de Dados, como apresentamos na Figura 1.13.

Foi por meio da comunidade de Web Semântica e da base de dados da DBPedia que a Web de Dados começou a virar uma realidade. Projeto iniciado em 2007 (*The Linked Open Data Project*), a DBPedia contém atualmente mais de 30 milhões de recursos descritos em 120 idiomas. Só na língua inglesa, descreve 832 mil pessoas, 639 mil lugares, 372 mil trabalhos criativos (i.e., músicas, filmes, jogos), 209 mil organizações, 226 mil espécies e 5,6 mil doenças. Na língua portuguesa, a DBPedia descreve 736 mil recursos, sendo esta uma das 12 línguas mais povoadas na ontologia da DBPedia. Além disso, a DBPedia reúsa recursos de quase 40 diferentes catálogos de dados e foi conectada por meio de mais de 40 milhões de links a uma centena de catálogos de dados (Lehmann et al., 2014). A Figura 2.13 apresenta a arquitetura atual da DBPedia.

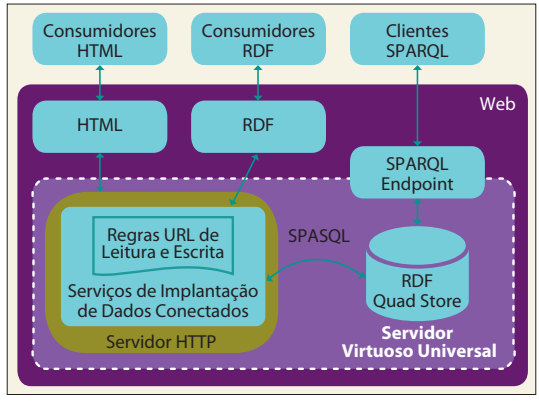

Figura 2.13 – Arquitetura do DBPEDIA. Fonte: adaptado de Bizer (2009).

Como podemos observar, a arquitetura dá suporte a três tipos de ator diferentes, sendo dois deles para consumo utilizando diretamente as tecnologias e os padrões para Dados Conectados. Para cada tipo de usuário, faz-se uso de uma tecnologia diferente. Ou seja, usuários podem acessar via HTML, via RDF (por exemplo, *Open Link Data Explorer*[21]) e via *SPARQL Endpoint*[22]. Toda a comunicação

21 Disponível em: <*http://ode.openlinksw.com*>.

22 SPARQL Endpoints são mecanismos de acesso por meio de consultas à base de dados utilizando a linguagem SPARQL. Veremos em capítulos posteriores sobre a linguagem SPARQL e como fazer consultas via Endpoints.

cliente-servidor ocorre, pelo lado cliente, por meio dos mecanismos supracitados, e do lado servidor, por meio do servidor Virtuoso[23].

2.4.2 Consumindo Dados Eficientemente: BBC

Considerada um dos exemplos de Dados Conectados mais famosos, a BBC está na Web desde 1994 e vem trabalhando com enriquecimento de páginas *web* há mais de dez anos, inicialmente com a programação de rádio e depois expandindo para TV, música, gastronomia, entre outros meios. Por volta de meia década, a BBC começou a publicar dados sobre as notícias de música, reusando o vocabulário da *MusicBrainz*[24]. Atualmente, a BBC faz o reúso de diversos vocabulários e do catálogo de dados.

A arquitetura de Dados Conectados da BBC pode ser vista na Figura 2.14.

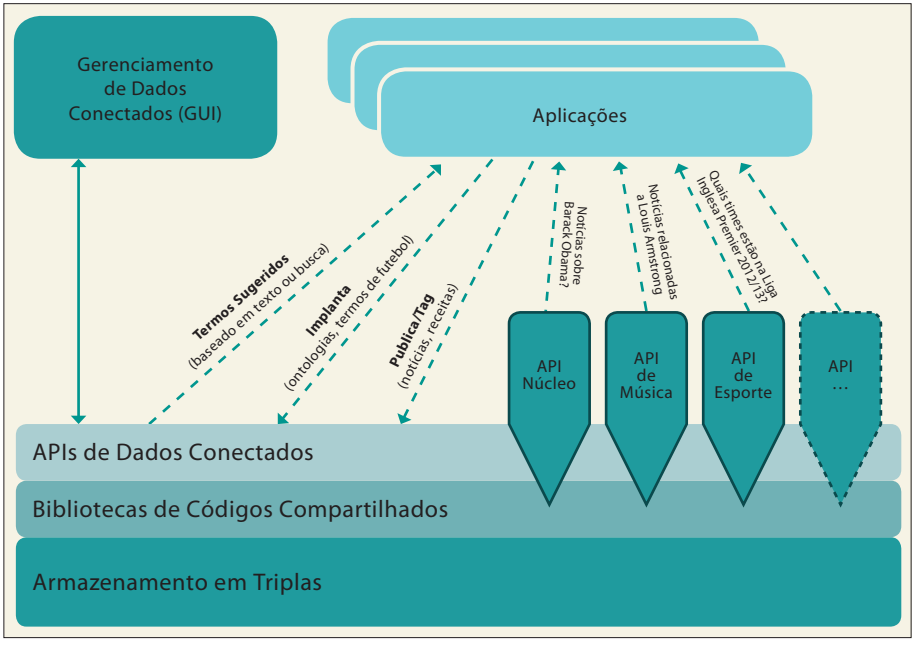

Figura 2.14 – Arquitetura de Dados Abertos da BBC. Fonte: adaptado de Bartlett (2013).

23 Plataforma para gerenciamento, acesso e integração de dados, permitindo dados em formato RDF (via armazenamento de triplas). Discutiremos sobre o Virtuoso em capítulos posteriores.

24 Veja mais detalhes em: <*http://derivadow.com/2007/06/21/bbc-music-brainz/*>.

A BBC tem o seu conteúdo armazenado utilizando uma *TipleStore* (ex.: OWLIM[25]) e tendo uma API para cada conteúdo com Dados Conectados.

A título de exemplo sobre o funcionamento da solução de Dados Abertos da BBC, apresentamos o caso a seguir. Ao entrarmos na página que oferece informações sobre o cantor do Pink Floyd, Roger Waters[26], podemos visualizar a página conforme mostra a Figura 2.15.

Figura 2.15 – Exemplo de dados conectados da BBC.

Esta página é criada (semi) automaticamente por meio do uso de dados abertos e tecnologias da Web Semântica, fazendo reúso do conteúdo tanto da Wikipédia (caso você abra a página do Roger Waters na Wikipédia, encontrará o mesmo conteúdo) quanto do MusicBranz. De acordo com a Figura 2.15, o retângulo inferior direito com a biografia do artista indica o trecho recuperado automaticamente da Wikipédia.

25 OWLIM, da mesma forma que o Virtuoso, oferece uma solução para gerenciamento, acesso e publicação de dados em formato RDF.

26 Disponível em: <*http://www.bbc.co.uk/music/artists/0f50beab-d77d-4f0f-ac26-0b87d3e9b11b*>.

Outro exemplo da própria BBC é a parte de programação da Rádio e TV. Esta parte faz uso de uma ontologia chamada *Programme Ontology* (Raimond et al., 2009) e reúsa diversas outras ontologias. Esta ontologia da BBC está disponível na Figura 2.16. Diferentemente da categoria de música, que só serializa em JSON e XML, a parte de programação da BBC serializa também em RDF[27].

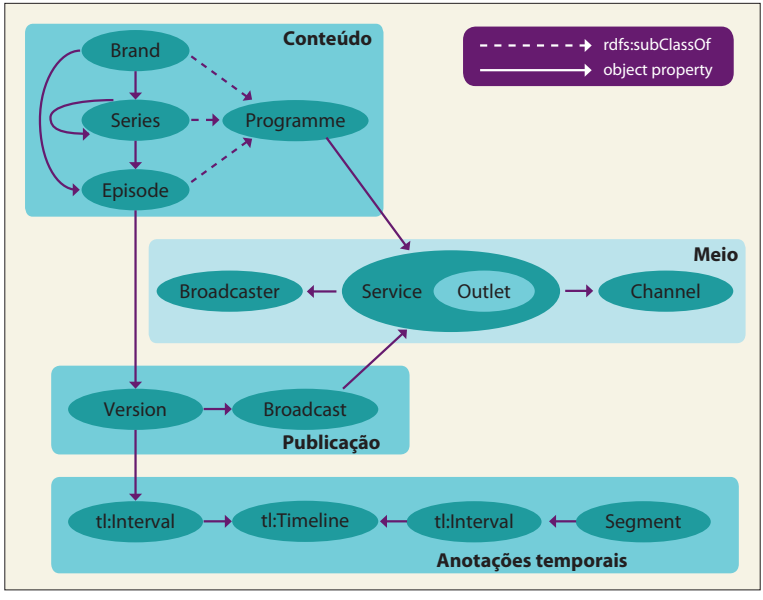

Figura 2.16 – Programmes Ontology da BBC. Fonte: http://www.bbc.co.uk/music/artists/0f50beab-d77d-4f0f-ac26-0b87d3e9b11b.

2.4.3 Um Caso Brasileiro: Globo.com

A Globo.com é um caso brasileiro que está fazendo uso de ontologias não somente para a publicação de conteúdo, mas também para decidir que tipo de *gadget* é disponibilizado em uma página de notícias. A Globo tem um grupo de profissionais da Web Semântica dedicado a organizar e distribuir todo o conteúdo produzido pelas

27 A título de exemplo, pegue o link *http://www.bbc.co.uk/programmes/b03ywp2h e coloque logo após o .rdf*.

organizações Globo. Atualmente a Globo.com tem uma ontologia de base que conecta e distribui dados de diferentes portais de informações da Globo (G1, Globo Esporte, Ego e TVG). Um exemplo é apresentado na Figura 2.17, em que a notícia sobre o Barcelona foi gerada com especificações de *tags* que se apoiam na descrição semântica das páginas, e, por consequência, os *gadgets* (por exemplo, na parte direita da página, em *leia mais sobre*) são gerados e conectados a outras notícias automaticamente (Medeiros, 2013).

Figura 2.17 – Exemplo de uso de semântica no Globo Esporte.

Observe que, na Figura 2.17, tanto a notícia quanto os *gadgets* estão relacionados com o time de futebol Barcelona, e não com a cidade espanhola de Barcelona. Isso é possível exatamente por causa da anotação semântica das notícias, que permite diferenciar que Barcelona está relacionado ao time de futebol e não necessariamente à cidade de Barcelona.

Isso é possível graças à definição de uma ontologia-base para conectar as notícias publicadas em diferentes portais da Globo e à nova arquitetura baseada em Dados Conectados que faz uso de uma API Restful, chamada Brainiak, como mostrado na Figura 2.18.

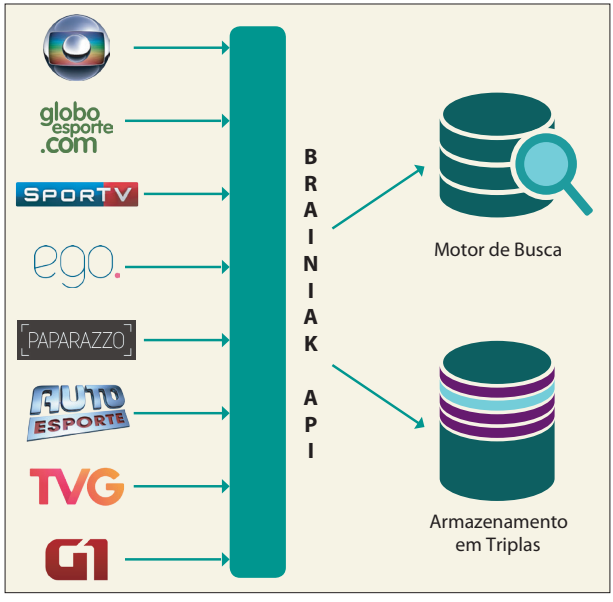

Figura 2.18 – Arquitetura da Globo para interoperabilidade semântica dos portais. Fonte: Globo.com.

2.4.4 Governo Conectado: Data.gov.uk

O Portal de Dados do governo britânico é uma das grandes iniciativas governamentais de abertura de Dados Conectados. O data.gov.uk tem em sua base quase 19 mil catálogos de dados disponíveis categorizados de diferentes formas para facilitar a busca e a descoberta de catálogos. Uma das formas que o Portal disponibiliza é por meio do esquema de classificação de Dados Abertos das 5 Estrelas (vide Figura 2.2), de acordo com a Tabela 2.5.

Tabela 2.5 – Classificação dos catálogos do Portal Britânico com o esquema 5 estrelas

Número de estrelas	Quantidade de catálogos
-	13.965
★	281
★★	1.145
★★★	2.345
★★★★	-
★★★★★	168

Podemos observar que, de acordo com a Tabela 2.5, ainda há muitos catálogos de dados para os quais não foi definida uma licença e que não podem ser considerados abertos. Alguns desses catálogos têm formatos de dados abertos, como XML e CSV, no entanto não definiram ainda uma licença.

Um exemplo de catálogo de dados com 5 estrelas é o *Land Registry Price Paid Data* (ou Dados dos Registros de Preços Pagos por Terra) sobre a Inglaterra e o País de Gales. Por meio deste dataset, é possível ter acesso a uma série temporal disponível em diferentes formatos (inclusive RDF) sobre o preço de cada venda de terra nestas duas regiões.

Outro cenário possível é que a maioria dos catálogos com formatos RDF disponíveis é sobre organogramas, fornecendo informações dos profissionais de cada organização e nível hierárquico nesta mesma organização. Um tipo de aplicação que pode ser construída utilizando-se dos dados dos diversos organogramas é a que verifica acúmulo de cargos públicos e horas de trabalhos semanais.

2.4.5 Nova York Conectada pelo povo e para o povo

A cidade de Nova York tem uma proposta de abertura de Dados Conectados integrada com a sociedade civil. A cidade não somente abre os dados do governo como estimula a sociedade a desenvolver aplicações utilizando estes dados, criando assim uma cidade conectada pelo povo e para o povo.

Uma das políticas do Portal de Dados Abertos da Cidade de Nova York é que os dados são abertos por padrão. No entanto alguns dos dados que são considerados confidenciais não são publicados de acordo com o relatório de Políticas de Segurança da Informação da Cidade de Nova York [*http://www.nyc.gov/html/doitt/downloads/pdf/data_classification.pdf*]. A arquitetura lógica do Portal de Dados Abertos da Cidade de Nova York é apresentada na Figura 2.19.

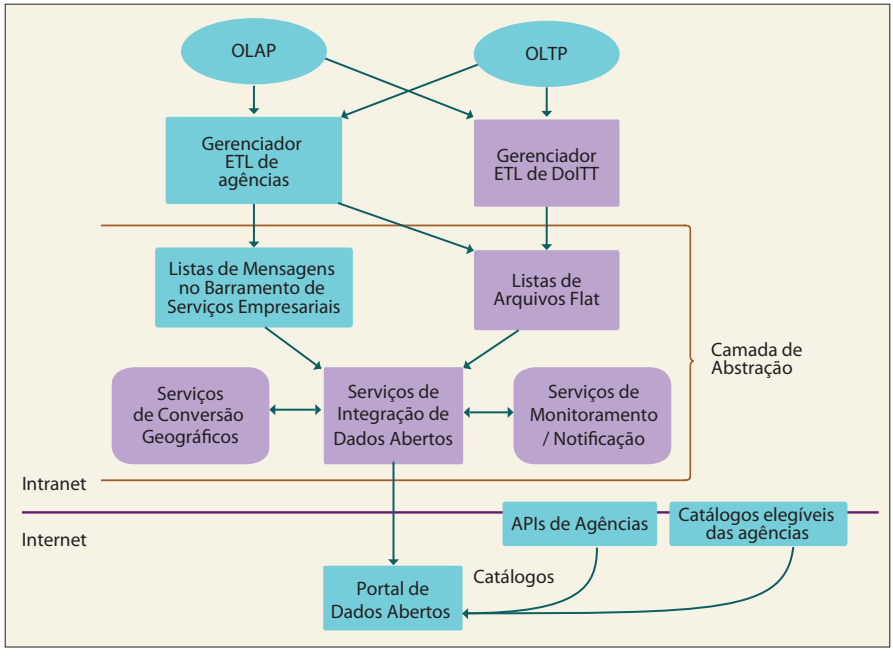

Figura 2.19 – Arquitetura lógica do Portal de Dados Abertos da cidade de Nova York. Fonte: adaptado de Bloomberg et al. (2012).

Observa-se que o Portal faz uso de uma camada de serviços de integração tanto de dados geográficos quanto de serviços de monitoramento. O Portal tem mais de mil catálogos de dados disponíveis em vários formatos, inclusive RDF. Por meio da plataforma utilizada, é possível tanto fazer a publicação dos dados abertos de forma simplificada e sistemática quanto exportar os dados para diferentes formatos. Além disso, a plataforma disponibiliza uma API na qual a sociedade civil pode consumir esses dados. É o que vem acontecendo por meio da iniciativa chamada BetaNYC[28], que tem mais de 1.500 desenvolvedores cadastrados e é responsável por organizar Hackathons para promover o desenvolvimento de soluções utilizando Dados Conectados.

Como um exemplo deste tipo de iniciativa, citamos a WayCount, que é uma solução para Cidades Inteligentes. O objetivo do WayCount é prover um hardware que possa medir em tempo real dados sobre o tráfico de bicicletas e automóveis. Diferente das soluções concorrentes, o WayCount segue dois princípios essenciais: baixo custo e dados abertos. Ou seja, utilizando tecnologias de open source, o WayCount coleta dados em tempo real e os disponibiliza como dados abertos, criando assim um rico repositório de dados abertos do tráfico para proporcionar a criação de cidades mais inteligentes.

Outro exemplo é o Data Explorer for US Politics (Explorador de Dados para a Política Americana), que é um plugin para o Chrome que disponibiliza dados sobre contribuições de campanhas. O Data Explorer está integrado à DBPedia, em que os dados são extraídos automaticamente (vide Figura 2.20).

28 Disponível em: <*http://betanyc.us*>.

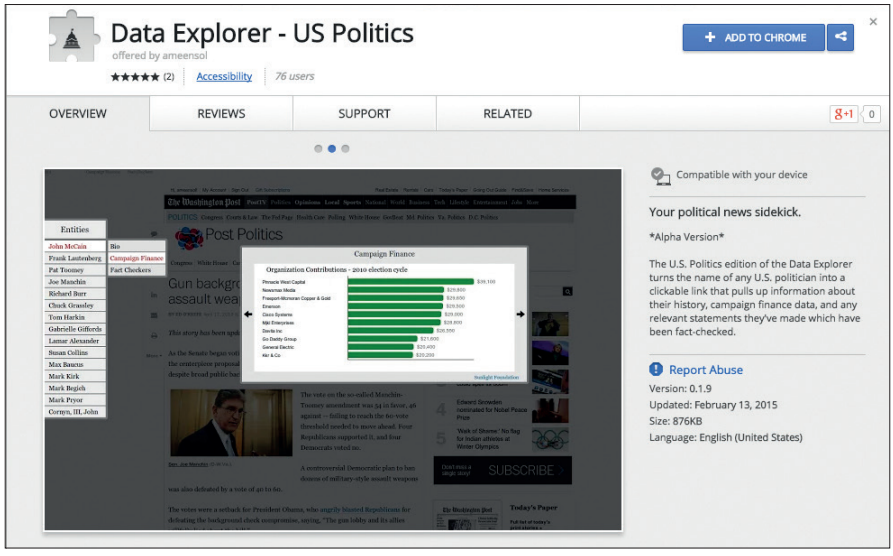

Figura 2.20 – Exemplo de aplicações sendo desenvolvidas por meio da iniciativa BetaNYC. Fonte: autores.

2.5 Considerações Finais

O principal objetivo deste capítulo foi oferecer ao leitor uma visão geral sobre como a estruturação de dados é feita na Web e como esta estrutura pode ser utilizada para representar Dados Abertos Conectados. Este capítulo foi organizado por meio de exemplos visuais com o intuito de trazer para o leitor pouco acostumado com o conceito de estruturação de dados na Web uma perspectiva incremental de estruturação de dados abertos. Foi também intenção deste capítulo dissertar sobre o modelo RDF, apresentando suas características principais, a semântica por trás do RDF e as principais formas de serialização de documentos RDF. Por fim, foram apresentados casos de sucesso da utilização de Dados Conectados, pelas perspectivas acadêmicas, governamentais e empresariais.

Esperamos que as seguintes mensagens tenham sido passadas:

- Compreensão sobre estruturação de dados na Web de Documentos e na Web de Dados.
- Conhecimento sobre como estruturar dados segundo o esquema de 5 estrelas para abertura de dados.
- Entendimento sobre a importância do modelo RDF, bem como de suas principais características.
- Aprendizado sobre alguns conceitos fundamentais para a semântica de documentos RDF.
- Compreensão sobre os diversos modelos de serialização de documentos RDF e suas aplicações.
- Visualização de casos de sucesso sendo aplicados nos contextos empresarial, governamental e acadêmico.

CAPÍTULO 3

Ontologias e Representação de Conhecimento

3.1 Introdução

No Capítulo 2, trabalhamos o conceito de estruturação de dados e vimos como ele é utilizado para auxiliar na descrição de Dados Conectados. Abordamos mais a fundo também o "sistema de 5 estrelas" de divulgação de dados abertos, além de apresentarmos alguns exemplos de sua utilização tanto no setor público quanto no privado. Contudo publicar um dado de maneira indiscriminada torna muito difícil o seu posterior uso. Isso ocorre porque um dado pode ser descrito, representado e interpretado de diferentes maneiras.

Para entender mais o problema de representação do conhecimento, vamos discutir alguns dos problemas ao lidar com a tripla *descrição-representação-interpretação* de dados. Um dos problemas na *descrição* de dados é o vocabulário a ser utilizado para representar um conceito desejado. Por exemplo, para descrever uma pessoa que está assistindo às aulas em uma universidade, poderíamos utilizar o termo "aluno". Entretanto existem outros termos, como "estudante", "aprendiz" ou até "universitário". Além disso, no caso de uma pessoa que está lecionando aulas, poderíamos usar os termos "professor", "instrutor", "mestre" e assim por diante. Nesses exemplos, apenas

dois conceitos sendo apresentados podem ser descritos por meio de uma diversidade de termos. Dessa forma, um usuário que quiser publicar dados abertos terá que escolher um conjunto de termos que descrevem adequadamente seus dados. Esse conjunto de termos é conhecido como vocabulário.

Além de escolher o vocabulário que melhor identifique um conjunto de dados, outro desafio é *representar* esses dados de maneira a aumentar a expressividade do dado dentro do contexto em que foi criado e reduzir a ambiguidade de sua posterior interpretação. Por exemplo, para os conceitos apresentados anteriormente, que podem ser descritos pelos termos "aluno" e "professor", existe a necessidade de dar mais riqueza de representação para que as seguintes perguntas possam ser respondidas facilmente: Quem pode ser professor? Quem pode ser aluno? Uma pessoa pode ser professor e aluno ao mesmo tempo?

Para responder a essas perguntas é preciso propor alguma forma de representação. No caso desse exemplo, uma representação seria indicar que professor e aluno são papéis (em inglês, *roles*) que um ator, representado pelo conceito de "pessoa", pode exercer no contexto de uma universidade. Assim deixa-se claro que o papel de professor ou aluno só existe dentro do contexto da universidade e é assumido por uma pessoa. Ademais, é possível colocar restrições nos atributos de uma pessoa para que ela possa assumir um determinado papel. Desse modo, apenas uma pessoa contratada pela universidade com o título de mestre ou doutor poderia assumir o papel de professor. Similarmente, apenas uma pessoa que finalizou o Ensino Médio e está regularmente matriculada em um curso da universidade poderia assumir o papel de aluno. Nessa situação, uma pessoa também poderia, ao mesmo tempo, assumir o papel de professor e aluno dentro de uma mesma universidade.

Essas restrições precisam estar claras para que *interpretações* incorretas não ocorram. Caso as restrições não sejam explicitamente definidas, os dados disponibilizados podem ser interpretados

erroneamente. No exemplo apresentado, os dados poderiam ser interpretados incorretamente em países em que um aluno não precisa cursar o Ensino Médio para conseguir se matricular em uma universidade (nos Estados Unidos, por exemplo).

Dessa maneira, para reduzir o problema de interpretação das representações dos dados, devem ser utilizados mecanismos e linguagens de representação/modelagem (visual ou lógica/formal) como o UML[1] (*Unified Modeling Language*) e o OWL (*Web Ontology Language*), que explicitam as relações (restrições e hierarquia, por exemplo) entre conceitos de maneira (semi) formal, permitindo que tanto pessoas quanto máquinas possam compreender os conceitos que representam os dados disponibilizados.

No contexto da Web, uma das formas de representação mais robustas atualmente ocorre por meio do uso de ontologias e OWL. Dessa forma, neste trabalho, introduziremos esses conceitos para que o leitor seja capaz de entender o potencial uso das ontologias e da OWL na disponibilização de dados abertos. E, no próximo capítulo, discutiremos as técnicas de criação de ontologias provindas da área de Engenharia de Ontologias.

3.2 Ontologias

O termo "ontologia" tem origem em um ramo da Filosofia (Metafísica) que estuda a natureza do "ser" e a "existência". Para os filósofos, a Ontologia visa a explicar todas as coisas do mundo, estabelecendo sistematicamente sua linhagem conceitual. Na Ciência da Computação, o significado e a finalidade desse termo são (um pouco) diferentes; uma ontologia pode ser definida como um conjunto de conceitos fundamentais e suas relações, que capta como as pessoas entendem (ou interpretam) o domínio em questão e permite a representação de tal entendimento de maneira formal, compreensível por humanos e computadores (Mizoguchi, 2004).

1 Disponível em: <*http://www.uml.org/*>.

Pesquisadores e demais especialistas na área têm diferentes entendimentos sobre o que significa uma ontologia. Neste capítulo, centralizaremos na área da Ciência da Computação. Segundo Gruber (1993), Ontologia é especificação explícita de uma conceitualização. A conceitualização refere-se ao significado de conceitos e suas relações, dado o contexto do domínio. E "especificação" significa uma representação formal, declarativa e explícita dos mesmos conceitos e das mesmas relações. Outra definição, dada por Swartout et al. (1999), é que uma ontologia é a estrutura básica ou a couraça em torno da qual uma base de conhecimento pode ser construída. Guarino (1997) também enfatiza que uma ontologia pode ser modelada para permitir o compartilhamento de conhecimento e a sua reutilização em diferentes aplicações.

Essas definições caracterizam bem o porquê de ontologias estarem atraindo recentemente muitos pesquisadores e desenvolvedores na área de Dados Abertos Conectados. Primeiro, elas fornecem uma estrutura conceitual comum sobre a qual podemos desenvolver bases de conhecimento compartilháveis e reutilizáveis. E, em segundo lugar, facilitam a interoperabilidade e a fusão das informações (*mash-ups*[2]), que viabilizam a criação de aplicações computacionais poderosas e mais inteligentes.

3.2.1 Composição de uma ontologia

Em Ciência da Computação, mais especificamente no campo da Inteligência Artificial, uma ontologia é constituída pelo seguinte (Mizoguchi, 2004; Isotani, 2009):

1. Um *conjunto de conceitos essenciais* resultantes da articulação do conhecimento básico presente em um determinado domínio. Esses conceitos podem ser representados por meio de um vocabulário especializado.

2 O termo *mash-up* refere-se ao processo de coleta e utilização de dados de diferentes fontes em uma única aplicação.

2. O *corpo de conhecimento*, que descreve o domínio por meio de conceitos essenciais. Ele é composto de:
    - uma hierarquia (classe/subclasse) resultante das relações *is-a*[3] entre conceitos.
    - um conjunto de relações importantes entre conceitos além das relações *is-a* (por exemplo, *part-of*[4]).
    - uma *axiomatização* de restrições semânticas entre esses conceitos e essas relações.

Formalmente, podemos definir uma ontologia como um relacionamento entre quatro elementos representado por $O = \{C, R, I, A\}$, em que (Kiryakov, 2006):

- C é o conjunto de classes que representam os conceitos em um dado domínio de interesse.
- R é o conjunto de relações ou associações entre os conceitos do domínio.
- I é o conjunto de instâncias derivadas das classes, ou ainda os exemplos de conceitos representados em uma ontologia.
- A é o conjunto de axiomas do domínio que servem para modelar restrições e regras inerentes às instâncias.

Esses elementos que constituem uma ontologia são fundamentais para a criação de uma estrutura que representa o conhecimento de um domínio.

Muitas vezes, durante o desenvolvimento de uma ontologia, as pessoas acabam gastando muito tempo para discutir a terminologia a ser utilizada (isto é, vocabulário), em vez de discutir e compreender os conceitos essenciais do domínio. Gostaríamos de enfatizar que, do ponto de vista dos autores (seguindo uma perspectiva orientada

3 *Is-a* define uma relação de conceitos de forma hierárquica, sendo esse tipo de relacionamento fundamental na construção de uma ontologia.

4 *Part-of* é outro tipo de relacionamento fundamental na construção de uma ontologia. Essa relação define como as coisas são especificadas de acordo com a composição de suas partes.

a conceito), uma ontologia não é um conjunto de termos interligados que formam um vocabulário (embora, algumas vezes, possa ser utilizada dessa forma). Mesmo que um termo possa ser considerado similar a um conceito, eles são diferentes em sua essência. *Um termo é essencialmente um rótulo para um conceito*. Quando falamos de um termo, o rótulo (palavra do vocabulário) torna-se o enfoque do problema. No entanto, para criar uma ontologia, o enfoque do problema é a definição de um conceito; e, portanto, o rótulo dado a ele passa a ter menos importância. Resumindo, uma ontologia propõe-se a representar e expressar o significado de conceitos, e não de termos (Mizoguchi, 2004; Guizzardi, 2007; Isotani, 2009). Isso ficará mais claro no próximo capítulo, quando apresentarmos a Figura 4.2, na qual abordamos os diferentes significados para a palavra *Bank*. Ou seja, caso os projetistas da ontologia se concentrem no termo *Bank*, o enfoque será dado de forma inadequada.

Se o leitor concorda com o ponto de vista dos autores, isto é, de que apenas a definição de um vocabulário interconectado não resulta em uma (boa) ontologia, então como é criada uma ontologia? Como é que se decide a qualidade de uma ontologia?

Uma possível resposta a essas perguntas é dada por Mizoguchi (2004): "*quanto mais ontológica for a ontologia, melhor*". Isso significa que uma ontologia é bem desenvolvida e tem mais qualidade quando o domínio pode ser explicado e as propriedades essenciais dos conceitos são explicitamente representadas, chegando perto da conceituação fundamental do conhecimento. Desse ponto de vista, o corpo de conhecimento que descreve o domínio é o núcleo das ontologias. Assim, ao criar uma ontologia, além da definição de conceitos e termos para identificá-los, deve-se: (a) fazer clara distinção entre as funções e os conceitos básicos; (b) identificar o uso apropriado das relações, especialmente as relações *is-a* e *part-of*; (c) evitar a herança múltipla; (d) distinguir corretamente o que é atributo e o que é propriedade; e (e) realizar muitas outras decisões importantes, a fim de produzir uma boa ontologia.

No próximo capítulo, abordaremos mais profundamente a metodologia utilizada para criar tais ontologias, apresentando a área de *Engenharia de Ontologias*.

3.2.2 Por que utilizar ontologias?

A concepção e o uso das ontologias sempre fizeram parte da proposta da Web Semântica, conforme introduzimos no Capítulo 1. Elas estão no centro da arquitetura proposta por Tim Berners-Lee (Figura 1.7 do Capítulo 1) e, ao longo da última década (2004-2014), têm-se mostrado uma das tecnologias-chave na criação de aplicativos mais adequados para lidar com grandes quantidades de informações de maneira inteligente (McGuinness, 2004; Horrocks, 2008).

De acordo com diversos autores, as ontologias oferecem o apoio necessário para resolver alguns dos problemas que cercam a construção de tecnologias que utilizam bases de dados com representação formal (ou bases de conhecimento – do inglês, *knowledge-bases*), como a seguir (Mizoguchi, 2004; D'aquin and Noy, 2012; Devedzic, 2006):

1. Premissas básicas são deixadas implícitas nas bases de dados, com isso, impedindo a reutilização e o compartilhamento do conhecimento representado.
2. Não há modelos genéricos comuns sobre os quais possamos construir bases de dados e aplicativos de maneira simplificada.
3. Não há tecnologia viável que permita acumulação incremental dos dados (isto é, estender rapidamente a base de dados).

Na era da Web de dados, em que a informação e o conhecimento são fragmentados na rede e os recursos estão em constante evolução, o desenvolvimento de aplicativos baseados em dados abertos não pode seguir o paradigma em que as bases de dados são estáticas e criadas para um problema muito específico em um domínio restrito. Atualmente, tanto o mercado quanto a academia exigem bases

de dados conectados, altamente compartilháveis, que permitam a interoperabilidade e a possibilidade de lidar com o acúmulo de conhecimento (isto é, novos dados conectados) disponível na Web.

Essa mudança de paradigma requer que os aplicativos atuais mudem de enfoque (Isotani et al., 2015; Dermeval et al., 2015):

1. *Do paradigma de desenvolvimento orientado a objetos para processo orientado a dados*. Geralmente, um dos principais problemas durante o desenvolvimento de aplicativos que lidam com grandes quantidades de dados de maneira inteligente tem sido como criar heurística e bases de conhecimento para guiar ou reproduzir comportamentos considerados adequados. Em outras palavras, o enfoque tem sido sobre os processos computacionais que guiam o comportamento do computador. No entanto, a partir do paradigma da Web de dados e com a crescente disseminação dos Dados Abertos Conectados, a ênfase dá-se em como utilizar os dados de maneira inteligente para auxiliar pessoas durante a tomada de decisão e na resolução de tarefas complexas. Aplicativos inteligentes, baseados em Dados Abertos Conectados, devem trazer informações e recursos adequados para as pessoas, para que elas possam – juntas – resolver os problemas e realizar as tarefas de forma mais eficiente. Para atingir tal objetivo, o processo de desenvolvimento de *software* deve seguir um paradigma orientado a dados. Assim os aplicativos inteligentes poderão utilizar de maneira eficaz os dados (em formato aberto e conectado) disponibilizados na Web para fornecer novas funcionalidades e novos conhecimentos aos usuários finais.

2. *Do paradigma de desenvolvimento centrado no usuário para o paradigma de desenvolvimento centrado na informação*. O objetivo principal do desenvolvimento de sistemas baseados em dados não é o de melhorar como desenvolver funcionalidades para

> resolver um problema específico. Em vez disso, estamos interessados em ampliar a capacidade humana, criando sistemas que têm a habilidade de oferecer informações provindas da análise de grandes conjuntos de dados e, dessa forma, ajudar na resolução de problemas complexos do mundo real. Para conceber tais sistemas, existe a necessidade de compreender a semântica por trás dos Dados Conectados e os processos humanos de aquisição de informação para desenvolver adequadamente sistemas e interfaces para visualização de dados.

Além dessa mudança de enfoque no desenvolvimento de sistemas, as ontologias também oferecem diversos benefícios para publicação e consumo de dados com alta qualidade, isto é, atingindo as 5 estrelas apresentadas nos capítulos anteriores. A publicação de Dados Abertos Conectados em conjunto com ontologias exige menos esforço tanto na definição do modelo de dados quanto na integração e no reúso de outras informações disponíveis na Web. Por exemplo, as ontologias não permitem que a publicação e o consumo de dados abertos ocorram de maneira inconsistente ao usar regras de inferência que possibilitam checar automaticamente a corretude das relações entre os dados e as suas instâncias. Outro benefício é a possibilidade de utilizar métodos para representação de dados tanto no nível procedural (por exemplo, utilizando SKOS – *http://www.w3.org/2004/02/skos/*) quanto no nível conceitual (por exemplo, utilizando ontologias de topo[5] como as disponíveis em *http://suo.ieee.org/*). As ontologias também oferecem um bom balanço entre os esforços de criar um bom modelo de dados (abordado no Capítulo 4) e o valor que ele proporciona para o entendimento e o consumo desses dados.

5 Do inglês *upper ontologies* – são ontologias que descrevem conceitos fundamentais e genéricos que têm o mesmo significado em diversos domínios diferentes, permitindo ser utilizadas como um guarda-chuva para criar outras ontologias.

3.2.3 Tipos de Ontologia

Existem vários tipos de ontologia (Guarino, 1997). Nesta seção, destacaremos as duas formas mais comuns de diferenciar ontologias: 1) ontologias pesadas *versus* ontologias leves; 2) ontologias de domínio *versus* ontologias de tarefa.

As *ontologias leves* (*lightweight ontologies*) são aquelas que não se preocupam em definir detalhadamente cada conceito representado. A principal ênfase das ontologias leves é definir a taxonomia que representa a relação hierárquica entre conceitos. Esse tipo de ontologia vem sendo utilizado na Web para categorizar grandes quantidades de dados, principalmente em portais como Yahoo! e AOL (Bechhofer et al., 2006; *Bizer*, 2009). *Ontologias pesadas* ou densas (*heavyweight ontologies*) enfocam não apenas a taxonomia, mas também a representação rigorosa da semântica entre os conceitos. O desenvolvimento de ontologias pesadas requer a definição de cada conceito, a organização desses conceitos baseados em princípios bem definidos, uma definição formal da semântica entre os conceitos e suas relações, além de outras considerações. Para criar bases de conhecimento reusáveis e compartilháveis é fundamental definir ontologias pesadas.

Ontologias de domínio e de tarefa são necessárias para criar sistemas mais flexíveis e inteligentes e que possam ser aplicados em diversos domínios. A *ontologia de domínio* define e caracteriza o domínio no qual as tarefas ocorrem, e a *ontologia de tarefa* representa os processos e as atividades para resolver um determinado problema abstraindo o contexto do domínio. Em outras palavras, a ontologia de domínio representa o conhecimento sobre um tópico, enquanto a ontologia de tarefa representa a habilidade de aplicar esse conhecimento para resolver problemas em diferentes situações. Essa distinção é muito importante, pois, por meio dela, torna-se possível criar sistemas e bases de conhecimento mais modulares, compartilháveis e extensíveis.

Para criar uma ontologia de tarefa, inicialmente, os passos de resolução de problemas precisam ser decompostos em ações básicas. Além disso, a estrutura hierárquica que representa as restrições para executar essas ações deve ser desenvolvida. De acordo com Mizoguchi (2004), as seguintes categorias conceituais devem ser consideradas no desenvolvimento de uma ontologia de tarefa: (a) papéis (*task roles*) – refletem o papel desempenhado pelos objetos do domínio durante o processo de resolução de problemas; (b) ações (*task actions*) – representam uma unidade de atividade contida no processo de resolução de problemas; e (c) os estados dos objetos e o fluxo da informação, que, além de papéis e tarefas, são fatores que precisam estar formalmente representados. A Figura 3.1 apresenta uma classificação de ontologias de acordo com as dimensões de expressividade, propósito e especificidade.

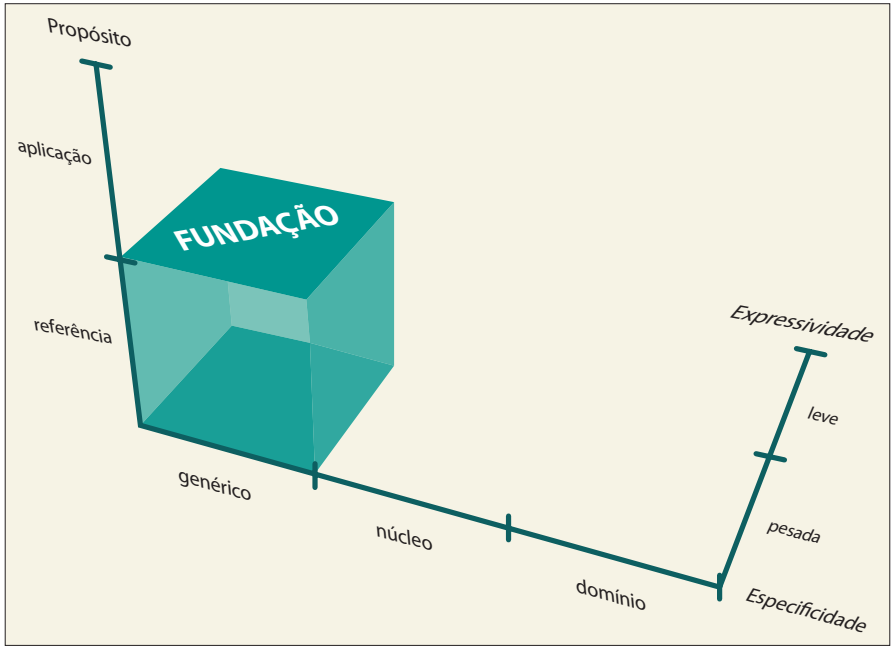

Figura 3.1 – Classificação de Ontologias de acordo com o Tripé Propósito x Especificidade x Expressividade. Fonte: adaptado de Gangemi et al.

3.2.4 Representação de ontologias

Basicamente existem duas maneiras de representar ontologias. A primeira delas é a representação formal e a segunda é a representação gráfica. A representação formal é usada para que as ontologias possam ser consumidas por computadores, enquanto a representação gráfica é usada para a compreensão humana. Ambas são importantes, uma vez que a falta de uma dessas representações afeta a qualidade/o uso da ontologia.

Quanto à representação formal consumida por computadores, existem algumas linguagens para representação de ontologias (Patel-Schneider, 2005). Normalmente, usa-se lógica de predicados, lógica descritiva ou linguagens baseadas em "quadros" (*frames*). No entanto linguagens mais expressivas (conhecidas como linguagens de descrição de ontologias) têm sido propostas (Horrocks et al., 2003). Atualmente, as linguagens mais populares para descrever ontologias são RDF[6]/RDF-S[7] e OWL[8].

Resource Description Framework (RDF) é a especificação proposta pelo *World Wide Web Consortium* (W3C) para descrever metadados. Ela permite criar triplas que contêm um nó *sujeito*, uma relação chamada de *predicado* e o nó *objeto* (*sujeito, predicado, objeto*). Mediante essa tripla, é possível indicar a relação entre dados e usá-la para representar a semântica contida neles. Por exemplo, é possível indicar a relação entre um autor e seus livros de forma que não apenas pessoas compreendam o significado dessa relação, mas que computadores também possam compreender esta informação. RDF fornece uma maneira simples de representar triplas[9].

Para expressar a semântica por meio de triplas, também conhecidas como vocabulário RDF, é necessária a definição de *tags*. RDF-Schema (ou RDF-S) é utilizada para tal fim. RDF-S é a especificação que define classes, propriedades e seus relacionamentos,

6 Resource Description Framework: <*http://www.w3.org/RDF/*>.
7 RDF Schema: <*http://www.w3.org/TR/rdf-schema/*>.
8 Web Ontology Language: <*http://www.w3.org/TR/owl-features/*>.
9 Para mais informações sobre RDF, veja Capítulo 2.

que podem ser usados para descrever triplas. Isso inclui a definição de *tags* e sua estrutura hierárquica (taxonomia) para restringir os valores de uma tripla representada em RDF. Assim a RDF-S fornece os elementos mínimos para a descrição de ontologias[10]. Embora RDF-S possa ser usada para descrever ontologias, ela tem algumas limitações, especialmente para apoiar o raciocínio computacional dos dados disponíveis na Internet (Patel-Schneider, 2005). Assim, uma linguagem mais expressiva foi desenvolvida e é conhecida como *Web Ontology Language* (OWL).

A OWL também é uma linguagem desenvolvida e aprovada pelo W3C. Ela tenta satisfazer ao formalismo exigido pela comunidade de Web Semântica para que programas possam compreender e responder a consultas de agentes (pessoas ou outros programas) por meio do uso de descrições ontológicas (Horrocks et al., 2003). Atualmente, a OWL é a linguagem mais utilizada para representar ontologias formalmente, apresenta variantes da linguagem que lidam com a escalabilidade e a expressividade das ontologias e permite que aplicações com diferentes propósitos sejam construídas (Mizoguchi, 2004)[11].

Existem muitas maneiras de representar graficamente uma ontologia, uma vez que esta é composta principalmente de conceitos e suas relações. Algumas formas comuns para representar graficamente ontologias são grafos, UML, estrutura de árvore, além de outras. Para exemplificar a diferença entre uma representação formal e uma representação gráfica, vamos definir o conceito de bicicleta (Isotani, 2009): uma bicicleta (*bicycle*) é um tipo de veículo (*vehicle*) de transporte, ou seja, pode-se definir a relação "*bicycle is-a vehicle*". Além disso, uma bicicleta pode ser especializada em bicicletas esportivas (*sport bicycle*) e bicicletas para uso na cidade (*city bicycle*). Finalmente, é possível identificar os atributos e as propriedades da bicicleta como cor, peso, tamanho etc. Como resultado, o trecho de código a seguir mostra parte da representação da ontologia que descreve

10 Para mais informações sobre RDF-S, veja Capítulo 2.
11 Para mais informações sobre OWL, veja Seção 3.3.

uma bicicleta utilizando OWL, enquanto a Figura 3.2 mostra a representação gráfica de uma parte mais completa da mesma ontologia utilizando uma ferramenta gráfica conhecida como Hozo[12].

```
<owl:Class rdf:ID="Vehicle">
   <rdfs:label>Vehicle</rdfs:label>
   <rdfs:SubClassOf rdf:resource="#Any" />
</owl:Class>

<owl:Class rdf:ID="sport_cycle">
   <rdfs:label>sport_cycle</rdfs:label>
   <rdfs:SubClassOf rdf:resource="#bicycle" />
</owl:Class>

<owl:Class rdf:ID="city_cycle">
   <rdfs:label>city_cycle</rdfs:label>
   <rdfs:SubClassOf rdf:resource="#bicycle" />
</owl:Class>

<owl:Class rdf:ID="bicycle">
   <rdfs:label>bicycle</rdfs:label>
   <rdfs:SubClassOf rdf:resource="#Vehicle" />
   <rdfs:SubClassOf>
      <owl:Restriction>
         <owl:cardinality rdf:datatype="http://www.w3.org/
         <owl:onProperty rdf:resource="#has_body_color" />
      <owl:Restriction>
   </rdfs:SubClassOf>
   <rdfs:SubClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#has_body_color" />
         <owl:allValuesFrom rdf:resource="#Color" />
```

Na Figura 3.2, *p/o* e *a/o* que aparecem na frente dos conceitos representam, respectivamente, as relações de *part-of* e *attribute-of*. Assim o corpo da bicicleta (*frame*), o pedal, o guidão e a roda são parte (*part-of*) da bicicleta. Enquanto cor e peso são atributos dela.

12 Hozo Ontology Editor: <*http://www.hozo.jp/*>.

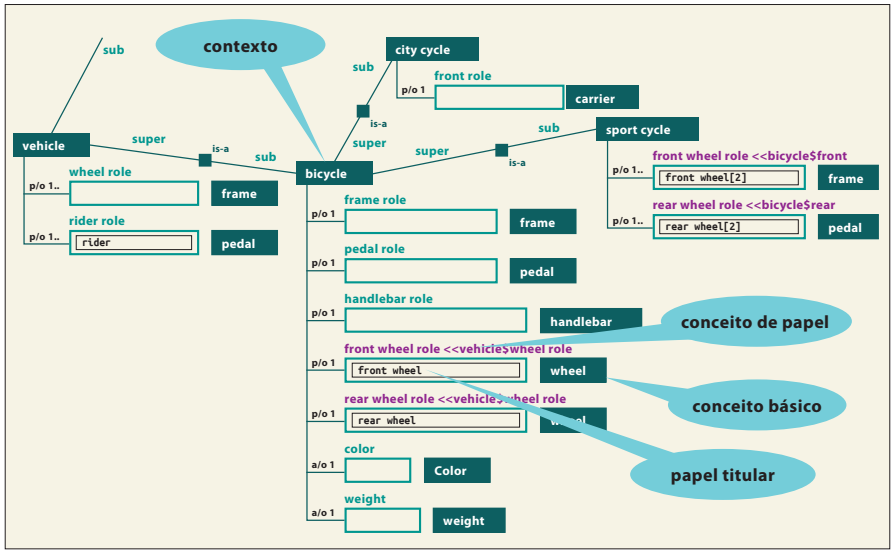

Figura 3.2 – Representação gráfica da ontologia de bicicleta utilizando Hozo. Fonte: adaptado de http://www.hozo.jp/.

O corpo do conhecimento de uma ontologia é responsável por representar o núcleo do domínio e dar significado (e propriedades) aos conceitos representados. Portanto, para criar boas ontologias pesadas, é fundamental ter uma metodologia que auxilie sua definição e construção. Mizoguchi (2004) indica que, para criar uma boa ontologia, além da definição dos conceitos e dos termos para rotulá-los, é necessário:

1. Fazer uma distinção entre conceitos básicos e papéis ou função (*roles*) que um conceito pode ter dentro de um contexto. Por exemplo, o conceito *professor* pode ser definido como a função de uma pessoa dentro do contexto escolar (se a escola desaparecer, o professor também desaparece), enquanto o conceito *pessoa* é um conceito básico que não depende de contexto.
2. Identificar as várias relações entre conceitos representando de forma explícita as relações *is-a*, *part-of* e *attribute-of*.
3. Evitar a herança múltipla.

4. Distinguir e definir corretamente o que é um atributo e uma propriedade de um conceito, além de realizar decisões importantes a fim de produzir uma ontologia de boa qualidade.

3.3 Linguagem de ontologias da Web

A OWL (do inglês, *Ontology Web Language*) é considerada a linguagem de ontologias da Web e é bastante utilizada para o desenvolvimento de aplicações baseadas na Web Semântica. Como descrito nas Figuras 1.7 e 1.8, a OWL é uma linguagem baseada nas especificações do RDF/RDF-S. Isso quer dizer que OWL, por um lado, herda as características do RDF como a estrutura baseada em triplas e a descrição de recursos com URI, e, por outro lado, herda a semântica descrita no Esquema RDF.

É importante destacar que há muita descrição errada sobre o que é OWL e como aplicá-la. Isto ocorre pela complexidade inerente ao termo Ontologias e pela expressividade da linguagem OWL. Destacamos a seguir três características que *não* são inerentes à OWL (Hitzler et al., 2012):

1. **Não é uma linguagem de programação**: OWL é uma *linguagem declarativa* que descreve um determinado universo do discurso de forma lógica. A partir do momento que se descreve conhecimento por meio de ontologias, pode-se fazer uso de ferramentas para inferir[13] novas informações sobre o universo de discurso. No entanto a forma que essas ferramentas fazem inferência sobre ontologias não faz parte do escopo da OWL.
2. **Não é uma linguagem de esquema para conformidade sintática**: não faz parte do escopo da OWL prescrever como certo documento deve ser sintaticamente estruturado. Ou seja, a Linguagem de Ontologias da Web não obriga que determinada parte do conhecimento esteja sintaticamente presente.

13 Esses tipos de ferramenta são chamados de *Reasoners*, e alguns bem conhecidos são FaCT++, Racer Pellet e HermiT.

3. **Não é um banco de dados**: essa característica faz com que haja muita aplicação errada de OWL. O que se deve compreender é que os documentos OWL armazenam as informações (instâncias) e, por essa razão, faz o armazenamento dos dados. A principal diferença entre banco de dados e OWL é a semântica utilizada em cada um deles. Os bancos de dados são mundos fechados (do inglês, *Closed-World Assumptions*), enquanto as ontologias são mundos abertos (do inglês, *Open-World Assumptions*). Isso implica dizer que, se determinado fato não está presente em um banco de dados, ele é considerado falso. Já em OWL ou nos mundos abertos, a implicação é diferente, pois, se determinado fato não está presente, ele é considerado desconhecido, porque é possível que seja verdadeiro. Por exemplo, se houver uma determinada declaração como a descrita a seguir sobre o cidadão brasileiro *Seiji*, não havendo descrição de que *Ig* também é um cidadão brasileiro, essa pergunta será considerada falsa no mundo fechado.

   - Declaração: <Seiji> <é cidadão> <Brasil>
   - Pergunta: <Ig> <é cidadão> <Brasil>?
   - Resposta (mundo fechado): Não!
   - Resposta (mundo aberto): Não Sei!

De um ponto de vista didático, podemos dividir a OWL em duas camadas, sendo uma para descrever a sintaxe e outra para a semântica. A Figura 3.3 apresenta a estrutura da linguagem OWL 2 (W3C OWL Working Group, 2012).

As duas subseções que seguem têm por objetivo descrever tanto a camada sintática quanto a camada semântica.

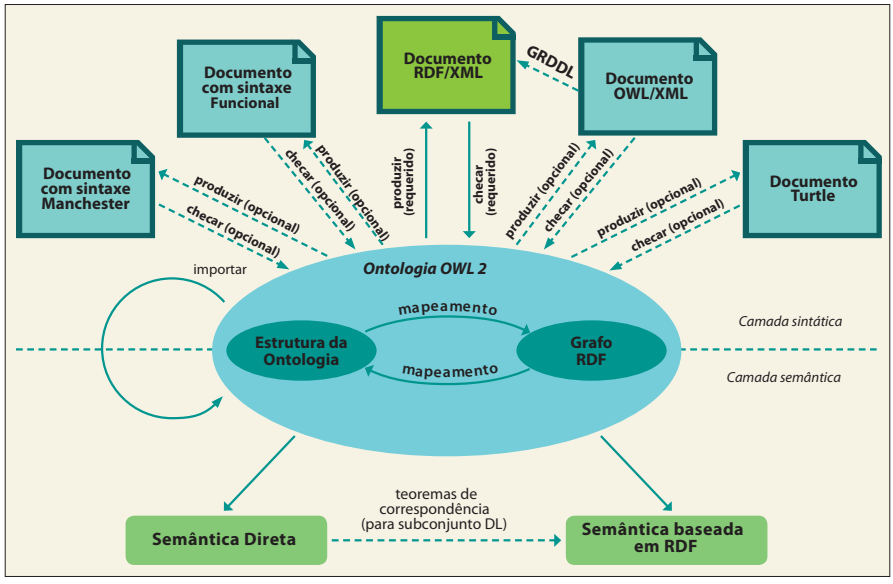

Figura 3.3 – Estrutura da Linguagem OWL. Fonte: adaptado de W3C OWL Working Group (2012).

3.3.1. Sintaxe da linguagem OWL

Como podemos observar na Figura 3.3, toda ontologia criada em OWL 2 tem uma estrutura sintática mandatória baseada em RDF/XML. Além do documento RDF/XML, é possível criar documentos OWL com mais quatro diferentes formas de serialização, de acordo com a Tabela 3.1.

Tabela 3.1 – Formatos de serialização OWL e seus propósitos

Serialização	Status	Propósito
RDF/XML	Mandatória	Formato obrigatório, podendo ser reconhecido por qualquer software OWL.
OWL/XML	Opcional	Processamento simples com ferramentas XML
Sintaxe Funcional	Opcional	Visualização simples da estrutura formal da OWL
Sintaxe Manchester	Opcional	Leitura/Escrita simples de Ontologias em Lógica de Descrição
Turtle	Opcional	Leitura/Escrita simples de triplas RDF

Os formatos RDF/XML e Turtle foram apresentados no Capítulo 2. Já o formato OWL/XML respeita a sintaxe de documentos XML e tem o propósito de processar documentos OWL por meio de ferramentas de leitura/escrita de XML, como *XQuery*. Esse é um formato mais comum para soluções corporativas que já fazem uso de documento XML. A sintaxe funcional apresenta uma forma simples de visualizar e compreender documentos OWL, enquanto a serialização em sintaxe Manchester apresenta uma maneira simples de manipular estruturas em lógica de descrição.

Apesar de OWL apresentar cinco formatos de serialização, apenas um é mandatório, e os outros devem ser usados de acordo com a necessidade do consumidor da ontologia. Os ontologistas só precisam se preocupar com um formato e, da mesma forma que explicamos sobre os formatos de serialização RDF, não há a necessidade de criar uma descrição em diferentes formatos. Frisamos que, independentemente do formato de serialização, a linguagem OWL 2 tem uma estrutura conceitual bem-definida, que é transformada para os diferentes formatos.

Não podemos nos esquecer de que o propósito da OWL é representar conhecimento. As noções básicas de modelagem de conhecimento em OWL consideram três aspectos básicos (Hitzler et al., 2012):

1. **Entidades**: elementos usados para referenciar um objeto do mundo real. Mediante as entidades, os termos primitivos de uma determinada ontologia são definidos. Por exemplo, podemos representar o grupo de todas as pessoas pela entidade <Person>. As entidades descritas em OWL são identificadas por um IRI e podem ser de diferentes tipos.
2. **Expressões**: combinação de entidades para formar descrições mais complexas, desenvolvidas por meio das expressões criadas pelos termos primitivos. Por exemplo, podemos representar o conjunto de todos os professores doutores mediante a conjunção das classes `<Professor>` e `<Doctor>` e criar a nova entidade `<DoctorProfessor>`.

3. **Axiomas**: as declarações básicas que uma ontologia em OWL expressa. Por meio da utilização dos axiomas, pode-se fazer inferências sobre as entidades. Por exemplo, se definimos que <Student> <SubClassOf> <Person>, pode-se concluir que uma determinada instância de estudante é também uma instância de pessoa.

Além desses elementos básicos, não podemos nos esquecer de que os recursos descritos em OWL têm um IRI associado. Dessa forma, a Figura 3.4 apresenta a estrutura de ontologias descritas em OWL 2 (Bock et al., 2012).

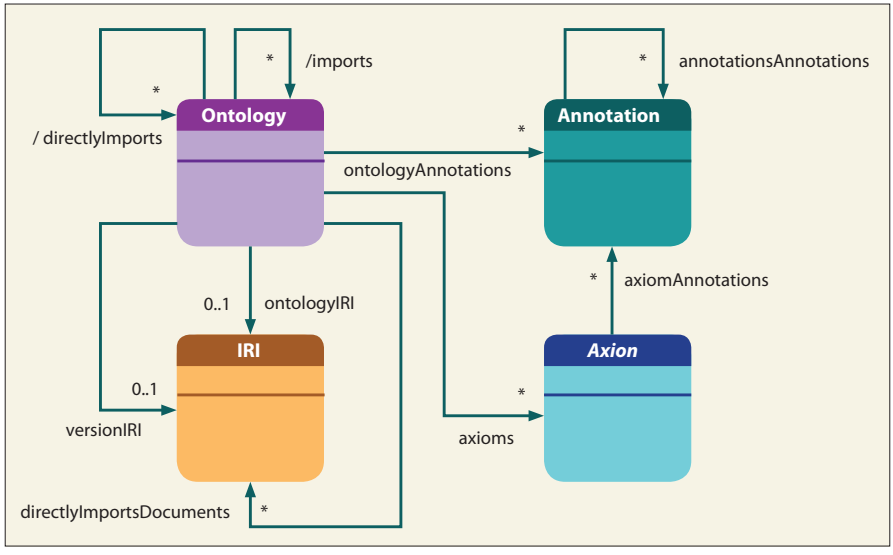

Figura 3.4 – Estrutura das ontologias descritas em OWL 2. Fonte: adaptado de Bock et al. (2012).

Como apresentamos na Figura 3.4, todo documento OWL contém um conjunto de axiomas e um IRI associado. Observe que uma ontologia pode importar outras ontologias, o que caracteriza o reúso de ontologias. Além disso, uma ontologia pode ter diferentes IRIs, com o objetivo de caracterizar versões daquelas ontologias. Outro elemento presente é a *Annotation*, que permite que a ontologia contenha anotações, de modo que a ontologia tenha um conjunto de informações

sobre ela (ou metainformação). Por exemplo, podemos querer identificar a organização que criou o documento, bem como os seus autores.

Uma vez explicada a estrutura das ontologias descritas em OWL 2, dissertamos sobre os elementos básicas da modelagem OWL. Da mesma forma que identificamos na Subseção 2.3.2 que um Esquema RDF também o faz, uma ontologia OWL contém classes (`owl:Class`), propriedades (`owl:ObjectProperty` e `owl:DataProperty`) e indivíduos ou instâncias (`owl:Individual`). A seguir é apresentado um exemplo em sintaxe funcional de classe, propriedade e indivíduo.

```
Class ( :Person )
NamedIndividual ( :Mary )
NamedIndividual ( :John )
ClassAssertion ( :Person :Mary)
ClassAssertion ( :Person :John)
ObjectProperty ( :hasWife )
ObjectPropertyAssertion ( :hasWife :John :Mary )
```

Pelo exemplo anterior, temos a descrição de uma classe (`<Person>`), duas instâncias (`<Mary>` e `<John>`), uma propriedade (`<hasWife>`) e uma relação (`<John> <hasWife> <Mary>`). Outra característica presente em OWL que também está presente no RDF-S é a possibilidade de gerar hierarquias de classes e propriedades. As hierarquias são definidas mediante a especificação dos axiomas subclasses (`owl:SubClassOf`) e subpropriedades (`owl:SubObjectPropertyOf` e `owl:SubDataPropertyOf`). Este exemplo apresenta a utilização destes conceitos:

```
SubClassOf (:Woman :Person)
SubPropertyOf (:hasWife :hasSpouse)
```

De acordo com o exemplo, temos a identificação de que mulher (`<Woman>`) é uma subclasse de pessoa (`<Person>`), o que implica que qualquer instância de `<Woman>` é também uma instância de `<Person>`. Esse mesmo cenário está presente nas propriedades, nas quais indicamos no exemplo que esposa (`<hasSpouse>`) é subpropriedade de

mulher casada[14] (<hasWife>). Esse conceito pode ser aplicado também na utilização de sinonímias e quase-sinonímias, definindo assim propriedades equivalentes. No entanto é importante frisar que classes, propriedades e indivíduos contêm elementos para descrever conceitos equivalentes. Por exemplo, podemos dizer que a classe pessoa (<Person>) é equivalente à classe humano (<Human>) e que as propriedades de filho e idade de cada classe são também equivalentes.

```
EquivalentClasses (ontA:Person ontB:Human )
EquivalentObjectProperties (ontA:hasChild ontB:child)
EquivalentDataProperties (ontA:hasAge ontB:age)
```

Podemos observar que, nesse exemplo, adicionamos os espaços de nomes *ontA* e *ontB*, pois a equivalência é comumente usada na integraçãoe no reúso de ontologias. A partir do momento que essas relações foram definidas, infere-se que as instâncias dessas entidades são equivalentes. O mesmo acontece quando temos duas instâncias diferentes, porém representando o mesmo indivíduo. Por exemplo, o ator Lima Duarte tem como nome de batismo Ariclenes Venâncio. Nesse caso, especifica-se em OWL como segue:

```
SameIndividual (:LimaDuarte :AriclenesVenancio )
```

A implicação do estabelecimento desse tipo de axioma é que tudo o que estiver relacionado com Ariclenes Venâncio está também relacionado com Lima Duarte. Esses axiomas de equivalência que foram supra-apresentados não existem no RDF-S, o que já demonstra um aumento na expressividade da linguagem e consequentemente um maior poder de especificação de uma conceitualização.

Podemos, além de estabelecer que um indivíduo pode ser instância de diferentes classes (isto é, por meio da utilização de `SubClassOf` e `SameIndividual`), estabelecer também que os indivíduos não podem pertencer à mesma classe. Isso ocorre mediante a utilização de classes disjuntas (`owl:DisjointClasses`), como descrito a seguir:

14 Consideramos aqui o termo composto "mulher casada" como tradução para *wife*, no entanto há vários sinônimos presentes nos diferentes dicionários.

```
DisjointClasses ( :Man :Woman )
```

Outras noções disponíveis em OWL, que não estão disponíveis em RDF-S, são as entidades complexas que podem ser especificadas. Podemos criar novas entidades por meio da união de outras entidades. Por exemplo, a seguir, criamos a entidade "pais" (<*Parents*>), sendo formada pela união das entidades "mãe" (<*Mother*>) e "pai" (<*Father*>).

```
EquivalentClasses (
   :Parent
   ObjectUnionOf ( :Mother :Father )
)
```

Estes são alguns dos conceitos que estão presentes na sintaxe de OWL e que não estão em RDF-S. Outros elementos que estão presentes em OWL 2 são descritos na Tabela 3.2.

Tabela 3.2 – Exemplos de outros elementos presentes na OWL 2

Elemento	Propósito	Utilização
`DifferentIndividuals`	Especificar diferentes instâncias	`DifferentIndividuals ( :John :Mary )`
`ObjectIntersectionOf`	Especifica nova classe a partir da intersecção de outras	`EquivalentClasses ( :Mother ObjectIntersectionOf( :Woman :Parent ) )`
`ObjectComplementOf`	Especifica nova classe a partir da diferença entre duas classes	`EquivalentClasses ( :ChildlessPerson ObjectIntersectionOf ( :Person :ObjectComplementOf ( :Parent ) ) )`
`ObjectSomeValuesFrom`	Quantificador Existencial – descreve que, para todos os indivíduos de uma classe, há pelo menos um indivíduo de outra classe relacionada por determinada propriedade	`EquivalentClasses ( :Parent ObjectSomeValuesFrom ( :hasChild :Person ) )`

Elemento	Propósito	Utilização
`ObjectAllValuesFrom`	Quantificador Universal –descreve uma classe de indivíduos na qual todos os indivíduos relacionados devem ser de uma determinada classe	`EquivalentClasses ( :HappyPerson ObjectAllValuesFrom ( :hasChild :HappyPerson ) )`
`ObjectHasValue`	Restrição que especifica que uma classe de indivíduos se relaciona com um indivíduo em particular	`EquivalentClasses ( :JohnsChildren ObjectHasValues ( :hasParent :John ) )`
`ObjectHasSelf`	Restrição que especifica que um indivíduo se relaciona com ele mesmo	`EquivalentClasses ( :NarcisisticPerson ObjectHasSelf (:loves ) )`
`ObjectMaxCardinality`	Restrição que descreve o número máximo de indivíduos de determinada relação	`ClassAssertion ( ObjectMaxCardinality ( 32 :hasTeams :Team ) :FIFAWorldCup )`
`ObjectMinCardinality`	Restrição que descreve o número mínimo de indivíduos de determinada relação	`ClassAssertion ( ObjectMinCardinality ( 1 :hasHostCountry :Country ) :FIFAWorldCup )`
`ObjectExactCardinality`	Restrição que descreve o número exato de indivíduos de determinada relação	`ClassAssertion ( ObjectExactCardinality ( 2 :hasTeams :Team ) :FIFAWorldCupGame )`
`ObjectOneOf`	Descreve uma classe com todas as suas instâncias	`EquivalentClasses ( :WorldCupGroupA ObjectOneOf (:Brazil :Croatia :Mexico :Camaroon ) )`
`InverseObjectProperties`	Descreve que, se a classe A está relacionada com B por propriedade X, implica que a classe B está relacionada com A por propriedade Y	`InverseObjectProperties ( :hasParent :hasChild )`
`SymmetricObjectProperty`	Descreve que, se a classe A está relacionada com B por propriedade X, implica que a classe B está relacionada com A da mesma forma	`SymmetricObjectProperty ( :hasSpouse )`

Elemento	Propósito	Utilização
`ReflexiveObjectProperty`	Descreve que determinada classe se relaciona por meio de determinada propriedade com ela mesma	`ReflexiveObjectProperty ( :hasRelative )`
`FunctionalObjectProperty`	Descreve que determinado indivíduo contém apenas uma instância de determinada classe em uma relação	`FunctionalObjectProperty ( :hasHusband )`
`HasKey`	Restrição que especifica uma chave para determinada instância de uma classe	`HasKey ( :Person () ( :hasSSN) )`

Muitas das implicações da utilização desses novos elementos são semânticas, podendo aumentar a capacidade de inferência e novas derivações a partir do que foi especificado[15]. A próxima subseção descreve a semântica da linguagem OWL.

3.3.2 Semântica da linguagem OWL

Esta subseção tem por objetivo descrever a semântica por trás da linguagem OWL, mais especificamente abordando o poder de inferência da OWL. A Figura 3.5 apresenta como a linguagem OWL é constituída e como a lógica formal compõe a linguagem.

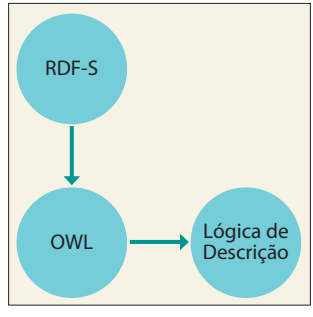

Figura 3.5 – Relacionamento da semântica formal na linguagem OWL.

15 Nem todos os elementos foram especificados na Tabela 3.2, sendo importante olhar a documentação disponível em: <*http://www.w3.org/TR/owl2-primer/*>.

Podemos observar que a OWL faz uso de construtores presentes na lógica de descrição. A lógica de descrição equivale a um formalismo de representação de conhecimento baseado em lógica, tendo sido remanescente das redes semânticas. Diferentemente de lógica de primeira ordem, a lógica de descrição está interessada em um procedimento que garanta que as respostas (positivas ou negativas) sejam dadas em tempo finito (Lutz et al., 2005)[16].

De fato, como pode ser visto na Figura 3.3, há duas formas alternativas de especificar semântica em OWL:

1. **Semântica Direta (ou OWL 2 DL)**: a Semântica Direta (do inglês, *Direct Semantics*) provê significado para as ontologias em OWL por meio da Lógica de Descrição. É por esta razão que a semântica direta é também associada à OWL 2 DL, em que DL tem o significado de *Description Logic*[17].
2. **Semântica baseada em RDF (ou OWL 2 FULL)**: a Semântica Baseada em RDF (do inglês, *RDF-Based Semantics*) é uma extensão da semântica presente no Esquema RDF e é utilizada para a visualização de grafos RDF.

De um modo geral, podemos enxergar a semântica presente em OWL como OWL 2 DL ou OWL 2 FULL, em que algumas diferenças podem ser identificadas:

- OWL 2 DL tem um estilo baseado na Lógica de Descrição e OWL 2 FULL tem um estilo baseado em Grafos RDF.
- OWL 2 DL está relacionada com a Semântica Direta enquanto OWL 2 FULL, com Semântica baseada em RDF.
- OWL 2 DL é uma versão mais simplificada e restrita que OWL 2 FULL.

16 Não é objetivo deste documento dissertar em detalhes as características sobre a Lógica de Descrição. Para isso, veja (Donini, 2003).

17 É importante frisar que é possível interpretar OWL 2 DL com a Semântica baseada em RDF, porém a OWL 2 DL está comumente vinculada à Semântica Direta.

- OWL 2 DL (Semântica Direta) é decidível (ou seja, responde a uma pergunta em tempo finito) enquanto OWL 2 FULL (Semântica baseada em RDF) é indecidível.

É importante frisar que, na versão 1.0 da OWL, havia as propostas de OWL 1 DL e OWL 1 FULL, bem como a proposta OWL 1 Lite, que, de um modo geral, pode ser entendida como uma versão simplificada da OWL 1 DL.

No entanto novos conceitos importantes para a semântica foram adicionados à versão 2.0 da OWL. A grande diferença está na definição de perfis para OWL 2, tendo sido adicionados principalmente pela complexidade de implementação de OWL. Com isso, os perfis foram projetados para atender a necessidades de uso e capacidade computacional e, como consequência, apresentam diferentes níveis de expressividade. A Figura 3.6 apresenta os diferentes níveis de expressividade da linguagem OWL e uma comparação de suas versões (Open Semantic Framework, 2014).

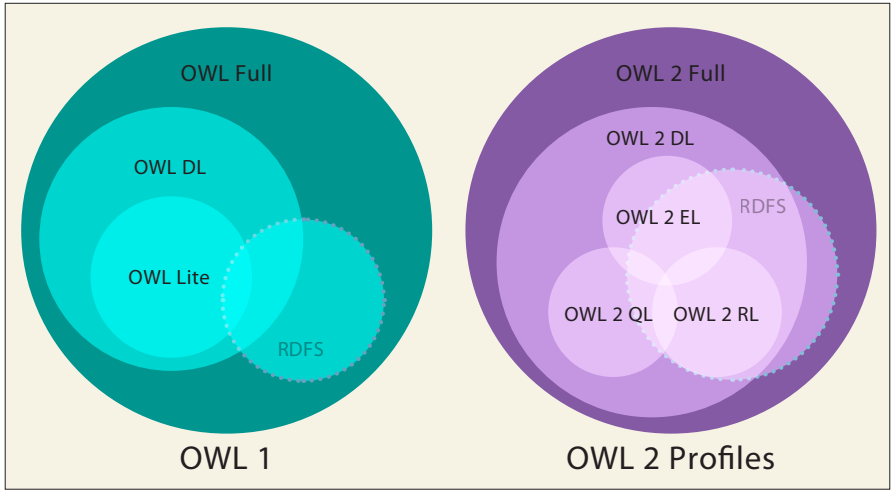

Figura 3.6 – Perfis e expressividades das versões OWL 1 e OWL 2. Fonte: adaptado de Open Semantic Framework (2014).

De acordo com a Figura 3.6, foram especificados três perfis para a versão 2.0 da OWL, sendo eles (Motik et al., 2014):

- **OWL 2 EL**: esse perfil é voltado para trabalhar com a família EL (Baader et al., 2005), da lógica de descrição[18]. Desta forma, OWL 2 EL pode ser usada para especificar classes complexas e axiomas sobre elas. Esse perfil foi projetado para sistemas de alta complexidade e necessidade de maior poder de expressividade. Além disso, apesar de a OWL 2 EL ser independente de domínio, ela foi concebida para atender a domínios com um grande número de classes que tenham objetos estruturalmente complexos, como configuração de sistemas, inventário de produtos, estruturas biológicas e muitos domínios científicos. No entanto esse perfil não permite a utilização de construtores como negação (`owl:ObjectComplementOf`) e disjunção (`owl:DisjointClasses`)[19], quantificadores universais (`owl:ObjectAllValuesFrom`) sobre as propriedades e propriedades inversas (`owl:InverseObjectProperties`). Dessa forma, a seguinte construção ("todos os filhos de uma pessoa rica são ricos") não pode ser especificada:

```
EquivalentClasses (
   :RichPerson
   ObjectAllValuesFrom ( :hasChild :RichPerson )
)
```

  Por outro lado, a família de linguagens EL provê principalmente a utilização de quantificadores existenciais. Sendo assim, este exemplo ("os pais têm pelo menos um filho") é permitido:

```
EquivalentClasses (
   :Parent
   ObjectSomeValuesFrom ( :hasChild :Person )
)
```

- **OWL 2 QL**: esse perfil é voltado para o trabalho com bancos de dados relacionais tradicionais (isto é, que fazem uso da linguagem SQL). Como consequência, trabalhar com OWL 2 QL permite à

18 A lógica de descrição é uma linguagem atributiva que, à medida que aumenta os seus atributos, aumenta a sua expressividade. Por exemplo, a linguagem OWL 2 DL é da família ALIC.

19 De fato, tanto o perfil OWL 2 EL não tem os construtores de negação e disjunção como todos os outros perfis.

equipe os benefícios providos pelos Sistemas Gerenciadores de Bancos de Dados Relacionais, como uma implementação robusta e características multiusuários. Ou seja, o OWL 2 QL é um perfil independente de domínio, que foi projetado para permitir a especificação de esquemas de bancos de dados e integração via consultas. Esse perfil pertence à família DL-Lite, podendo, dessa forma, reescrever consultas em SQL (Calvanese et al., 2007). Nesse perfil, além dos construtores de negação e disjunção, não é permitida a utilização de quantificadores existenciais em classes complexas (apenas de classes simples), bem como axiomas para encadeamento de propriedades e igualdade. Dessa forma, a construção a seguir ("toda pessoa tem um pai que é mulher") não pode ser especificada, pois mulher, que é um dos parentes (sendo parente como um dos pais), é uma classe complexa.

```
EquivalentClasses (
   :WomenParent
   ObjectSomeValuesFrom ( :hasParent :Person )
)
```

Assim, este exemplo ("toda pessoa tem um pai") é permitido:

```
EquivalentClasses (
   :Parent
   ObjectSomeValuesFrom ( :hasParent :Person )
)
```

- **OWL 2 RL**: esse perfil é voltado para aplicações que necessitem de raciocínio de forma escalável. Além disso, esse perfil suporta tanto a semântica direta quanto a semântica baseada em RDF. Dessa forma, OWL 2 RL é o perfil ideal para o enriquecimento de dados especificados e conectados via RDF, pois pode ser implementado usando famílias de linguagens de regras (Grosof et al., 2003), como por meio da utilização de RIF (do inglês, *Rule Interchange Format*). Conclui-se então que, para aplicações voltadas para Dados Abertos Conectados, o perfil OWL 2 RL é mais adequado. Ou seja, diferentemente dos outros dois perfis, este perfil é mais

adequado quando os projetistas já têm uma modelagem em RDF e estão trabalhando com RDF. No entanto OWL 2 RL não permite declarações em que a existência de um indivíduo implica na existência de outro. Dessa forma, a construção a seguir ("toda pessoa tem um pai") não pode ser especificada, pois a existência de uma pessoa implica na existência de um pai.

```
EquivalentClasses (
   :Person
   ObjectSomeValuesFrom ( :hasParent :Parent ) )
```

Por outro lado, este exemplo ("a propriedade tio é definida pela relação entre o pai e o irmão do pai") é permitido:

```
SubPropertyOf (
   ObjectPropertyChain ( :hasFather :hasBrother )
   :hasUncle
)
```

Por fim, a Figura 3.7 apresenta parte da semântica baseada em RDF, que pode ser usada no perfil OWL 2 RL (Schneider, 2012).

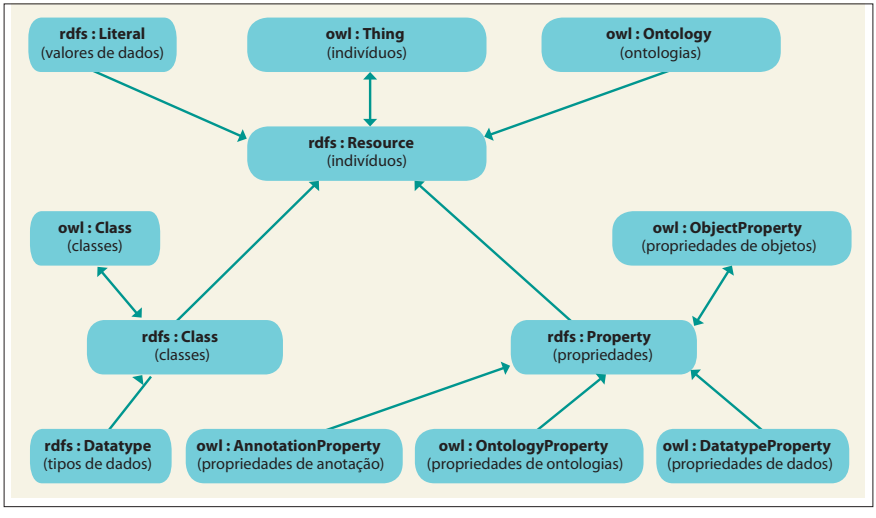

Figura 3.7 – Parte da hierarquia da semântica baseada em RDF. Fonte: adaptado de Schneider (2012).

3.4 Considerações Finais

O principal objetivo deste capítulo foi oferecer ao leitor uma visão geral sobre ontologias e representação de conhecimento na Web. Este capítulo foi organizado por meio de exemplos com o intuito de trazer para o leitor pouco acostumado com o conceito de ontologias uma perspectiva incremental de descrição semântica de forma simplificada. Foi também intenção deste capítulo apresentar as linguagens utilizadas no contexto da Web para a descrição semântica por meio de ontologias. Esperamos que as seguintes mensagens tenham sido passadas:

- Compreensão sobre o conceito de ontologias e sua importância para a Web.
- Conhecimento sobre os diferentes tipos de ontologia e de padrão da Web para descrição semântica.
- Entendimento sobre as características e diferenças presentes em RDF, RDF-S e OWL.
- Compreensão sobre a aplicação de OWL e RDF-S.

CAPÍTULO 4

Engenharia de Ontologias

4.1 Introdução

No Capítulo 3, apresentamos o conceito de ontologias e sua aplicabilidade para descrever dados conectados. Mostramos também a complexidade e dificuldade para criar uma ontologia que representa adequadamente o domínio de um conjunto de dados para que eles sejam acessíveis e reutilizáveis tanto por pessoas quanto por programas de computador. De fato, a criação de ontologias é uma tarefa importante no processo de disponibilização de dados abertos de nível 5 estrelas, mas que demanda conhecimentos específicos tanto sobre a modelagem de dados quanto do próprio domínio de conhecimento no qual os dados serão extraídos e/ou utilizados.

Dessa forma, este capítulo tem como objetivo apresentar um pouco mais sobre a área de Engenharia de Ontologias, que tem como um dos seus objetivos auxiliar no processo de criação de ontologias adequadas para satisfazer às necessidades de representação e uso correto da informação.

A Engenharia de Ontologias surgiu como uma evolução da área conhecida como Engenharia de Conhecimento (*Knowledge Engineering*), que estuda a teoria e a tecnologia para o desenvolvimento das bases de conhecimento utilizadas nos sistemas especialistas

(*expert systems*). Esta evolução ocorreu devido a diversas razões, porém podemos enfatizar três delas:

1. Pressupostos e premissas fundamentais para compreender os conceitos básicos representados são deixados implícitos nas bases de conhecimento definidas como regras de produção, impedindo a reutilização e o compartilhamento do conhecimento.
2. São poucos os esforços conduzidos na direção de criar um conhecimento genérico comum que pode ser utilizado como base para construção de outras bases de conhecimento.
3. As tecnologias para viabilizar o acúmulo incremental do conhecimento são pouco desenvolvidas.

Estas restrições inviabilizam a criação de bases de conhecimento requeridas na área da Internet, na qual o conhecimento está distribuído na Web e em constante evolução (Bittencourt et al., 2009). Assim, a Engenharia de Ontologias oferece métodos e ferramentas para sanar estes problemas. Para o problema (1) foi introduzido o conceito de ontologias e sua representação, que permitem representar de maneira explícita e sem ambiguidade os conceitos de um determinado domínio. Além disso, ontologias de topo foram desenvolvidas para prover o entendimento de conceitos básicos fundamentais (ex.: O que é objetivo? O que é um processo?) para construir novos conhecimentos e, dessa forma, ajudar a sanar o problema (2). Finalmente, para lidar com o problema, (3) diversas linguagens e ferramentas que podem lidar com diferentes tipos de expressividade e representação foram desenvolvidas, permitindo que um ecossistema fosse criado para modelagem, agregação e disseminação de conhecimento utilizando ontologias. Essas qualidades ao desenvolver ontologias são fundamentais também para disseminar dados abertos em nível 5 estrelas. Assim, nas próximas seções, discutiremos mais sobre algumas das atividades relacionadas à Engenharia de Ontologias.

4.2 Metodologias de Desenvolvimento de Ontologias

O objetivo desta seção é apresentar metodologias para o desenvolvimento de ontologias, destacando a complexidade inerente à criação de ontologias, o ciclo de vida de um processo de criação de ontologias e finalmente os tipos de metodologia existentes.

Como abordado na Seção 3.2, o termo Ontologia surgiu na Filosofia como o estudo do "ser enquanto ser", com o objetivo de estudar a natureza do ser e a estrutura da realidade. Na computação, a pesquisa em ontologias se estendeu por toda a comunidade, na qual o conceito de ontologias tem sido aplicado em diferentes áreas da computação, como Inteligência Artificial, Linguística Computacional, Teoria de Banco de Dados, Engenharia de Software, Busca, Interação Humano-Computador, entre outras (Guarino, 1997; 2010). É importante frisar que a aplicação tem ocorrido em diferentes áreas do conhecimento como Medicina, Comércio Eletrônico, Direito, Biologia, Educação, entre outras.

A importância que as ontologias têm recebido por diferentes comunidades ocorre devido ao seu propósito de descrever um conjunto de conceitos fundamentais e suas relações. Por um lado, ontologias podem ser utilizadas para diferentes propósitos, sendo aplicadas em diferentes domínios e reusadas por diferentes comunidades. Por outro lado, ontologias podem ser projetadas com diferentes abordagens, formalizadas de diferentes maneiras, expressadas com diferentes coberturas e reusadas por diversas razões.

Desta forma, a construção de ontologias envolve diferentes atividades, como análise conceitual e modelagem de domínio. De fato, construção de ontologias envolve peculiaridades arquiteturais e metodológicas (Holanda et al., 2013). Em uma perspectiva arquitetural, o aspecto mais importante é a função central que uma ontologia tem. Em uma perspectiva metodológica, a abordagem altamente interdisciplinar é importante, na qual a filosofia e a linguística têm

papéis fundamentais, com o objetivo de analisar uma determinada estrutura da realidade e formular um vocabulário rigoroso e claro (Guarino, 1998).

4.2.1 Complexidade na criação e ontologias

De acordo com (Gruber, 1993), ontologia é a especificação explícita de uma conceitualização. Para o termo conceitualização, Gruber se apoiou em definição (Genesereth et al., 1987), na qual conceitualização representa uma visão abstrata e simplificada do mundo que se pretende representar com determinado propósito. No entanto tal definição é baseada em uma relação ordinária (*extensional*[1]) de objetos, significando que esta relação reflete um estado particular[2] do mundo. Entretanto Guarino (Guarino et al., 1994) propôs uma adaptação na definição de conceitualização, na qual o foco é dado para o significado da relação em vez do estado particular do mundo. Com isso, essa relação é conhecida como relação conceitual[3].

Na Figura 4.1 (Guarino et al., 1995), apresentam-se duas mesas com diferentes organizações de blocos. Na visão de Gruber, cada figura representa um objeto diferente, pois tem diferentes estados do mundo. Para Guarino, as duas mesas significam a mesma coisa, mudando apenas a composição dos blocos. Ou seja, apresentam a mesma conceitualização.

É muito importante enfatizar que uma relação conceitual é sempre definida em um espaço de domínio, pois o estabelecimento de conceitos pode ter diferentes significados de acordo com o universo de discurso (ou mundo possível). Na Figura 4.2, pode-se observar que o conceito "BANK" tem diferentes significados de acordo com o universo de discurso.

1 Do inglês, *Extensional Relation*.
2 Do inglês, *State of Affair*.
3 Do inglês, *Intensional Relation* ou *Conceptual Relation*.

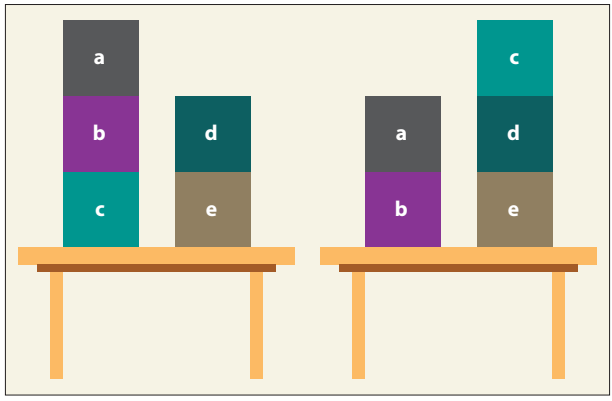

Figura 4.1 – Blocos em uma mesa: diferentes arranjos e a mesma conceitualização. Fonte: adaptado de Guarino et al. (1995).

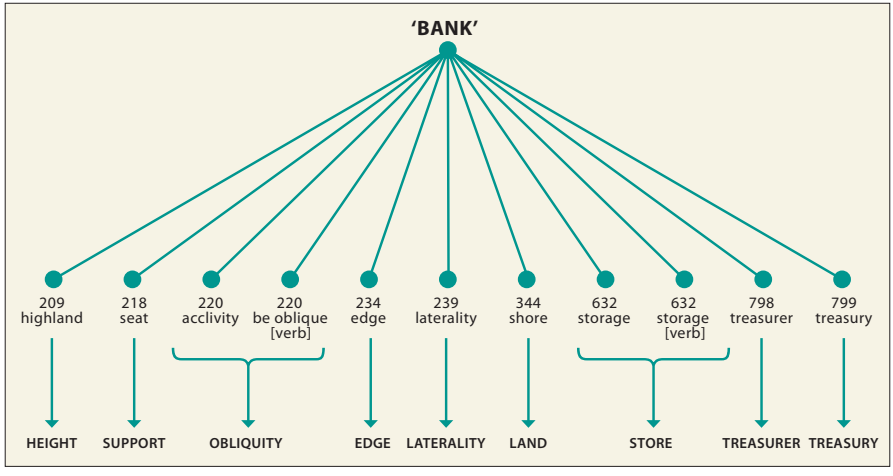

Figura 4.2 – Diferentes significados para a mesma palavra. Fonte: adaptado de Sowa (2006).

Diante disso, pode-se observar que o cenário de complexidade na construção de ontologia é grande, em que cada palavra traz diferentes significados de acordo com o universo do discurso em que se pretende estabelecer uma determinada teoria lógica. A Figura 4.3 sumariza isso, na qual uma palavra pode representar diferentes sentidos; um sentido pode ser representado por relações conceituais, em que diferentes relações conceituais podem estabelecer o mesmo

sentido; um sentido ou uma palavra pode ter diferentes grafos canônicos, representando um padrão no qual o conceito ocorre; o grafo canônico expressa uma teoria lógica, que caracteriza estruturas matemáticas, domínios de conhecimento ou simplesmente fragmentos de pensamentos ou ideias.

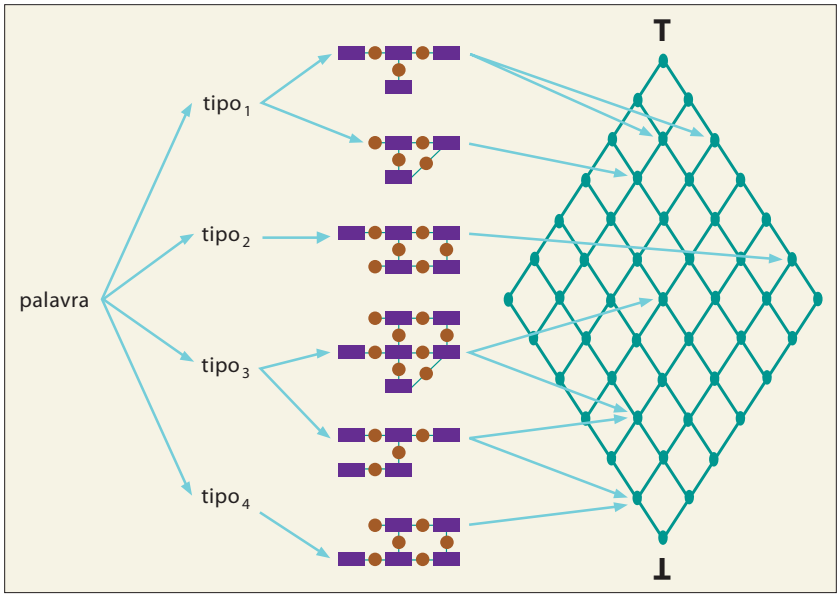

Figura 4.3 – Palavra Tipos Grafos Canônicos. Teoria Lógica. Fonte: adaptado de Sowa (2006).

Destaca-se então que uma conceitualização é estruturada de acordo com os padrões apresentados e percebidos de uma determinada realidade ou um fenômeno que se observa. Tal fenômeno é observado por um conjunto de agentes (comumente humanos) que mapeia os padrões de apresentação em partes. Estas partes constituem a reificação de objetos que são finalmente modelados para representar uma conceitualização. Esta conceitualização é organizada independentemente de um vocabulário e uma instanciação específica. Ou seja, dado que uma conceitualização tem diferentes estruturas de mundos possíveis, essas estruturas apresentam modelos descritos por meio de uma linguagem lógica com um vocabulário específico.

Com isso, cada modelo tem uma interpretação conceitual que é fixada pelo compromisso ontológico para uma linguagem. Como resultado, uma linguagem se compromete com (implementa) uma conceitualização por meio do compromisso ontológico. Finalmente, uma ontologia é um conjunto de axiomas lógicos projetados para representar os modelos esperados de um vocabulário (Guarino et al., 1994). Neste sentido, ontologias podem ser construídas utilizando diferentes vocabulários, enquanto compartilham a mesma conceitualização. Ou seja, uma conceitualização é independente de linguagem, enquanto uma ontologia é dependente de linguagem. Este percurso do Fenômeno à Ontologia é representado na Figura 4.4.

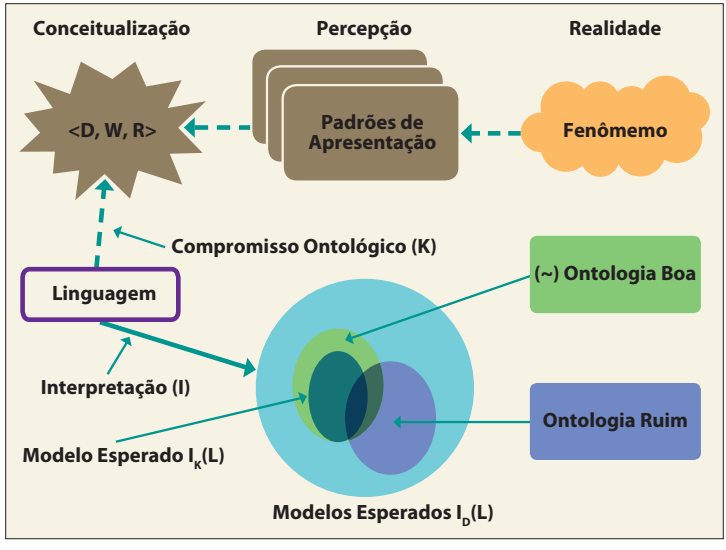

Figura 4.4 – Percepção, concepção e representação de um domínio. Fonte: adaptado de Guarino (2010).

Como se pode observar na Figura 4.4, o percurso que se deve fazer, desde a observação e percepção de um determinado fenômeno até a construção de uma ontologia, é muito longo, podendo haver grandes perdas conceituais ao longo deste processo. No entanto, da observação do mundo real para a conceitualização, ocorre perda devido à limitação do agente observador em compreender em sua

totalidade os componentes do fenômeno observado. Da mesma forma, do estabelecimento de uma conceitualização à construção de uma ontologia, também há perda por causa da limitação da própria linguagem utilizada (vide Figura 4.5).

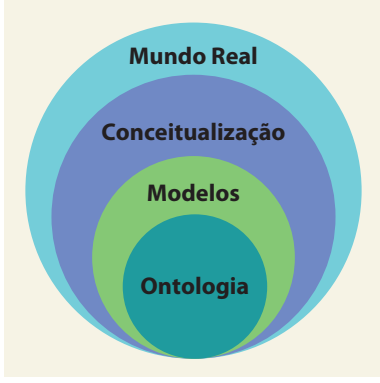

Figura 4.5 – Perda desde a percepção do mundo até a representação do conhecimento. Fonte: autores.

O grande desafio na construção de ontologias é fazer com que haja um estreitamento e uma aproximação da ontologia com o mundo real. Quanto mais distante uma ontologia estiver do modelo que se espera, pior será a qualidade da ontologia. É nesta perspectiva que metodologias para a engenharia de ontologias são utilizadas. A próxima subseção apresenta tipos de metodologia que podem ser usados para a construção de ontologias.

4.2.2 Tipos de Metodologia

A engenharia de ontologias engloba um conjunto de atividades para a conceitualização, o projeto, a implementação e a evolução de ontologias. Já as metodologias para a engenharia de ontologias estabelecem, de forma sistemática, um conjunto de atividades desde a produção até a manutenção de uma ontologia. Estas metodologias comumente consideram atividades de gerenciamento (ex.: agendamentos),

desenvolvimento (e.g., conceitualização e formalização) e suporte (ex.: documentação, alinhamento). Ou seja, as metodologias consideram atividades presentes no ciclo de vida de uma ontologia. A Figura 4.6 apresenta o ciclo de vida de uma ontologia.

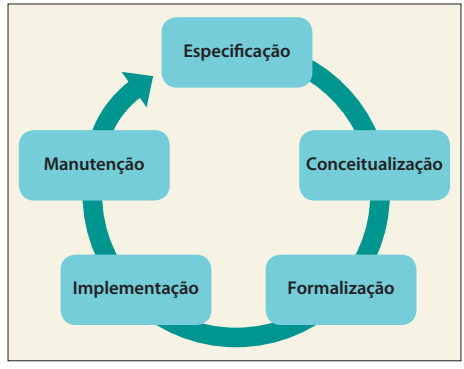

Figura 4.6 – Ciclo de vida de uma ontologia. Fonte: autores.

É importante frisar que o ciclo de vida de uma ontologia estabelece o conjunto de estágios que serão seguidos ao longo do desenvolvimento de uma ontologia. Isso quer dizer que o ciclo de vida de uma ontologia pode variar de acordo com a metodologia a ser utilizada. A seguir é descrita cada etapa presente na Figura 4.6 (Calero et al., 2006).

- **Especificação**: esta etapa estabelece o início das atividades, mostrando por que uma ontologia será construída, quais os possíveis usos e quem são os usuários interessados na ontologia.
- **Conceitualização**: esta etapa organiza e estrutura o conhecimento do domínio em que se pretende construir a ontologia. Nesta etapa, atividades de aquisição de conhecimento podem ser consideradas.
- **Formalização**: na etapa de formalização é construído o modelo conceitual, estabelecendo os conceitos, as relações e os axiomas presentes na ontologia[4].

4 Algumas metodologias podem considerar as etapas de conceitualização e formalização como um único estágio do ciclo de vida.

- **Implementação**: com a utilização de ferramentas CASE (vide Subseção 4.4), a ontologia é construída em uma determinada linguagem.
- **Manutenção**: com a ontologia implementada em uma linguagem, ocorre a etapa de manutenção da ontologia.

De acordo com a IEEE[5], uma metodologia é uma série integrada de técnicas e métodos, criando uma teoria de sistemas geral sobre como uma classe de trabalho intelectual intensivo deve ser executada. É importante enfatizar que a área de ontologias teve seu nascedouro na computação na área de Inteligência Artificial, mais especificamente na área de representação de conhecimento. Isso fez com que as primeiras propostas de metodologias para a construção de ontologias fossem inspiradas na Inteligência Artificial. Ao longo do tempo, o processo de construção de ontologias considerou a construção como um processo de desenvolvimento de software, sendo assim inspirada na Engenharia de Software. Por fim, outras metodologias que consideram o envolvimento maior do usuário começam a surgir e são inspiradas na Interação Humano-Computador (IHC). A seguir são descritas algumas metodologias (Casellas, 2011):

- **Metodologias Inspiradas na Inteligência Artificial**: nestas metodologias, os processos estão mais próximos aos processos de aquisição de conhecimento, bem como às atividades que foquem na aplicação de técnicas de aquisição do conhecimento. Algumas metodologias são TOVE, Enterprise Model Approaches e IDEF5.
- **Metodologias Inspiradas na Engenharia de Software**: estas metodologias têm seu processo inspirado nos processos de desenvolvimento de software, bem como em metodologias da própria engenharia de software. As metodologias para a construção de ontologias foram inspiradas tanto em modelos de processos tradicionais da engenharia de software, como o RUP (*Rational Unified Process*), quanto em metodologias ágeis, como XP (*eXtreme*

5 *www.ieee.org*

Programming). Algumas metodologias para a construção de ontologias são METHONTOLOGY, Ontology Development 101 e RapidOWL.

- **Metodologias Inspiradas na Interação Humano-Computador**: estas metodologias têm um envolvimento maior do usuário final e consideram atividades de colaboração e negociação no processo de construção da ontologia. Estas metodologias são inspiradas na IHC pelo fato de que o foco da construção da ontologia está na interação. Algumas metodologias inspiradas na IHC são DILIGENT e HCOME.

Independentemente da metodologia que a equipe for utilizar, alguns aspectos são sempre comuns a todas elas, como: i) respeitam um determinado ciclo de vida – toda metodologia vai apresentar um ciclo de vida, com estágios e atividades bem definidas para guiar a equipe de desenvolvimento da ontologia; ii) tarefas de aquisição do conhecimento – compreender e extrair os aspectos importantes de um domínio são fundamentais; iii) aspectos informais primeiro – toda metodologia vai considerar primeiro aspectos informais para posteriormente haver uma especificação formal da ontologia; iv) reúso e compartilhamento – apesar de algumas metodologias não explicitarem ou estabelecerem atividades de reúso e compartilhamento, todas elas abordam a importância de considerar o reúso e o compartilhamento; e v) recomendação – sempre haverá uma série de recomendações a serem seguidas à medida que a equipe avançar no processo de construção de uma ontologia.

4.3 Ferramentas

Atualmente existem dezenas de editores de ontologias desenvolvidos pela comunidade. De acordo com (Alatrish, 2012), alguns dos editores frequentemente adotados para criação e manipulação de

ontologias são: *Apollo* (KMI, 2014), *OntoStudio* (Semafora, 2014), *Protégé* (Stanford, 2014), e *TopBraid Composer* (TopQuadrant, 2014). Além destes, o editor *Hozo* (Mizoguchi-Lab, 2014) também é frequentemente utilizado por pesquisadores na Ásia. Dentre estes, o *Protégé* é o mais conhecido e utilizado pela comunidade internacional e será selecionado para desenvolver as ontologias presentes ao longo deste capítulo. Uma comparação geral das ferramentas adaptando o trabalho de (Alatrish, 2012) é apresentada na Tabela 4.1.

Tabela 4.1 – Comparação entre editores de ontologias

	Apollo	Hozo	OntoStudio	Protégé	TopBraid
Desenvolvedor					
Licença	Código Aberto	Software Proprietário	Software Proprietário	Código Aberto	Software Proprietário
Extensibility	Plugins	Não	Plugins	Plugins	Plugins
Import/export	*Apollo Meta language*, OCML, CLOS	XML(S), RDF(S), OWL, outros	XML(S), RDF(S), OWL, Diagrama UML, outros	XML(S), OWL, RDF(S), Excel, outros	RDFa, WOL, RDF(s), XHTML, Excel, outros
Taxonomia gráfica	Não	Sim	Sim	Sim	Não
Trabalho colaborativo	Não	Sim	Sim	Sim	Sim
Permite Zoom	Não	Sim	Sim	Sim	Não

4.4 Criando uma Ontologia

O objetivo desta seção é descrever de forma simplificada a criação de uma ontologia por meio da utilização de uma metodologia. Para isso utilizaremos a Metodologia *Ontology Development 101*[6] (Noy et al., 2001), uma metodologia simples e bastante usada para a construção de ontologias.

6 Apesar de vários pesquisadores da área de ontologia não considerarem a Ontology Development 101 como adequada para a criação de ontologias, esta é uma das mais citadas pela literatura e oferece um conjunto de passos simples e de fácil compreensão para alguém que pretende criar ontologias pela primeira vez.

4.4.1 Metodologia 101

A metodologia 101 (ou *Ontology Development 101*) foi criada por pesquisadores da Universidade de Stanford e é a metodologia mais utilizada para a construção de ontologias, atualmente. Uma das razões que justifica a grande utilização da metodologia é a sua simplicidade. Esta metodologia foi construída baseada na experiência dos autores com a utilização dos ambientes de edição de ontologias Protégé, Ontolingua e Chimaera.

A metodologia é composta de sete passos objetivos e simples que guiam os engenheiros de ontologias no processo de construção de uma ontologia (vide Figura 4.7).

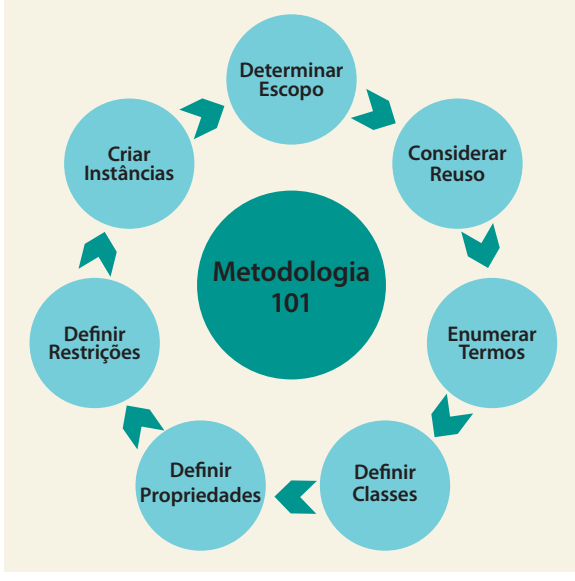

Figura 4.7 – Ciclo de vida na metodologia 101. Fonte: adaptado de Noy et al. (2001).

O primeiro passo presente na metodologia equivale à determinação do domínio e escopo da ontologia. Nesta etapa os engenheiros devem estabelecer questões básicas que servirão para compreender o propósito da construção da ontologia. Questões básicas que são levantadas como (Noy et al., 2001):

- Qual o domínio que a ontologia deve cobrir?
- Quais serão os usos da ontologia?
- Quem irá usar e manter a ontologia?
- A quais tipos de questão a ontologia deveria responder?

É importante enfatizar que é natural que as respostas mudem ao longo do desenvolvimento da ontologia. Com relação à última questão, ela tem o propósito de definir questões de competência da ontologia (Gruninger et al., 1995), em que os engenheiros elaborarão questões específicas sobre o domínio. Estas questões serão utilizadas tanto para estabelecer um limite do escopo da ontologia como servirão de base para a validação da ontologia.

O próximo passo está relacionado ao reúso de outras ontologias para apoiar na construção da ontologia de interesse. Esta etapa é fundamental e nunca deve ser negligenciada, pois é por meio do reúso de outras ontologias que a Web de Dados pode ser mais enriquecida e conectada. É importante frisar que o reuso não deve ocorrer exclusivamente de toda ontologia, mas sim de conceitos específicos que os engenheiros considerarem similares com a ontologia que está sendo construída. A Tabela 4.2 apresenta alguns vocabulários que podem ser considerados para reúso pelos engenheiros de ontologias.

Tabela 4.2 – Lista de vocabulários

Nome	Prefixo	URI	Propósito
Bio	bio:	*http://purl.org/vocab/bio/0.1/*	Informação bibliográfica
Creative Commons Rights Expression Language	cc:	*http://creativecommons.org/ns#*	Licenças
DOAP	doap:	*http://usefulinc.com/ne/doap#*	Projetos
Dublin Core Elements	dc:	*http://purl.org/dc/elements/1.1/*	Publicações
Dublin Core Terms	dct:	*http://purl.org/dc/terms/*	Publicações
FOAF	foaf:	*http://xmlns.com/foaf/0.1/*	Pessoas

Nome	Prefixo	URI	Propósito
Geo	pos:	*http://www.w3.org/2003/01/geo/wgs84_pos#*	Posições
GeoNames	gn:	*http://www.geonames.org/ontology#*	Localizações
Good Relations	gr:	*http://purl.org/goodrelations/v1#*	Produtos
Object Reuse and Exchange	ore:	*http://www.openarchives.org/ore/terms*	Recursos de mapa
SIOC	sioc:	*http://rdfs.org/sioc/ns#*	Comunidades online
SKOS	skos:	*http://www.w3.org/2004/02/skos/core#*	Vocabulários controlados
vCard	vcard:	*http://www.w3.org/2006/vcard/ns#*	Cartão de visita
VoID	void:	*http://rdfs.org/ns/void#*	Vocabulários
WordNet	wn:	*http://xmlns.com/wordnet/1.6/*	Palavras em inglês
Food	food:	*http://www.w3.org/TR/2003/PR-owl-guide-20031215/food#*	Alimentos
DCAT	dcat:	*http://www.w3.org/ns/dcat#*	Catálogo de dados
PROV-O		*http://www.w3.org/ns/prov#*	Proveniência de dados
ORG	org:	*http://www.w3.org/ns/org#*	Estruturas organizacionais
QB	qb:	*http://purl.org/linked-data/cube#*	Dados multidimensionais
SCOVO	scv:	*http://purl.org/NET/scovo*	Dados estatísticos
Hydra Core Vocabulary	hydra:	*http://www.w3.org/ns/hydra/core#*	APIs
Internationalization Tag Set	its:	*http://www.w3.org/2005/11/its*	Internacionalização

Após a definição de quais ontologias poderão ser reusadas, a equipe deve enumerar os conceitos que devem estar na ontologia. Nesta etapa, não se deve considerar se um determinado termo é uma classe, uma propriedade ou um elemento que deverá ser reusado de outra ontologia. Isso será feito nas etapas posteriores. A próxima etapa é dedicada à definição das classes e hierarquias de classes. A estratégia de desenvolvimento da hierarquia de classes vai depender da equipe, podendo ser Top-Down, Bottom-Up ou a combinação destes (Uschold et al., 1996).

As duas etapas seguintes são para definir as propriedades presentes nas classes, bem como as restrições que cada propriedade e classe

deve ter. É muito importante que a equipe de engenharia de ontologias não negligencie a etapa de definição de restrições. Devido ao nível de granularidade fino, no qual a equipe deverá observar cada termo e suas restrições, isso costuma ser relaxado e, consequentemente, o compromisso ontológico é perdido.

Por fim, a equipe criará as instâncias que estarão presentes na ontologia. É importante frisar que esta etapa poderá ocorrer de três diferentes formas:

- **Instâncias obrigatórias**: algumas instâncias podem ser obrigatórias e, desta forma, farão parte da própria ontologia que está sendo criada.
- **Testes**: algumas instâncias podem ser criadas com o objetivo de testar a ontologia que está sendo criada, com o objetivo de responder a questões de competências definidas na primeira etapa do processo. Estes testes ocorrerão em conjunto com a criação de consultas.
- **Mapeamento**: as instâncias de uma ontologia podem vir de uma fonte externa de dados que podem ser mapeados de acordo com a ontologia. Esta última é mais comum quando se pretende publicar dados de forma conectada. Neste caso, a publicação de dados irá considerar o reúso da ontologia que foi criada.

A próxima subseção descreverá o processo de criação de uma ontologia utilizando a metodologia 101.

4.4.2 Cenário 1: Wine Ontology (Ontologia de Vinho)

O cenário que será construído é do domínio de vinhos e também foi utilizado como exemplo em diversos tutoriais sobre a construção de ontologias. A ontologia está disponível no link que segue:

http://www.w3.org/TR/2003/PR-owl-guide-20031215/wine#

1. **Determinar o domínio e o escopo da Ontologia**: o domínio da ontologia equivale ao de vinho e a ontologia está sendo construída para aplicações que façam recomendações de vinhos em restaurantes. Um aspecto importante ao se recomendar vinho é que há uma dependência do tipo de alimento que irá acompanhar o vinho. Consequentemente, para que haja uma boa recomendação de vinho, deve-se considerar na ontologia os alimentos. Algumas questões de competência que devem ser consideradas na ontologia são:
   - **i.** Quais características do vinho devem ser consideradas ao recomendar um vinho?
   - **ii.** O Bordeaux é um vinho branco ou tinto?
   - **iii.** O Cabernet Sauvignon combina com frutos do mar?
   - **iv.** Qual o melhor vinho para carne grelhada?
2. **Considerar reúso de ontologias existentes**: com o estabelecimento do escopo da ontologia, pode-se perceber que ontologias relacionadas a alimentos e vinhos podem ser reusadas. Uma ontologia disponível que pode ser reusada é a ontologia *food*, disponível em: *http://www.w3.org/TR/2003/PR-owl-guide-20031215/food#*

   A ontologia Food contém 65 classes, 8 propriedades e 57 instâncias. Parte da Ontologia pode ser visualizada na Figura 4.8.
3. **Enumerar importantes termos**[7] **na Ontologia**: diversos termos são considerados para esta ontologia, tanto relacionados a vinho quanto a alimentos, como: *wine, grape, winery, location, wine's color, body, white wine, red wine, rosé wine, syrah, cabernet sauvignon, medoc, bordeaux, flavor, fish, red meat*, entre outros[8]. É muito importante para esta etapa consultar fontes confiáveis para a aquisição de conhecimento relacionado a vinhos.

7 Apesar de esta etapa objetivar a enumeração de termos, como descrevemos no Capítulo 3, os ontologistas devem se preocupar com os conceitos e não exclusivamente com os termos.

8 Não é objetivo desta subseção descrever a ontologia de vinhos por completo, mas apenas ilustrar o processo de construção da ontologia.

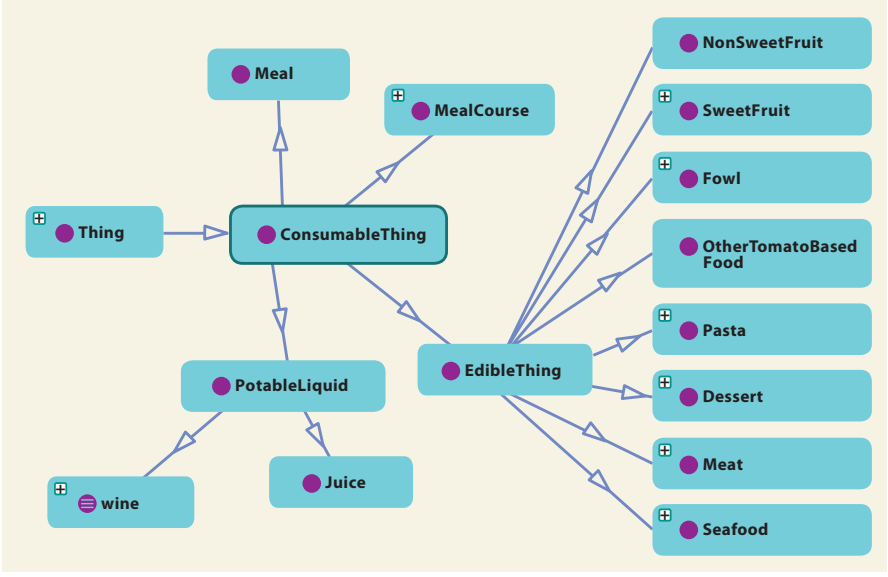

Figura 4.8 – Parte da hierarquia de classes da ontologia Food. Fonte: adaptado de Noy et al. (2001).

4. **Definir classes e hierarquia de classes**: com os termos enumerados, há aqui a identificação das classes da ontologia, bem como de sua hierarquia. Em uma abordagem Top-Down (de cima para baixo), pode-se identificar que a classe `Wine` é provavelmente um dos conceitos mais gerais. Ao especializar a classe `Wine`, pode-se considerar como suas subclasses os tipos de vinho, neste caso *White Wine*, *Red Wine* e *Rosé Wine* (vide Figura 4.9). Seguindo na especialização de classes, pode-se estabelecer as subclasses de cada tipo de vinho. Por exemplo, como subclasses de *Red Wine* se têm *Syrah, Cabernet Sauvignon, Red Bordeaux, Pinot Noir, Chianti*, entre outros. Este processo deve ocorrer recursivamente até o nó folha.

5. **Definir as propriedades das classes**: algumas propriedades que são definidas estão relacionadas com a própria composição dos vinhos como *body, flavor, sugar contente, location*, entre outras (vide Figura 4.10).

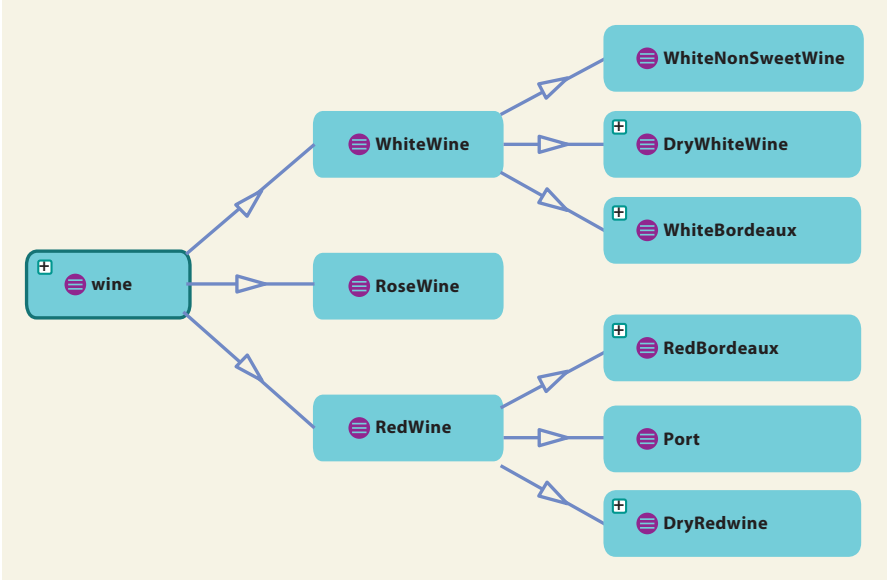

Figura 4.9 – Parte da hierarquia de classes da ontologia Wine. Fonte: adaptado de Noy et al. (2001).

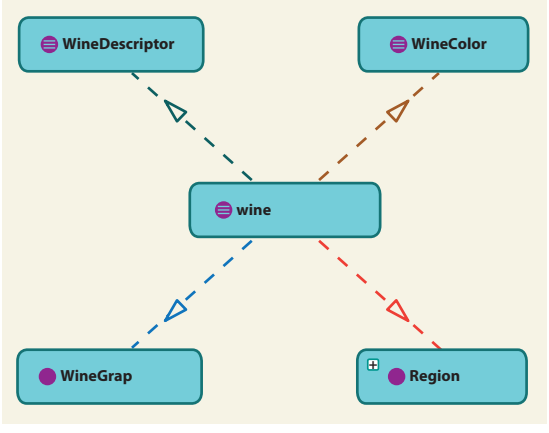

Figura 4.10 – Parte da ontologia Food com relacionamentos entre classes. Fonte: adaptado de Noy et al. (2001).

6. **Definir as restrições**: esta é uma das etapas mais trabalhosas e, muitas vezes, é negligenciada. Algumas restrições relacionadas a classe Wine são de cardinalidade – como um vinho é feito somente por uma vinícola ou todo vinho americano é localizado na região dos Estados Unidos. A Tabela 4.3 apresenta uma matriz com as propriedades e suas restrições.

Tabela 4.3 – Parte de restrições da ontologia wine

Propriedade	Funcional	Simétrica	Transitiva	Inversa
madeIndoWine	Não	Não	Não	madeFromGrape
madeFromGrape	Não	Não	Não	madeIntoWine
hasMaker	Sim	Não	Não	producesWine
producesWine	Não	Não	Não	hasMaker
adjacentRegion	Não	Sim	Não	-
hasVintageYear	Sim	Não	Não	-
locatedIn	Não	Não	Sim	
hasSugar	Sim	Não	Não	-
hasBody	Sim	Não	Não	-
hasFlavor	Sim	Não	Não	-
hasColor	Sim	Não	Não	-

Criar instâncias: as instâncias foram criadas tanto relacionadas com as características dos vinhos quanto com cada tipo de vinho. É importante frisar que estas instâncias são obrigatórias e fazem parte do escopo da ontologia. A Figura 4.11 apresenta algumas instâncias da ontologia de Vinho.

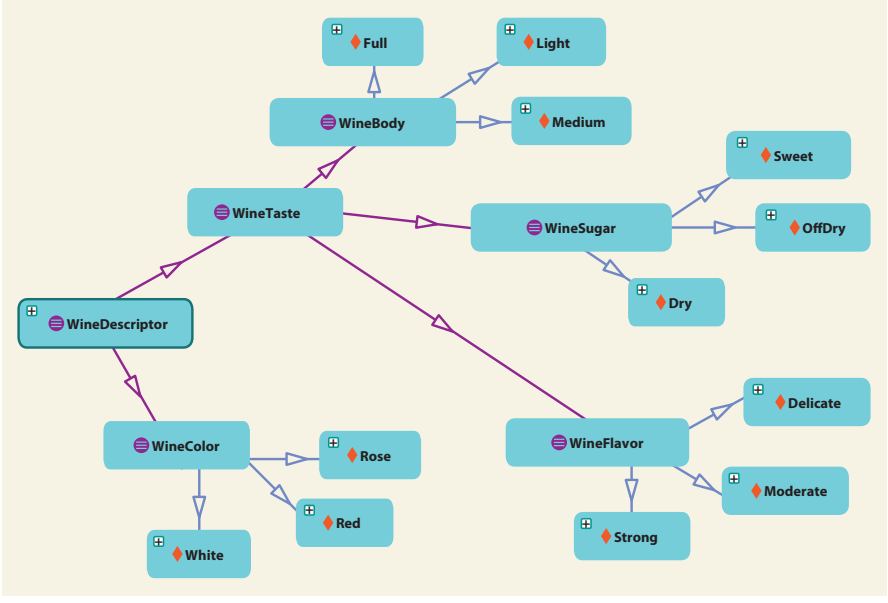

Figura 4.11 – Parte da ontologia Wine com algumas instâncias. Fonte: adaptado de Noy et al. (2001).

4.5 Considerações Finais

O principal objetivo deste capítulo foi oferecer ao leitor uma visão geral sobre engenharia de ontologias tanto do ponto de vista conceitual quanto prático. Esperamos que as seguintes mensagens tenham sido passadas:

- Compreensão sobre a importância da análise ontológica.
- Conhecimento sobre diferentes metodologias para o desenvolvimento de ontologias.
- Entendimento sobre o ciclo de vida no desenvolvimento de uma ontologia.
- Compreensão sobre como criar uma ontologia utilizando a metodologia 101.

CAPÍTULO 5

Desenvolvimento de Aplicações Semânticas

5.1 Introdução

Vimos nos capítulos anteriores que a Web atual ainda é composta de grande quantidade de páginas estáticas ou dinamicamente geradas, as quais se interligam, e que as pessoas podem lê-las e compreendê-las. Todavia os dados contidos nessas páginas não têm um significado associado de modo a possibilitar sua interpretação por parte dos computadores. Nesse cenário, abordamos os conceitos de Web Semântica, cuja ideia original é estender a Web atual, descrevendo os dados e os documentos atuais para que máquinas possam também entender e processar essa vasta coleção de informações (Cardoso et al., 2007). Nessa perspectiva, a Web Semântica trará estrutura para o significado do conteúdo presente nas páginas *web*, criando um ambiente em que agentes de *software* percorrem página por página, que proporciona a facilidade de realizar tarefas sofisticadas para os usuários finais, tais como agendamento de consultas médicas ou compras de pacotes de turismo. Tim Berners-Lee propôs um modelo de camadas que foi e ainda é revisto pelo *World Wide Web Consortium* (W3C), apresentado na Figura 1.8, no qual os principais

componentes eram RDF, RDFS, OWL e SPARQL. Já na Figura 1.9, foi apresentada uma nova versão das camadas da Web Semântica, criada por Benjamin Nowack.

Apesar de a Figura 1.9 tornar mais claras a distribuição de conceitos e a forma de uso de cada tecnologia, as formas de desenvolvimento de aplicações semânticas variam e dependem do propósito de uso. A Figura 5.1 apresenta um mapa conceitual descrevendo os conceitos e as principais ferramentas de apoio ao desenvolvimento de aplicações baseadas em Ontologias. Cada um dos conceitos apresentados na Figura 5.1 são explicados ao longo deste capítulo. O objetivo deste capítulo é apresentar como desenvolver aplicações semânticas por meio da utilização de dados abertos.

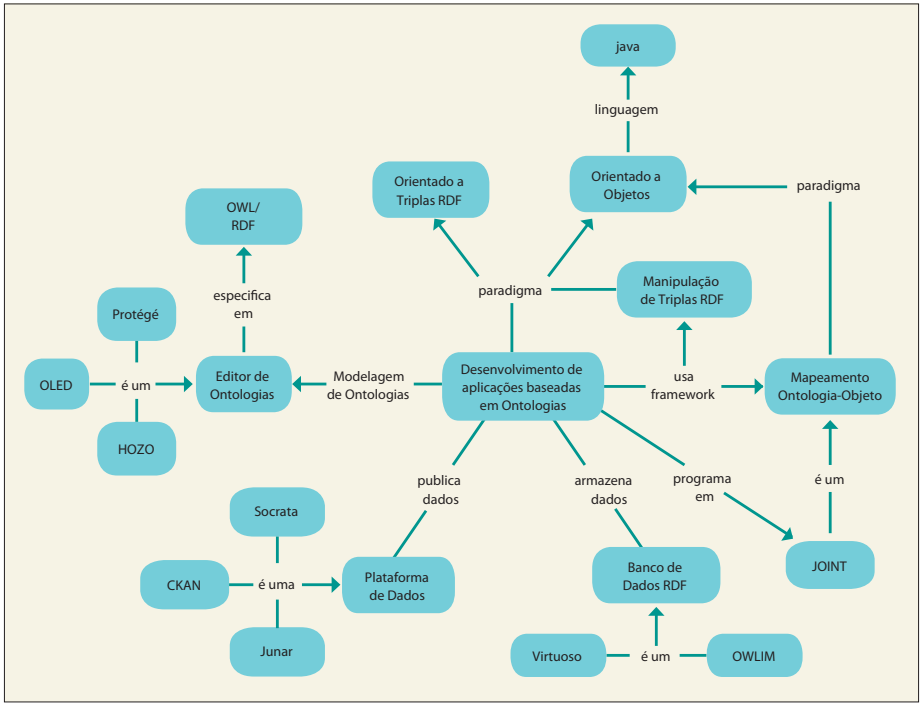

Figura 5.1 – Mapa conceitual sobre o desenvolvimento de aplicações baseadas em ontologias. Fonte: autores.

5.2 Padrões de desenvolvimento

A manipulação de instâncias é um importante passo no processo de desenvolvimento de aplicações baseadas em ontologias. Atualmente, existem duas abordagens principais utilizadas pelas ferramentas: desenvolvimento em triplas RDF e desenvolvimento orientado a objetos. Nas próximas seções, serão apresentadas as principais distinções e os benefícios das abordagens mencionadas.

5.2.1 Desenvolvimento orientado a triplas RDF

A maioria das APIs atuais ainda trabalha com o desenvolvimento baseado em triplas RDF (Holanda et al., 2013; Dermeval et al., 2015b). Dessa forma, os desenvolvedores da aplicação devem estar cientes de como funciona a ontologia em RDF para poder manipular os dados mediante cada tripla (sujeito, predicado e objeto) em código.

Para exemplificar, caso se deseje adicionar na ontologia um recurso (sujeito) com várias propriedades inerentes a ele, são necessárias diversas linhas de código representando cada tripla do recurso, as quais capturam um único valor de cada propriedade. Similarmente, caso se deseje remover esse recurso, várias triplas devem ser removidas.

Por exemplo, várias linhas de código são necessárias para criar a instância `Alice`, da classe `Person` com a propriedade de *name* tendo valor "Alice", usando a API do Sesame, que manipula instâncias mediante o desenvolvimento baseado em triplas RDF. O trecho de código a seguir ilustra como adicionar esse recurso em um repositório do Sesame.

```
...
ValueFactoryf = myRepository.getValueFactory();

//create some resources and literals to maje statements out of
URI alice = f.createURI("http://example.org/people/alice");
URI name = f.createURI("http://example.org/ontology/name");
URI person = f.createURI("http://example.org/ontology/Person");
```

```
Literal alicesName = f.createLiteral("Alice");
RepositoryConnection con = myRepository.getConnection();

//alice is a person
con.add(alice.RDF.TYPE,person);

//alice's name is "Alice"
con.add(alice,name, alicesName);
...
```

5.2.2 Desenvolvimento orientado a objetos

Diferentemente de triplas RDF, aplicações orientadas a objetos manipulam dados por meio de objetos e seus atributos. Tais objetos são caracterizados por um conjunto de atributos e valores. Nesse sentido, faz-se necessária uma ferramenta que "traduza" as operações em objetos para a infraestrutura de triplas RDF da camada inferior. Algumas ferramentas foram criadas para prover esse paradigma para manipulação de instâncias em ontologias.

Com isso, os desenvolvedores não precisarão ter um conhecimento profundo da linguagem de representação da ontologia. Um objeto no código representa uma instância na ontologia, seus atributos são mapeados com as propriedades das instâncias e as classes em RDF se tornam classes na linguagem de programação. Logo, para adicionar um recurso da ontologia, basta adicionar o objeto, facilitando, assim, o desenvolvimento dessas aplicações. Em comparação com o exemplo do Sesame (código apresentado na Seção 5.2.1), o trecho de código a seguir ilustra o mesmo recurso, sendo adicionado ao repositório do Sesame usando a ferramenta Alibaba, que permite o desenvolvimento baseado no paradigma de orientação a objetos.

```
...
ObjectConnection con = repository.getConnection();

//create a Person
Person alice = new Person();
```

```
alice.setName("Alice");

//add a Person to the repository
con.addObject(alice);
...
```

5.3 Ferramentas para desenvolvimento de aplicações semânticas

Nesta seção, serão detalhadas algumas categorias de ferramentas que ajudam o desenvolvedor a construir um sistema semântico, sendo responsáveis por determinada função na aplicação. Além disso, alguns exemplos de ferramentas em cada categoria serão apresentados e, sempre que possível, serão feitas analogias com sistemas de informação tradicionais que utilizam bancos de dados relacionais.

5.3.1 Plataformas para publicação de dados

Dentre as plataformas para publicação de dados, destacam-se três, sendo elas: CKAN, Socrata e Junar. O CKAN é uma plataforma de código aberto para publicação, compartilhamento, pesquisa e uso de dados. Com o CKAN é possível publicar, por meio de interface Web ou de uma API, dados como imagens, documentos e dados geoespaciais, permitindo que os mais diversos organismos (privados ou públicos) utilizem essa ferramenta para publicar suas informações para toda a sociedade. Assim como na publicação, a ferramenta permite que as informações armazenadas por ela sejam recuperadas por interface Web bem como por API. O CKAN é extensível, permitindo que sejam criados *plugins* para estender suas funcionalidades. Personalizável, apresenta versionamento dos catálogos de dados, permite o reúso de tecnologias existentes e é multilinguagem.

Assim como o CKAN, o Socrata é uma plataforma para publicação, compartilhamento, pesquisa e consumo de dados. Construído com as tecnologias mais recentes, o Socrata utiliza MongoDB e

Elasticsearch e disponibiliza uma API para a publicação e o consumo de dados. O Socrata disponibiliza ainda um conjunto de bibliotecas e SDK para o desenvolvimento de clientes em várias linguagens, tais como: R, PHP, Java, Objective-C e outros. Atualmente existem duas versões para o Socrata, sendo uma para a comunidade Open Source e outra para uso privado. Por fim, o Junar é uma plataforma de código fechado e pago que cobre também todas as etapas do processo de publicação de dados.

5.3.2 Frameworks para manipulação de RDF

A primeira ferramenta de que se necessita ao construir um sistema baseado em ontologias são as APIs, que oferecem a manipulação de triplas RDF e a conexão com o banco de dados RDF. Tais ferramentas funcionam como *middlewares* entre a aplicação e a ferramenta *triplestore.* Portanto elas têm a mesma responsabilidade que um *Java Database Connectivity* (JDBC), com a diferença de que, na área de ontologias, essas ferramentas geralmente têm um banco de dados RDF próprio. As APIs Sesame (Broekstra et al., 2002) e Jena (McBride, 2002) são as duas mais populares nessa categoria.

Sesame é um arcabouço Java para armazenamento e consulta em dados RDF. O Sesame é bem extensível e configurável no que diz respeito aos mecanismos de armazenamento (memória principal, arquivos binários ou banco de dados relacional), máquinas de inferência (RDFS), formatos de arquivo de ontologias (OWL, RDF ou N3) e linguagens de consulta (SPARQL e *Sesame RDF query language*). O Sesame oferece uma API para o usuário para que este possa ter acesso a seus repositórios, bem como uma interface HTTP que suporta o protocolo SPARQL. Várias extensões para o arcabouço foram desenvolvidas por terceiros.

Sesame é a principal ferramenta utilizada neste trabalho. Sua escolha deve-se à alta flexibilidade da API, que é uma das mais populares quando se trabalha com Web Semântica e ontologias. Além disso,

a interface HTTP que o Sesame provê é muito útil para visualizar os dados que estão sendo manipulados durante o desenvolvimento.

Outra vantagem do Sesame é que, por ser extensível, o mecanismo de armazenamento pode ser alterado sem que haja qualquer impacto na aplicação construída com o Sesame. Dessa forma, pode-se desfrutar de toda a potencialidade da API e usar um repositório de alta performance como OWLIM ou Virtuoso.

O Jena é um *framework* para construção de aplicações semânticas. Jena também é uma coleção de ferramentas e bibliotecas Java com o objetivo de suportar o desenvolvimento de sistemas baseados na Web Semântica. A ferramenta inclui: uma API para leitura e escrita de dados RDF em arquivos; uma API para manipulação de ontologias em OWL e RDFS; um motor de inferência baseado em regras para raciocínio; mecanismos de armazenamento de grandes volumes de dados em triplas RDF; e um motor de consulta conforme a nova especificação do SPARQL.

5.3.3 Bancos de dados RDF

A segunda categoria a ser descrita fornece as ferramentas responsáveis por armazenar RDF em algum tipo de banco de dados. Essas ferramentas são conhecidas por diversos nomes, como banco de dados RDF, *triplestore* e repositórios de ontologias, entre outros. Em comparação com sistemas de informação tradicionais, essas ferramentas são similares aos Sistemas de Gerenciamento de Banco de Dados (SGBD). Contudo apenas algumas ferramentas utilizam bancos de dados relacionais para armazenar as triplas em RDF (caso do Virtuoso); as outras utilizam um sistema de índice de arquivos (caso do OWLIM). A seguir, três bancos de dados RDF serão apresentados: OWLIM, Virtuoso e AllegroGraph.

OWLIM é uma extensão do Sesame que apresenta uma diversidade de repositórios semânticos (Kiryakov et al., 2005). Esses

repositórios têm características como: armazenamento de RDF implementado em Java; alta performance; suporte à inferência das representações RDFS e OWL; escalabilidade e balanceamento de carga. OWLIM tem três versões de repositórios:

- **OWLIM-Lite**: é o repositório de maior performance e o único grátis. Apesar de ser em memória principal, ele tem um mecanismo de persistência em arquivos binários. Além disso, ele suporta inferência com dezenas de milhões de triplas mesmo em *desktops* atuais.
- **OWLIM-SE**: é o repositório de maior escalabilidade. Esse repositório apresenta suporte à inferência e alta performance em consultas concorrentes. Ademais, ele consegue operar com o carregamento de dezenas de bilhões de triplas.
- **OWLIM-Enterprise**: é uma infraestrutura de *cluster* baseada nos repositórios do tipo OWLIM-SE. Esse repositório oferece infraestrutura escalável mediante o alto desempenho paralelizando consultas. Também oferece o balanceamento de carga e recuperação automática de falhas.

O Virtuoso é um servidor multiprotocolo que provê acesso ao banco de dados relacional interno por meio de ODBC/JDBC. Além de ter um motor de busca SQL, o Virtuoso contém um servidor HTTP para usuários administradores, com terminais em diferentes protocolos (ex.: serviços *web*) e linguagem de *script* interna (Erling; Mikhailov, 2010).

Como o propósito do Virtuoso é ser um banco de dados universal, foi realizada uma adaptação para suportar o armazenamento em triplas RDF. Nessa perspectiva, a ferramenta mapeia as triplas RDF em tabelas dentro do seu banco de dados relacional. O Virtuoso também oferece um motor de busca em SPARQL (com suporte a nova especificação 1.1), que "traduz" as consultas em SPARQL feitas pelo desenvolvedor para a correspondente em SQL.

Assim como o OWLIM, o Virtuoso tem uma versão grátis; não bastasse isso, é código aberto. A grande diferença dessa versão para as versões pagas do Virtuoso é que as pagas têm a opção de *cluster*. O Virtuoso provê *drivers* de acesso tanto usando a API do Sesame quanto a do Jena.

AllegroGraph é um banco de dados RDF moderno e de alta performance (Aasman, 2006). A ferramenta tem um mecanismo de persistência que faz uso, de maneira eficiente, da memória em combinação com o armazenamento interno baseado em disco. Dessa forma, AllegroGraph pode ser escalado para o armazenamento de bilhões de triplas RDF mantendo uma boa performance.

AllegroGraph também tem um motor de busca em SPARQL, inclusive suportando a nova especificação 1.1. Além disso, a ferramenta oferece suporte ao raciocínio em construtores RDFS e uma interface administrativa com diversos recursos. A versão grátis do AllegroGraph é limitada ao armazenamento de cinco milhões de triplas; acima disso, é necessário comprar a versão *Enterprise*.

5.3.4 Sistemas de mapeamento objeto-ontologia

Finalmente, a última categoria de ferramenta a ser detalhada nesta seção engloba os sistemas de mapeamento objeto-ontologia (OOMS). Esse tipo de ferramenta tem como função facilitar o desenvolvimento da aplicação semântica, ao permitir que o desenvolvedor foque na lógica da aplicação. Assim, não é necessário escrever códigos para conexão com o banco de dados RDF nem acessar os dados via triplas RDF, preservando as características de orientação a objetos da linguagem de programação.

Como o próprio nome refere, um sistema de mapeamento objeto-ontologia cria classes na linguagem de programação correspondentes às entidades em uma ontologia. Portanto, instâncias dessas classes podem ser representadas na aplicação como objetos. Em sistemas de

informação que usam bancos de dados relacionais, essas ferramentas são chamadas de sistemas de mapeamento objeto-relacional (ORMS), sendo alguma delas já bem consolidadas e amplamente utilizadas, como a ADO.NET (Esposito, 2002) e o Hibernate (Bauer; King, 2006). Na área de ontologia, já existem algumas ferramentas, como o Jastor, o Empire, o Elmo/Alibaba e o JOINT.

Jastor é um gerador de código Java que cria JavaBeans[1] a partir de ontologias descritas em OWL (Szekely; Betz, 2006). Com isso, os desenvolvedores podem convenientemente acessar uma ontologia armazenada em um modelo do Jena (vide Seção 2.4.2). O Jastor consegue gerar interfaces Java, suas implementações e fábricas, tudo baseado nas propriedades e na hierarquia de classes descritas na ontologia.

Empire é uma implementação da Java Persistence API (JPA) para RDF, mediante o mapeamento de objeto-ontologia, permitindo consultas de bancos de dados RDF (armazenamento em triplas) com a linguagem SPARQL ou Sesame SeRQL (Grove, 2010). JPA é uma especificação para gerenciamento de objetos Java, normalmente usada em conjunto com um RDBMS (padrão da indústria para ORMS em Java). Esse mapeamento de classes Java para triplas RDF é obtido por meio do uso de anotações JPA padrões, as quais são estendidas com algumas especificações RDF, seja para *namespaces*, classes RDF ou propriedades RDF. A ferramenta Empire provê um *framework* de persistência em Java para uso em projetos da Web Semântica, nos quais os dados são armazenados em RDF. Ao oferecer uma implementação do JPA, a ferramenta abstrai a manipulação de dados em RDF. Contudo o objetivo maior da ferramenta é substituir implementações JPA existentes para bancos de dados relacionais, ao simplificar a mudança desses sistemas para sistemas baseados em modelos RDF. O Empire faz uso da API do Sesame para manipulação em triplas RDF, mas o desenvolvedor pode configurar um *parser* para operar o Empire com a API do Jena.

1 Segundo a especificação da Sun Microsystems, os JavaBeans são "componentes reutilizáveis de software que podem ser manipulados visualmente com a ajuda de uma ferramenta de desenvolvimento".

O Elmo é um gerenciador de entidades RDF que mapeia implementações JavaBeans em triplas RDF para repositórios Sesame (Mika, 2007). Elmo provê interfaces Java estáticas para recursos RDF, ou seja, o desenvolvimento de aplicações com Elmo é feito mediante orientação a objetos centrada no sujeito. Por ser um sistema orientado a objetos, ele disponibiliza um modo de agrupar comportamentos comuns e separar papéis dentro de interfaces e classes. Os modelos gerados pelo Elmo são simples em expressar os conceitos envolvidos. O AliBaba foi desenvolvido como o sucessor do Elmo e, portanto, usa princípios similares a este para o mapeamento de objeto-ontologia. Assim como as outras ferramentas, o Alibaba é uma implementação de sistema objeto--ontologia centrada no sujeito da tripla RDF. Além disso, a ferramenta oferece implementações RESTful de bibliotecas cliente e servidor para armazenamento distribuído de documentos e metadados RDF.

Na próxima seção, discutiremos sobre o OOMS JOINT.

5.4 Desenvolvimento de uma aplicação semântica usando o JOINT

O JOINT é um *toolkit* Java código aberto que oferece uma gama de funcionalidades para facilitar o desenvolvimento de aplicações baseadas em ontologias (Holanda et al., 2013). Esse *toolkit* permite, por exemplo: manipulação de ontologias em um repositório, consultas na linguagem SPARQL e manipulação de instâncias por meio do paradigma de orientação a objetos. O JOINT fornece uma API para bancos de dados RDF parecida com o Hibernate para bancos de dados relacionais.

Inicialmente, o JOINT foi desenvolvido pelo Núcleo de Excelência em Tecnologias Sociais da Universidade Federal de Alagoas (NEES – UFAL) como um mecanismo de persistência para aplicações de pesquisas desenvolvidas na época em doutorados e mestrados. Em 2012, o JOINT se tornou oficialmente um projeto de pesquisa ao ser financiado pelo W3C Brasil/NIC.br/CGI.br. Atualmente, o JOINT é mantido principalmente pelo NEES, em cooperação com o Laboratório

de Computação Aplicada à Educação e Tecnologia Social Avançada da Universidade de São Paulo (CAEd – USP), desenvolvedores do MeuTutor e da Linked Knowledge, e com um número de voluntários que contribuem com ideias, descobertas de *bugs* e reparos.

5.4.1 O padrão KAO

O JOINT trabalha com o conceito do Knowledge Acess Object (KAO), o qual é um padrão de persistência similar ao Data Acess Object (DAO), com a diferença de que o KAO não trabalha somente com dados, mas com informação e conhecimento. Dessa forma, esse padrão pode separar a camada de negócios da aplicação da camada de acesso ao repositório semântico. Além disso, o KAO tem o objetivo de desacoplar os métodos de criação, remoção e recuperação de instâncias dos métodos de consulta em SPARQL. Portanto foi criada uma classe abstrata (`AbstractKAO`) que inclui a implementação dos métodos citados. Então para cada ontologia é criada uma classe concreta que herda a classe `AbstractKAO`, nessa classe concreta são implementados apenas métodos que envolvem consultas em SPARQL referentes à ontologia em questão, possibilitando a evolução do código da aplicação. A Figura 5.2 ilustra o padrão.

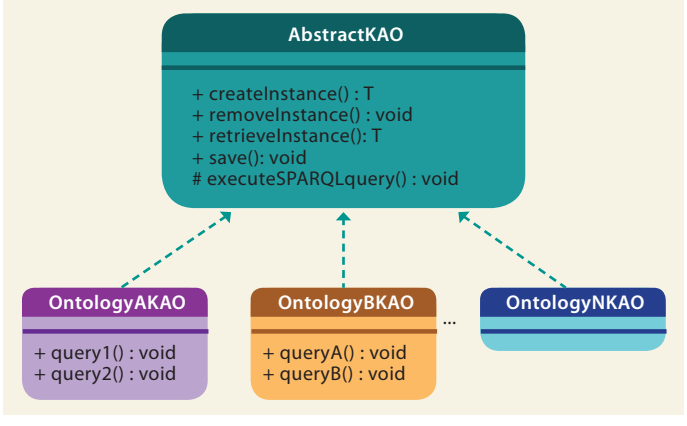

Figura 5.2 – Padrão de projeto KAO. Fonte: autores.

5.4.2 Obtendo e configurando o JOINT

Versões do JOINT podem ser baixadas a partir do SourceForge. Existem duas opções de *download.* A escolha deve ser feita dependendo do uso que será feito do JOINT:

- *Joint-jdk.tar.gz*: Esse é um arquivo *.tar* zipado para sistemas GNU contendo todo o código binário do *kit* de desenvolvimento do JOINT. Ele inclui todas as bibliotecas que o JOINT usa, assim como uma documentação.
- *Joint-jdk.zip*: Esse é um arquivo *.zip* que contém os mesmos arquivos do arquivo *tar.gz*.
- *Joint.jar*: Esse é um arquivo Java (*.jar*) encapsulando tudo do JOINT. O objetivo desse arquivo é facilitar a inclusão do JOINT no projeto Java da aplicação a ser desenvolvida. Só é necessário adicionar esse arquivo ao projeto para poder usar a API do JOINT.

Para configurar o ambiente de desenvolvimento para uso do JOINT, os usuários deverão executar apenas dois passos. O primeiro passo é a inclusão da biblioteca do JOINT, mencionada anteriormente, no *classpath* da aplicação Java. Com isso, todas as operações do JOINT podem ser acessadas pelo código da aplicação. O segundo passo é a configuração do repositório que será acessado pelo JOINT. Para isso, primeiro o desenvolvedor deverá criar um arquivo de propriedades na raiz do projeto com o nome *Repository.properties*. Depois, o desenvolvedor deverá criar uma classe concreta em seu projeto que implemente a interface RepositoryConfig do JOINT, que contém um método `createRepository`. Esse método deve retornar um objeto do tipo `Repository` do Sesame, permitindo a flexibilidade de configuração do repositório proveniente da API do Sesame. O local dessa classe criada deve ser especificado no arquivo *Repository.properties*.

5.4.3 Operações com ontologias no repositório

Para acessar a maioria das operações permitidas pelo JOINT, os desenvolvedores fazem uso da sua fachada: *RepositoryFacade*. Vale ressaltar que essas operações são feitas no repositório especificado na seção anterior. Para adicionar uma ontologia é utilizado o método `addOntology`, no qual o primeiro parâmetro é o caminho da ontologia e o segundo é o URI da ontologia. Para atualizar uma ontologia no repositório predefinido, chama-se o método `updateOntology` com os mesmos parâmetros do método anterior. Por fim, para remover uma ontologia do repositório, basta usar o método `deleteOntology`, passando apenas um parâmetro, que é o URI da ontologia, como apresentado no trecho de código a seguir.

```
import br.ufal.ic.joint.RepositoryFacade;

...

//Caminho local da ontologia e URI da ontologia
String ontologyPath = "C:/bbcOntology.owl";
String ontologyURI = "http://purl.org/ontology/po/";

//Chama a fachada do JOINT
RepositoryFacade fachada = new RepositoryFacade();
fachada.addOntology(ontologyPath, ontologyURI);//Adiciona

...

ontologyPath = "C:/bbcOntologyAtualizado.owl";
fachada.updateOntology(ontologyPath, ontologyURI);//Atualiza

...

fachada.deleteOntology(ontologyURI);//Deleta

...
```

5.4.4 Gerando código a partir de ontologias

Logo após a persistência das ontologias, os desenvolvedores usam a ferramenta para gerar código Java a partir delas, possibilitando,

assim, a manipulação de instâncias por meio das operações de criação, atualização, remoção e recuperação. O código desse gerador é uma extensão da ferramenta Alibaba.

A linguagem Java não suporta herança múltipla, ou seja, um objeto só pertence a uma determinada classe, não podendo ser uma instância de duas ou mais classes. Em contrapartida, indivíduos de uma ontologia podem ter múltiplos tipos associados a eles. Portanto a solução encontrada pela ferramenta Alibaba foi gerar interfaces Java, uma vez que estas permitem a herança múltipla. Assim, uma interface Java pode herdar duas ou mais interfaces. Para implementar tais interfaces, a ferramenta Alibaba gera, em tempo de execução, *proxies* dinâmicos para que os desenvolvedores só possam acessar e manipular as interfaces geradas.

O JOINT usa um gerador de código do Alibaba modificado e evoluído com o objetivo de não apenas gerar interfaces Java, porém classes concretas que implementam cada interface. Desenvolvedores ainda acessam e manipulam apenas as interfaces, porém, em vez de criar *proxies* dinâmicos em tempo de execução, JOINT retorna um objeto da classe que implementa a determinada interface. Isso resulta, teoricamente, em uma menor sobrecarga da ferramenta em termos de desempenho, pois, em vez de criar implementações em tempo de execução com *proxies*, a ferramenta usa códigos pré-compilados com as classes concretas geradas. Veja como gerar código Java automaticamente a partir de ontologias no trecho de código a seguir.

```
import br.ufal.ic.joint.RepositoryFacade;
import br.ufal.ic.joint.module.ontology.operations.OntologyCompiler;

...

//Caminho do arquivo jar que será gerado
String ontologyPath = "C:/bbc.jar";

//URL da ontologia que deseja ser compilada
String ontologyURL = "file://C:/bbcOntology.owl";
List<String> lista = new ArrayList<String>();
```

```
lista.add(ontologyURL);

//Chama a fachada do JOINT
RepositoryFacade fachada = new RepositoryFacade();
OntologyCompiler compiler = fachada.getOntologyCompiler(ontologyPath,
lista); // gerador
compiler.compile(); // Compila a ontologia em classes Java

...
```

Como evidencia o exemplo, a fachada da ferramenta deve ser inicializada, para que as operações da ferramenta possam ser utilizadas. O primeiro parâmetro do método `getOntologyCompiler` (retorna uma instância da classe que compila a ontologia em Java) é o caminho do arquivo *.jar* que se deseja gerar com as classes da ontologia. O segundo parâmetro é uma lista de URLs de cada ontologia. Se a ontologia estiver na Web, coloca-se o protocolo "`http://...`"; se for um arquivo local do computador, coloca-se "`file://...`". Em seguida, basta chamar o método `compile` para gerar o arquivo *.jar*.

5.4.5 Criando um KAO

Como mencionado anteriormente, para cada ontologia que se deseja manipular instâncias e executar consultas SPARQL, é necessária a criação de um KAO respectivo a essa ontologia. Nas próximas subseções vamos utilizar a ontologia *programmes* da BBC por motivos de exemplificação. Já que queremos manipular instâncias ou até mesmo consultar a ontologia, cria-se uma classe concreta que herda a classe `AbstractKAO` representando a ontologia BBC. O trecho de código a seguir apresenta o código da nova classe criada.

```
import br.ufal.ic.joint.module.kao.AbstractKAO;
public class ProgrammesBBCKAO extends AbstractKAO {
   public <T> ProgrammesBBCKAO(Class<T> classe, String ontologyURI){
      super(classe, ontologyURI);
   }
}
```

Ao herdar a classe `AbstractKAO`, será necessário criar o construtor. Esse construtor deve ser genérico, recebendo como parâmetro uma classe pertencente à ontologia, que deve ser gerada pelo JOINT, e o URI da ontologia em questão. Essas variáveis devem ser passadas para o `AbstractKAO` com o comando `super`.

5.4.6 Manipulando instâncias com o CRUD

Para executar as operações de CRUD e fazer consultas em uma ontologia com o JOINT é necessária a criação de uma classe KAO representando a ontologia que deseja ser manipulada. No caso do exemplo da BBC, foi criada a classe `ProgrammesBBCKAO`, que herda a classe `AbstractKAO`.

O trecho de código a seguir mostra o código de criação de uma instância do tipo `Episodes` utilizando a `ProgrammesBBCKAO`. Primeiro cria-se um objeto da classe `ProgrammesBBCKAO`, passando por parâmetro a interface `Episodes`. Em seguida, também cria-se um novo objeto com o método `create`, passando como parâmetro o URI da ontologia BBC e o nome da instância. O sistema, então, gera a instância no repositório e retorna para o desenvolvedor um objeto com a implementação da interface passada (no caso do exemplo `Episodes`). A partir desse ponto, o desenvolvedor pode utilizar o objeto como ele desejar, ressaltando que as alterações feitas no objeto não irão refletir no repositório até que o objeto seja atualizado.

```
// ...
String bbc = "http://purl.org/ontology/po";
String nomeEpisodio = "Episodio1";

// Cria o kao passando a interface que se deseja operar
String ontologyURL = "file://C:/bbcOntology.owl";
ProgrammesBBCKAO kao = new ProgrammesBBCKAO(Episodes.class);

// Cria o objeto passando a uri da ontologia e o nome da instância
Episodes episodio1 = kao.create(bbc, nomeEpisodio);
```

```
// ...
// Usa o objeto da forma que desejar
// ...
```

O código para recuperar uma instância contida no repositório é bastante similar ao código de criação, mudando apenas o método chamado no KAO, como mostra o trecho de código a seguir. Vale ressaltar que a instância deve estar criada no repositório para poder ser recuperada.

```
// ...
String bbc = "http://purl.org/ontology/po";
String nomeEpisodio = "Episodio1";

// Cria o kao passando a interface que se deseja operar
String ontologyURL = "file://C:/bbcOntology.owl";
ProgrammesBBCKAO kao = new ProgrammesBBCKAO(Episodes.class);

// Recupera o objeto passando a uri da ontologia e o nome da instância
Episodes episodio1 = kao.retrieveInstance(bbc, nomeEpisodio);

// ...
// Usa o objeto da forma que desejar
// ...
```

A próxima operação a ser detalhada é a de atualização de instâncias. O trecho de código a seguir indica, na perspectiva do desenvolvedor, como atualizar uma instância. Embora nessa figura o objeto seja recuperado e atualizado pelo mesmo KAO, isso não é necessário. Na verdade, a instância pode ser recuperada em outra parte da aplicação do desenvolvedor, depois enviada por diversas camadas da arquitetura e alterada em alguma delas. No momento em que a aplicação devolve a instância para o KAO atualizar, ele recupera as mudanças feitas no objeto e sincroniza com os dados no repositório.

```
String bbc = "http://purl.org/ontology/po";
String nomeEpisodio = "Episodio1";

// Cria o kao passando a interface que se deseja operar
String ontologyURL = "file://C:/bbcOntology.owl";
```

```
ProgrammesBBCKAO kao = new ProgrammesBBCKAO(Episodes.class);

// Recupera o objeto passando a uri da ontologia e o nome da instância
Episodes episodio1 = kao.retrieveInstance(bbc, nomeEpisodio);
episodio1.setName("Novo Nome do Episodio");

// Atualiza para salvar as mudanças
kao.update(episodio1);
//...
```

A última operação de CRUD é a de remoção de instâncias. Assim como nos métodos de criar e recuperar instância, o desenvolvedor cria um KAO passando a interface que ele deseja operar e, ao chamar o método `delete`, passa como parâmetros o URI da ontologia e o nome da instância. O trecho de código a seguir mostra o código de remoção de instâncias na perspectiva do desenvolvedor. O método `delete` também pode ser chamado; desse modo, passando como parâmetro o próprio objeto a ser removido.

```
// ...
String bbc = "http://purl.org/ontology/po";
String nomeEpisodio = "Episodio1";

// Cria o kao passando a interface que se deseja operar
String ontologyURL = "file://C:/bbcOntology.owl";
ProgrammesBBCKAO kao = new ProgrammesBBCKAO(Episodes.class);

// Remove o objeto passando a uri da ontologia e o nome da instância
kao.delete(bbc, nomeEpisodio);

//...
```

5.4.7 Executando Consultas

Ao usar consultas SPARQL em um repositório, o desenvolvedor, na maioria das vezes, consulta determinadas instâncias, ou seja, a consulta em SPARQL retorna uma ou um conjunto de instâncias. O JOINT provê que os retornos dessas consultas sejam também na

forma de objetos. Para executar consultas SPARQL no JOINT, o usuário deverá criar métodos dentro do KAO que façam essas consultas conforme mostra o trecho de código a seguir.

```
import br.ufal.ic.joint.module.kao.AbstractKAO;
public class ProgrammesBBCKAO extends AbstractKAO {
   public <T> ProgrammesBBCKAO(Class<T> classe, String ontologyURI){
      super(classe, ontologyURI);
   }
   public List<Object> getAllEpisodes(){
      String query = "PREFIX prog:<http://purl.org/ontology/po#>"
      +" PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
      + "SELECT?episodes WHERE {?episodes rdf.typr prog.Episode}";
      List<Object> lista = this.executeQueryAsList(query);
      return lista;
   }
}
```

Apesar de essa consulta retornar todas as instâncias da classe `Episodes`, esse mesmo resultado pode ser obtido usando o método `retrieveAllInstances`. Além disso, o desenvolvedor pode também criar um método de consulta genérico para receber uma *String* com a consulta e então executá-la, retornando o resultado (veja o trecho de código a seguir). Outros métodos de consultas SPARQL estão descritos na documentação Java da ferramenta[2].

```
import br.ufal.ic.joint.module.kao.AbstractKAO;
public class ProgrammesBBCKAO extends AbstractKAO {
   public <T> ProgrammesBBCKAO(Class<T> classe, String ontologyURI){
      super(classe, ontologyURI);
   }
   public boolean executeQuery(string query){
      return this.executeBooleanQuery(query);
   }
}
```

2 Disponível em: <*http://jointnees.sourceforge.net/documentacao.html*>.

5.5 Considerações Finais

O principal objetivo deste capítulo foi oferecer ao leitor uma visão geral sobre desenvolvimento de aplicações utilizando tecnologias semânticas. Foi dado um destaque especial na Plataforma JOINT para o desenvolvimento de aplicações semânticas em Java. Esperamos que as seguintes mensagens tenham sido passadas:

- Compreensão sobre diferentes tecnologias para desenvolvimento de aplicações semânticas.
- Conhecimento sobre o desenvolvimento de aplicações por meio da manipulação de triplas RDF e objetos.
- Entendimento sobre o desenvolvimento de uma aplicação utilizando a plataforma JOINT.

CAPÍTULO 6

Conclusão

Neste livro, abordamos o conceito de Dados Abertos e mostramos como a grande geração de dados no contexto da Web tem demandado que os sistemas sejam cada vez mais capazes de processar estes dados de maneira automatizada e com valor agregado aos usuários finais. Isto levou à proposição tanto do conceito de Web Semântica quanto do conceito de Dados Conectados.

Com isso, foram abordados os princípios de Dados Conectados e Dados Abertos Conectados. Destacamos a estruturação de dados por meio do uso de padrões recomendados pelo W3C como o RDF e RDF-S, apresentando suas características e as diferentes formas de serialização. Além disso, abordamos como fazer o enriquecimento de Dados Conectados por meio da utilização de ontologias, em especial da especificação recomendada pelo W3C, a OWL 2. Finalmente, apresentamos algumas das ferramentas existentes tanto para a publicação quanto para o consumo de dados abertos.

É importante frisar que este livro não teve a intenção de apresentar todas as especificações e ferramentas que proporcionam a publicação e o consumo de Dados Abertos Conectados. Por exemplo, não foram abordadas as especificações do SPARQL (para recuperação de dados), DCAT (para catalogação de dados), PROV-O (para proveniência de dados), entre outras. Além disso, não abordamos as publicações recentes do SLTI, como o Plano de Dados Abertos.

Referências

5 START Open Data. 2012. Disponível em: <*ttp://5stardata.info/*>.

AASMAN, J. *Allegro Graph:* RDF Triple Database. Cidade: Oakland Franz Incorporated. 2006.

ALATRISH, E. S. Comparision of Ontology Editors. e-RAF Journal on Computing, v. 4, 2012.

BAADER, F.; BRANDT, S.; LUTZ, C. Pushing the EL Envelope. In: proc. of the 19th joint int. Conf. On artificial intelligence - ijcai. Cidade: Edinburgh. 2005.

BARTLETT, O. Linked Data: Connecting together the BBC's Online Content. *Internet Blog.* [Online] BBC, 29 Feb 2013. Disponível em: <*http://www.bbc.co.uk/blogs/internet/posts/Linked-Data-Connecting-together-the--BBCs-Online-Content*>.

BAUER, C.; KING, G. *Java Persistence With Hibernate*. Greenwich: Manning Publications Company, 2007.

BAADER, F. et al. Pushing the EL Envelope. In Proceedings of the 19th international joint conference on Artificial intelligence (IJCAI), p. 364-369, 2005.

BARROS, H. et al. Steps, techniques, and technologies for the development of intelligent applications based on Semantic Web Services: A case study in e-learning systems. Engineering Applications of Artificial Intelligence. v. 24, p. 1355-1367.

BAUER, C.; KING, G. Java Persistence with Hibernate. New York: Manning Publications, 2006.

BECHHOFER, S.; HARPER, S.; Lunn, D. SADIe: Semantic Annotation for Accessibility. In: *Proceedings of the International Conference on the Semantic Web*, Springer, p. 101-115, 2006.

BERNERS-LEE, T. W3 future directions. *Talk*. [Plenary at WWW Geneva 94]. Geneve, Switzerland: W3C, Sep. 1994. Disponível em: <*http://www.w3.org/Talks/WWW94Tim/*>.

BERNERS-LEE, T. The Semantic Web Layer Cake. *XML-2000 Conference*. Cidade: Washington, 2000.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*., v. 1, p. 34-43, 2001.

BERNERS-LEE, T. Design Issues. [S.l.]: W3C, 2006. Disponível em: <*http://www.w3.org/DesignIssues/*>.

BERNERS-LEE, T. Open, Linked Data for a Global Community. *Gov 2.0 Expo. Cidade: Washington*. 2010.

BIZER, C. The DBpedia Data Provision Architecture. *DBPedia*. [Online], 9 nov. 2009. Disponível em: <*http://wiki.dbpedia.org/Architecture?v=14cg*>.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked Data – The story so far. [ed.] Tim Heath, M. Hepp and Christian Bizer. *International Journal on Semantic Web and Information System*, Special Issue on Linked Data, 2006.

Bizer, C.; Cyganiak, R. *RDF 1.1 TriG:* RDF Dataset Language. W3C, 2014. Disponível em: <*http://www.w3.org/TR/trig/*>.

BITTENCOURT, I. I. *Modelos e Ferramentas para a Construção de Sistemas Educacionais Adaptativos e Semânticos*. Universidade Federal de Campina Grande. Tese de Doutorado. 2009.

BITTENCOURT, I. I. et al. Research Directions on Semantic Web and Education. Scientia, v. 19, p. 1-9, 2008.

BITTENCOURT, I. I. et al. Towards a new generation of web-based educational systems: The convergence between artificial and human agents. IEEE Multidisciplinary Engineering Education Magazine, v. 3, 1, 17-24, 2008

BITTENCOURT, I. I. et al. A computational model for developing semantic web-based educational systems, Knowledge-Based Systems, v. 22, n. 4, p. 302-315, 2009.

BLOOMBERG, M. R.; MERCHANT, R. N. *Open Data Policy and Technical Standards Manual*. New York: New York City Departament of Information Technology and Telecommunications, 2012.

BOCK, C. et al. *OWL 2 Web Ontology Language:* Structural Specification and Functional-Style Syntax. 2nd edition. W3C, 2012. Disponível em: <*http://www.w3.org/TR/owl2-syntax/*>.

BRICKLEY, D.; GUHA, R. V. *Resource Description Framework (RDF) Schema Specification 1.0.* W3C, 2000. Disponível em: <*http://www.w3.org/TR/2000/CR-rdf-schema-20000327/*>.

BRICKLEY, D.; GUHA, R. V. *RDF Schema 1.1.* W3C, 2014. Disponível em: <*http://www.w3.org/TR/2014/REC-rdf-schema-20140225/*>.

BROEKSTRA, J.; KAMPMAN, A.; HARMELEN, F. V. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, Proceedings of the First International Semantic Web Conference, 54-68, 2002.

CAFEPRESS. *The 5-Star Linked Data Mug. CafePress.* 2011. Disponível em: <*http://www.cafepress.com/*>.

CALERO, C.; RUIZ, F.; PIATTINI, M. *Ontologies for Software Engineering and Software Technology.* Berlin: Springer, 2006.

CALVANESE, D. et al. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *Journal of Automated Reasoning.* v. 39, p. 385-429, 3 oct. 2007.

CARDOSO, J.; HEPP, M.; LYTRAS, M. D. *The semantic web: real-world applications from industry.* Berlin: Springer, 2007. v. 1.

CAROTHERS, G.; SEABORNE, A. *RDF 1.1 N-Triples:* A line-based syntax for an RDF graph. W3C, 2014. Disponível em: <*http://www.w3.org/TR/n-triples/*>.

CAROTHERS, G. *RDF 1.1 N-Quads:* A line-based syntax for RDF datasets. W3C, 2014. Disponível em: <*http://www.w3.org/TR/n-quads/*>.

CASELLAS, N. *Methodologies, Tools and Languages for Ontology Design. Legal Ontology Engineering:* Methodologies, Modelling Trends, and the Ontology of Professional Judicial Knowledge. Law, Governance and Technology. Berlin: Springer, 2011.

CHEN, Y. *Commercial Semantic Web of Digital Library*. 2007. Disponível em: <*http://escience.anu.edu.au/project/07S1/YinChen/YinChenInitialPresentation.ppt*>.

CHIGNARD, S. A Brief History of Open Data. ParisTech Review. March 29th, 2013.

CYGANIAK, R.; WOOD, D.; LANTHALER, M. *RDF 1.1 Concepts and Abstract Syntax*. W3C, 2014. Disponível em: <*http://www.w3.org/TR/rdf11-concepts/*>.

COBDEN, M. et al. A Research Agenda for Linked Closed Dataset. Proceedings of the Second International Workshop on Consuming Linked Data (COLD). Bonn, Germanyu, 2011.

DATA.GOV. Linked Data Datasets. *Openning Up Government*. Disponível em: <*http://data.gov.uk/data/search?openness_score=5*>.

D'AQUIN, M.; NOY, N. Where to publish and find ontologies? A survey of ontology libraries. Web Semantics: Science, Services and Agents on the World Wide Web, v. 11, p. 96-111, 2012.

DAVENPORT, T. H.; PATIL, D. J. Data Scientist: The Sexiest Job of the 21st Century. *Harvard Business Review*, v. 1, p. 70-77, 2012.

DEVEDZIC, V. *Semantic Web and Education*. Berlin: Springer, 2006. v. 1.

DONINI, F. M. *The description logic handbook*: Theory, implementation and applications. Cambridge: Cambridge University Press, 2003.

DERMEVAL, D. et al. Ontology-based feature modeling: An empirical study in changing scenarios. Expert Systems with Applications, v. 42, n. 11, 2015.

DERMEVAL, D. et al. Applications of ontologies in requirements engineering: a systematic review of the literature. Requirements Engineering, p. 1-33, 2015b.

DUBOST, K.; HERMAN, I. State of the Semantic Web. Talk at the INTAP Semantic Web Conference, 2008. Disponível em: <*http://www.w3.org/2008/Talks/0307-Tokyo-IH/Slides.pdf*>.

EMC. *The Digital Universe in 2020*: Big Data, Bigger Digital Shadows, Biggest Growth in the Far East. 2012. Disponível em: <*http://www.emc.com/leadership/digital-universe/index.htm*>.

ERLING, O.; MIKHAILOV, I. Virtuoso: RDF Support in a Native RDBMS. Semantic Web Information Management, p. 501-519, 2009.

ESPOSITO, D. *Building Web Solutions with ASP.NET and ADO.NET*. Redmond: Microsoft Press, 2002.

FLORIDI, L. *Information:* A Very Short Introduction. New York: Oxford University Press, 2010.

GANDON, F.; HERMAN, I. *An introduction to Semantic Web and Linked Data*: or how to link data and schemas on the Web. Rio de Janeiro: W3C, 2013. v. 1.

GANDON, F.; SCHREIBER, G. *RDF 1.1 XML Syntax*. Rio de Janeiro: W3C, 2014. Disponível em: <*http://www.w3.org/TR/rdf-syntax-grammar/*>.

GANGEMI, A. et al. *Core Software Ontology, Core Ontology of Software Components, Core Ontology of Services*. Disponível em: <*http://km.aifb.kit.edu/sites/cos/*>.

GENESERETH, M. R.; NILSSON, N. J. *Logical Foundations of Artificial Intelligence*. San Mateo: Morgan Kaufmann Publishers, 1987.

GROSOF, B. N. et al. Description Logic Programs: Combining Logic Programs with Description Logic. In: *Proceedings of the 12th international World Wide Web Conference (WWW)*, p. 48–57, 2003.

GROVE, M. Empire: RDF & SPARQL Meet Java & JPA. In: Semantic Technology Conference. [Online] 2010. Disponível em: <*http://semtech2010.semanticuniverse.com/sessionPop.cfm?confid=42&proposalid=3022*>.

GRUBER, T. R. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*. 5, v. 2, p. 199-220, 1993.

GRUNINGER, M.; FOX, M. S. Methodology for the Design and Evaluation of Ontologies. In: Proceedings of the Workshop On Basic Ontological Issues In Knowledge Sharing, IJCAI. Local: Montreal, 1995.

GUARINO, N.; CARRARA, M.; GIARETTA, P. Formalizing Ontological Commitments. Proceedings of Association for the Advancement of Artificial Intelligence Conference (AAAI), p. 560-567, 1994.

GUARINO, N.; GIARETTA, P. *Ontologies and Knowledge Bases*: Towards a Terminological Clarification. Towards Very Large Knowledge Bases. Amsterdam: IOS Press, 1995.

GUARINO, N. Understanding, Building, and Using Ontologies. International Journal of Human and Computer Studies, v. 46, n. 2-3, p. 293-310, 1997.

GUARINO, N. Formal Ontology in Information Systems. Proceedings of Formal Ontology in Information Systems, p. 3-15, 1998.

GUARINO, N. *Formal Ontology and Knowledge Representation*. Trento: Laboratorio de Ontologia Applicata (LOA), 2010.

GUIZZARDI, G. On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. Proceedings of the conference on Databases and Information Systems, p. 18-39, 2007.

HERMAN, I. et al. *RDFa 1.1 Primer* – 3rd edition. W3C, 2015. Disponível em: <*http://www.w3.org/TR/2015/NOTE-rdfa-primer-20150317/*>.

HITZLER, P. et al. *OWL 2 Web Ontology Language (Primer)*. [S.l.]: W3C, 11 Dec. 2012.

HORROCKS, I. Ontologies and the Semantic Web. Communication of the ACM. 2008, v. 51, 12, p. 58-67.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; van HARMELEN, F. From SHIQ and RDF to OWL: the Making of a Web Ontology Language. Journal of Web Semantics, v. 1, n. 1, p. 7-26, 2003.

HOLANDA, O. et al. JOINT: Java ontology integrated toolkit. Expert Systems with Applications, v. 40, p. 6469-6477, 2013.

HYLAND, B.; ATEMEZING, G.; VILLAZÓN-TERRAZAS, B. Best Practices for Publishing Linked Data. *W3C Consortium.* [Online] 9 Jan. 2014. Disponível em: <*http://www.w3.org/TR/2014/NOTE-ld-bp-20140109/*>.

IANNELLA, R. An Idiot's Guide to the Resource Description Framework. *The New Review of Information Networking*, v. 4, p.181-188, 1998.

ISOTANI. S. et al. Estado da Arte em Web Semântica e Web 2.0: Potencialidades e Tendências da Nova Geração de Ambientes de Ensino na Internet. *Revista Brasileira de Informática na Educação*, v. 17, p. 30-42, 2009.

ISOTANI, S. An Ontological Engineering Approach to Computer-Supported Collaborative Learning: From Theory to Pratice. PhD. Thesis in Information Engineering. Osaka University. 2009. DOI: 10.13140/RG.2.1.4055.8249.

ISOTANI, S. et al. Engenharia de Software Baseada em Ontologias: Uma Revisão dos Desafios e Oportunidades. Revista IEEE América Latina, v. 13, n. 3, 1-7.

JACOBS, I.; WALSH, N. *Architecture of the World Wide Web.* W3C, 2004. Disponível em: <*http://www.w3.org/TR/webarch/*>.

KIRYAKOV, A.; OGNYANOV, D.; MANOV, D. OWLIM—A Pragmatic Semantic Repository for OWL. Lecture Notes in Computer Science, Vol. 3807, p. 182-192, 2005.

KOIVUNEN, M-R.; MILLER, E. *W3C Semantic Web Activity.* Finland: Semantic Web Kick-off Seminar, 2001.

KMI. 2014. Apollo Ontology Editor. [Online] 2014. Disponível em: <*http://apollo.open.ac.uk/*>.

KRIVOV, S.; WILLIAMS, R.; VILLA, F. GrOWL: A tool for visualization and editing of OWL ontologies. Web Semantics: Science, Services and Agents on the World Wide Web, v. 5, n. 2, p. 54-57, 2007.

LEHMANN, J. et al. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal.* 1, v. 5, p. 1-29, 2014.

LOD PROJECT. *Linking Open (LOD) Data Project Cloud Diagram. Linked Data – Connect Distributed Data across the Web.* Sep. 2011. Disponível em: <*http://lod-cloud.net/versions/2011-09-19/lod-cloud_1000px.png*>.

LUTZ, C; SATTLER, U.; TENDERA, L. The complexity of finite model reasoning in description logics. *Inf. Comput. Academic Press*, v. 199, p. 132-171, 2005.

McBRIDE, B. Jena: A semantic web toolkit. *IEEE Internet Computing.*, v. 6, p. 55-59, 2002.

McGUINNESS, D. L. Question Answering on the Semantic Web. IEEE Intelligent Systems, v. 19, n. 1, p. 82-85, 2004.

MEDEIROS, I. Linked Data at globo.com. Web of Linked Entities (WoLE 2013). Proceedings of the World Wide Web Conference (WWW), Rio de Janeiro, Brazil. 2013

MIKA, P. 2007. *Social networks and the Semantic Web.* Berlin: Springer, 2007.

MIZOGUCHI, R. Tutorial on ontological engineering: part 3: Advanced course of ontological engineering. New Generation Computing, v. 22, n. 2, p. 198-220, 2004.

MIZOGUCHI-LAB. 2014. Hozo Ontology Editor. [Online] 2014. Disponível em: <*http://www.hozo.jp/*>.

MOTIK, B. et al. *OWL 2 Web Ontology Language:* Profiles – 2nd edition. W3C, 2014. Disponível em: <*http://www.w3.org/TR/owl2-profiles/*>.

NOWACK, B. *The Semantic Web Technology Stack (not a piece of cake...). Linked Data Developer.* [Online] 2009. Disponível em: <*http://linkeddatadeveloper.com/Projects/Linked-Data/media/fig11.2.png*>.

NOY, N. F.; McGUINNESS, D. L. *Ontology Development 101:* A Guide to Creating Your First Ontology. Knowledge Systems Laboratory and Stanford Medical Informatics. Stanford: Stanford University, 2001.

OBAMA, B. *Transparency and Open Government.* Washington, United States of America:The White House, 2008.

OPEN DEFINITION. *The Open Definition.* [Online] 2015. Disponível em: <*http://opendefinition.org/*>.

OPEN DATA COMMONS. *Open Data Commons Licenses.* [Online] 2014. Disponível em: <*http://opendatacommons.org/licenses/*>.

OPEN KNOWLEDGE FOUNDATION. *Open Data Handbook.* 2010. Disponível em: <*http://opendatahandbook.org/guide/en/*>.

OPEN SEMANTIC FRAMEWORK. *Metamodeling in Domain Ontologies.* 2014. Disponível em: <*http://www.mkbergman.com/913/metamodeling--in-domain-ontologies/*>.

OWLIM–a pragmatic semantic repository for OWL. Berlin: Springer, 2005.

PATEL-SCHNEIDER, P. F. Building the Semantic Web Tower from RDF Straw. Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), p. 546-551, 2005.

PETERSON, D. et al. *W3C XML Schema Definition Language (XSD) 1.1 Part 2*: Datatypes. W3C, 2012. Disponível em: <*http://www.w3.org/TR/xmlschema11-2/*>.

PRUD'HOMMEAUX, E.; CAROTHERS, G. *RDF 1.1 Turtle:* Terse RDF Triple Language. W3C, 2014. Disponível em: <*http://www.w3.org/TR/turtle/*>.

RAIMOND, Y. et al. *Programmes ontology. Programmes Developers.* [Online] BBC. 7 Sep. 2009. Disponível em: <*http://www.bbc.co.uk/ontologies/programmes/2009-09-07.shtml*>.

SEMAFORA. 2014. OntoStudio. [Online] 2014. <*http://www.semafora--systems.com/en/products/ontostudio/*>.

SCHNEIDER, M. *OWL 2 Web Ontology Language: RDF-Based Semantics* – 2nd edition. W3C, 2012. Disponível em: <*http://www.w3.org/TR/owl2--rdf-based-semantics/*>.

SCHREIBER, G.; RAIMOND, Y. *RDF 1.1 Primer.* W3C, 2014. Disponível em: <*http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/*>.

SHADBOLT, N.; HALL, W.; BERNERS-LEE, T. The Semantic Web Revisited. *IEEE Intelligent Systems Journal.* 3, v. 21, p. 96-101, 2006.

SOWA, J. A *Dynamic Theory of Ontology*. *Formal Ontology in Information Systems.* Local:Baltimore. Bennett and C. Fellbaum,2006.

SOWA, J. *Knowledge Representation:* Logical, Philosophical, and Computational Foundations. Boston: PWS Publishing, 1998.

SPORNY, M.; KELLOGG, G.; LANTHALER, M *JSON-LD 1.0: A JSON-based Serialization for Linked Data.* Rio de Janeiro: W3C, 2014.

STANFORD. 2014. Protégé Ontology Editor. [Online] 2014. Disponível em: <*http://protege.stanford.edu/*>.

SWARTOUT, W.; TATE, A. Ontologies. IEEE Intelligent Systems, v. 1, n. 14, p. 18-19, 1999.

SZEKELY, B.; BETZ, J. Jastor: Typesafe, ontology driven rdf access from java. *Jastor.* [Online] 2006. [Cited: Janeiro 20, 2014.] Disponível em: <*http://jastor.sourceforge.net/*>.

TENNISON, J. Good Practices for Capability URLs. W3C, 2014. Disponível em: <*http://www.w3.org/TR/capability-urls/*>.

TOPQUADRANT. 2014. TopBraid Composer Standard Edition. [Online] 2014. Disponível em: <*http://www.topquadrant.com/tools/modeling--topbraid-composer-standard-edition/*>.

USCHOLD, M.; GRUNINGER, M. Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review.* 11, v. 2, p. 1-63, 1996.

VILLAZÓN-TERRAZAS, B. et al. *Methodological Guidelines for Publishing Government Linked Data. Linking Government Data.* Local: New York David Wood, 2011.

W3C BRASIL. *Publicação de Dados em Formato Aberto. Escola de Políticas Públicas,* 2013. Disponível em: <*http://cursos.ep.org.br/course/view.php?id=25&section=1*>.

W3C OWL WORKING GROUP. *OWL 2 Web Ontology Language:* Document Overview. 2nd edition. W3C, 2012. Disponível em: <*http://www.w3.org/TR/owl2-overview/*>.

WOOD, D. *What's New in RDF 1.1.* W3C, 2014. Disponível em: <*http://www.w3.org/TR/rdf11-new/*>.

WOOD, D. et al. *Linked Data:* Structured Data on the Web. New York: Manning Publications, 2014.