**UNIVERSIDADE FEDERAL DO CEARÁ**

**PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO**

**PROGRAMA DE MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO**

**MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO**

**TIAGO DA SILVA VINUTO**

# UMA ABORDAGEM PARA A GERAÇÃO SEMIAUTOMÁTICA DE MAPEAMENTOS R2R BASEADO EM UM CATÁLOGO DE PADRÕES

**FORTALEZA**

**2017**

TIAGO DA SILVA VINUTO

# UMA ABORDAGEM PARA A GERAÇÃO SEMIAUTOMÁTICA DE MAPEAMENTOS R2R BASEADO EM UM CATÁLOGO DE PADRÕES

Dissertação de Mestrado submetida ao Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação.

Área de Concentração: Banco de Dados

Orientadora: Profa. Dra. Vânia Maria Ponte Vidal

Co-Orientadora: Profa. Dra. Valéria Magalhães Pequeno

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

V785a Vinuto, Tiago.
Uma Abordagem para a Geração Semiautomática de Mapeamentos R2R Baseado em um Catálogo de Padrões / Tiago Vinuto. – 2017.
79 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2017.
Orientação: Prof. Dr. Vânia Maria Ponte Vidal.
Coorientação: Prof. Dr. Valéria Magalhães Pequeno.

1. Padrões de Mapeamento, Mapeamento entre ontologias, Mapeamento R2R, Assertivas de Mapeamento, modelo RDF. I. Título.

CDD 005

---

TIAGO DA SILVA VINUTO

# UMA ABORDAGEM PARA A GERAÇÃO SEMIAUTOMÁTICA DE MAPEAMENTOS R2R BASEADO EM UM CATÁLOGO DE PADRÕES

Dissertação de Mestrado submetida ao Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação.

Área de Concentração: Banco de Dados

Aprovada em:

BANCA EXAMINADORA

---

Profa. Dra. Vânia Maria Ponte Vidal (Orientadora)
Universidade Federal do Ceará (UFC)

---

Profa. Dra. Valéria Magalhães Pequeno (Co-Orientadora)
Instituto de Engenharia de Sistemas e Computadores (INESC-ID)

---

Prof. Dr. José Antônio Fernandes de Macedo
Universidade Federal do Ceará (UFC)

---

Prof. Dr. José Maria da Silva Monteiro Filho
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

## AGRADECIMENTOS

O meu mestrado não se desenvolveu da maneira por mim esperada. Alguns acontecimentos tornaram o processo longo e deveras conturbado. Porém, mantive-me firme, sempre acreditando que tudo seria concluído da melhor forma. Entretanto, não posso deixar de agradecer as pessoas que foram fundamentais na participação deste mestrado. Sendo assim, agradeço:

A Deus, pelo amor incondicional, e ao meu mestre Jesus Cristo, por estar sempre ao meu lado, renovando as minhas forças e me ensinando a perseverar na caminhada. Aos meus pais Francisco Paulo Vinuto de Freitas e minha mãe, Maria Lucineide da Silva Vinuto, que estão sempre ao meu lado, oferecendo seu amor, atenção e suporte nos momentos necessários. Sem eles, eu não seria nada do que sou hoje. Aos meus irmãos, Jarles Vinuto e Daiane Vinuto, meus grandes amigos, pelo companheirismo e por compartilharem comigo tantos momentos importantes. A todos os demais familiares que sempre torcem por mim.

A minha querida noiva Lidiene Eugênia Braga Rodrigues que acompanhou e testemunhou toda esta caminhada de perto. Meu agradecimento pelo carinho, pelo precioso apoio, pelas palavras de incentivo que mudaram as situações mais difíceis e pelo seu amor para comigo.

As Professoras Vânia Maria Ponte Vidal e Valéria Magalhães Pequeno pela orientação e co-orientação, respectivamente. Agradeço por todo o aprendizado. Pela orientação presente, enorme paciência, dedicação, persistência, sugestões, incentivos, confiança e amizade. Com as senhoras, eu aprendi muitas lições, tanto acadêmicas quanto sobre a postura de alguém que está sempre disposta a crescer, a lutar, a debater, a ouvir e a ensinar. Sou muito grato pela oportunidade de concluir esse desafio.

Aos meus queridos amigos (grupo Arida) Ticiana Linhares, Samara Martins, Lívia Almada, Narciso Arruda, Regis Pires, Salomão Santos, Ricardo Ávila, Gabriel Lopes, Amanda Venceslau, Gustavo Coutinho e Wellington Franco que estiveram presentes em tantos momentos durante estes anos. Foi muito bom conviver com vocês! Agradeço particularmente ao Gabriel Lopes, que foi um companheiro de trabalho em algumas questões relacionadas à implementação e escrita e ao Narciso Arruda, pelas excelentes ideias. Aos meus amigos Andressa Bezerra, Carleandro Noleto e Ernando Gomes que, apesar da distância, uma hora ou outra se fizeram presentes com a sua amizade e troca de experiências pessoais e profissionais. Aos meus amigos Mardson Ferreira e Bruno Leal pela amizade, apoio e momentos de descontração.

Ao professor José Antônio Macedo pela oportunidade de participar do projeto Petrobras Ontologia. Aos meus amigos David Araújo, Tércio Jorge, Lucas Peres, Caio Viktor

e Victor Vieira pelo convívio e experiências repassadas, em especial ao David Araújo pela paciência e seus ensinamentos.

Aos professores José Antônio Macedo e José Maria Monteiro por terem aceitado o convite de participar da banca examinadora deste trabalho.

A todos os professores e funcionários do MDCC que me ajudaram, de alguma forma, durante este caminho.

À Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (Funcap) pelo auxílio financeiro que possibilitou a realização deste trabalho. Meus agradecimentos especiais à Universidade Federal do Ceará (UFC) por todo o suporte oferecido para apoiar o desenvolvimento desta dissertação.

“É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota".

(Theodore Roosevelt)

## RESUMO

A Web de dados ligados tem crescido consideravelmente nos últimos anos e abrange uma vasta gama de domínios diferentes hoje (BIZER *et al.*, 2011). Fontes de dados ligados usam diferentes vocabulários para representar dados sobre um tipo específico de objeto. Como por exemplo, *DBpedia*[1] e *Music ontology*[2] que usam seus vocabulários proprietários para representar dados sobre artistas musicais. Traduzir dados dessas fontes de dados para o vocabulário que é esperado por uma aplicação requer um grande número de mapeamentos e pode exigir muitas transformações estruturais, bem como transformações complexas no valor da propriedade. Diversas tecnologias despontam no sentido de traduzir ou mapear ontologias como, a linguagem SPARQL 1.1, a ferramenta *Mosto* e o *framework R2R*. Dentre estas escolhemos utilizar em nosso estudo a linguagem R2R, apontada em (BIZER *et al.*, 2012) como uma boa opção para mapear ontologias, pois se destaca em termos de expressividade e desempenho. A linguagem de mapeamento R2R é uma linguagem baseada, na linguagem SPARQL, que permite transformar dados de um vocabulário de origem em um vocabulário de destino definido pelo usuário. Contudo, a construção de mapeamentos utilizando essa linguagem é complexa e sujeita a diversos tipos de erros, tais como erros de escrita ou até mesmo erros semânticos, exigindo do usuário experiência para definir os mapeamentos. Diante deste cenário, propomos uma estratégia, usando padrões de mapeamento, para gerar automaticamente mapeamentos R2R a partir de AM. Nossa abordagem é dividida em duas etapas: (1) a especificação manual de um conjunto de AM entre o vocabulário de uma ontologia fonte e do vocabulário de uma ontologia alvo de escolha do usuário; e (2) a geração automática dos mapeamentos R2R com base no resultado do primeiro passo. Por último nós apresentamos a ferramenta **R2R By Assertions** para ajudar o usuário no processo de geração de mapeamentos R2R.



**Palavras-chave:** Padrões de Mapeamento. Mapeamento entre ontologias. Mapeamento R2R. Assertivas de Mapeamento. Modelo RDF. Ontologias.


[1] http://dbpedia.org/resource/
[2] http://musicontology.com/

## ABSTRACT

The web of linked data has grown considerably in recent years and covers a wide range of different domains today (BIZER *et al.*, 2011). Linked data sources use different vocabularies to represent data about a specific type of object. For example, *DBpedia* [3] and *Music ontology* [4] use their proprietary vocabularies to represent data About musical artists. Translating data from these bound data sources into the vocabulary that is expected by a linked data application requires a large number of mappings and may require many structural transformations as well as complex transformations in the property value. Several tools emerge to map ontologies such as the *SPARQL 1.1* language, LDIF framework and the *Mosto* tool. We choose to use in our study the R2R language, which was pointed out in (BIZER *et al.*, 2012) as a good option to map ontologies, as it stands out in terms of expressiveness and performance. The R2R mapping language is a language based on the SPARQL language that allows you to transform data from a source vocabulary into a user-defined target vocabulary. However, defining mappings using this language is complex and subject to several types of errors, such as writing errors or even semantic errors, requiring expect user most to define the mappings. In this scenario, we propose an approach, using mapping patterns to automatically generate R2R mappings from a AMs. The approach is divided into two steps: (1) the manual specification of a set of AM between the vocabulary of a source ontology and the vocabulary of a target ontology of the user's choice; and (2) the automatic generation of the R2R mappings based on the result of the first step. Finally, we present the R2R By Assertions tool to help the user in the process of generating R2R mapping.



**Keywords:** Mapping Patterns. Ontology mapping. R2R mapping. Mapping Assertion. RDF Model. Ontologies.


[3] http://dbpedia.org/resource/
[4] http://musicontology.com/

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AC | Assertiva de Correspodência |
| AM | Assertivas de Mapeamento |
| AMC | Assertiva de Mapeamento de Classe |
| AMD | Assertiva de Mapeamento de Tipo de Dados |
| AMO | Assertiva de Mapeamento de Objeto |
| AMs | Assertivas de Mapeamento |

**SUMÁRIO**

# 1 INTRODUÇÃO

A *Web* é atualmente um enorme espaço global de documentos e dados distribuídos em múltiplas fontes autônomas e heterogêneas, inteligíveis apenas por humanos, que se expande a cada dia, dificultando a recuperação de informações de maneira integrada. Neste contexto, surgiu uma nova abordagem, denominada Web de Dados (BIZER *et al.*, 2009). A Web de Dados, diferentemente da Web convencional, propõe a publicação de dados, na Web, ao invés de documentos. Dessa forma, *softwares* são capazes de interpretar as informações na Web e utilizá-las para os mais devidos fins (p.e. tomada de decisão e pesquisas). Além disso, a Web de Dados propõe que devam existir *links* entre as fontes de dados, possibilitando a descoberta de novas informações em tempo de execução.

O sucesso da Web de Dados se dá principalmente devido à adoção de tecnologias e padrões. Dentre as tecnologias, destacam-se os **Vocabulários** e as **Ontologias**. Os vocabulários definem os termos utilizados para descrever e representar uma área do conhecimento. Desta forma, os vocabulários funcionam como ferramentas conceituais que nos ajudam a descrever os dados e seus formatos. As ontologias fornecem uma base semântica, facilitando a comunicação entre sistemas e agentes (JACOB, 2003). Dentre os padrões, destacam-se os padrões de **infraestrutura** e da **Web Semântica**. Como padrões de infraestrutura têm-se as **URI**s que são usadas como mecanismo de nome global e o **HTTP** que é o núcleo do protocolo da *Web*. Como padrões da *Web* Semântica podem ser citados: o **RDF** (*Resource Description Framework*), um modelo de dados simples, expressivo, extensível e que permite interligar itens entre diferentes fontes de dados e a linguagem **SPARQL** (PRUD'HOMMEAUX *et al.*, 2008), uma linguagem de consulta de alto nível capaz de abstrair detalhes da sequência de passos necessários para a execução de consultas sobre fontes heterogêneas. Graças a essas tecnologias e padrões, uma quantidade sem precedentes de fontes de dados ligados tem sido produzida e continua a crescer rapidamente.

Uma boa prática na Web de Dados é a de reutilizar vocabulários existentes em ontologias disponíveis na *Web*. Embora grande parte das fontes de dados ligados sigam essa boa prática, pode ser necessária a criação de ontologias que definam vocabulários em um domínio mais específico. Isto pode levar a publicação de um mesmo conteúdo em diferentes locais, porém usando terminologias distintas. Por exemplo, o nome de uma pessoa na ontologia *FOAF*[1] é modelado por meio da propriedade *foaf:name*, enquanto na ontologia *Dublin Core*[2] é modelado

[1] http://xmlns.com/foaf/0.1/
[2] http://purl.org/dc/elements/1.1/

usando *dc:creator*.

A heterogeneidade resultante desse processo é um obstáculo na construção de aplicações sobre dados ligados. Dentre as causas dessa heterogeneidade, podem ser destacadas as divergências sintáticas, semânticas e estruturais entre as fontes. Essa problemática inviabiliza a promessa da Web de Dados de construir aplicações sobre um espaço de dados homogêneo que permita descobrir e integrar novas fontes de dados (BERNERS-LEE *et al.*, 2001). Uma forma para conciliar as heterogeneidades entre os diversos vocabulários existentes na Web de Dados é por meio da criação de mapeamentos entre eles.

## 1.1 Motivação e Caracterização do Problema

Em diversas áreas, como Web Semântica e Integração de Dados, ontologias têm sido adotadas para descrever formalmente a semântica das fontes de dados, com o intuito de facilitar a descoberta e a recuperação de informações (LOPES, 2010). Entretanto, a utilização de várias ontologias torna-se um problema quando é necessário consultar dados armazenados em ontologias distintas e que dizem respeito a um mesmo domínio. Para facilitar esse processo é comum o uso de mapeamentos entre ontologias distintas.

Mapeamentos são utilizados para especificar o relacionamento entre termos (classes e propriedades) de ontologias diferentes. Desta forma, termos representados por vocabulários distintos são mapeados por diferentes fontes de dados, as quais contêm conteúdos similares. Mapeamentos entre ontologias são frequentemente especificados de forma declarativa e independente de dados (YU; POPA, 2004). Além disso, eles são utilizados para encontrar e interligar entidades similares presentes em conjuntos de dados distintos.

A tarefa de traduzir dados de uma fontes de dados (*ontologias fonte*) para o vocabulário que é esperado por uma aplicação (*ontologia alvo*) requer um grande número de mapeamentos. Esses mapeamentos entre ontologias distintas pode exigir do usuário um conhecimento específico sobre as diversas ontologias que se deseja mapear. Além disso, o usuário deve ter alguma experiência em definir os mapeamentos (possivelmente complexos).

Diversas tecnologias despontam no sentido de traduzir ou mapear dados entre ontologias. Por exemplo, a linguagem *SPARQL 1.1* que traduz vocabulários por meio da cláusula *Construct*, a ferramenta *Mosto* (RIVERO *et al.*, 2011) que se baseia na linguagem SPARQL para mapear vocabulários e o *framework LDIF* (SCHULTZ *et al.*, 2012) que propõe a linguagem de mapeamento R2R (JUN *et al.*, 2012) para especificar mapeamentos entre esquemas RDF. Dentre

estas tecnologias, escolheu-se usar em nosso estudo a linguagem R2R apontada em (BIZER *et al.*, 2012) como uma boa opção para mapear ontologias, pois se destaca em termos de expressividade e desempenho. A linguagem de mapeamento R2R é uma linguagem, baseada na linguagem SPARQL, que permite transformar dados de um vocabulário de origem em um vocabulário de destino definido pelo usuário.

## 1.2 Contribuições

Como principal contribuição deste trabalho, está a proposta de uma abordagem semiautomática baseada em padrões de mapeamento para gerar mapeamentos R2R.

Os padrões de mapeamentos representam uma solução genérica para um determinado problema de mapeamento. Surgem como uma forma de facilitar a especificação do mapeamento entre ontologias, e são essenciais para representar diversos problemas de incompatibilidades entre ontologias distintas.

Os problemas de mapeamento em cada padrão são especificados por meio das AM (AM) (Capítulo 4), as quais permitem definir os mapeamentos simples ou mais complexos.

A terceira contribuição do trabalho é a criação de um catálago de padrões contendo os tipos mais comuns de problemas de mapeamento. A biblioteca é organizada usando uma taxonomia que distribuí os padrões de acordo com o tipo problema de mapeamento abordado

A quarta contribuição do trabalho é a criação da ferramenta *R2R By Assertions*. Uma Ferramenta gráfica desenvolvida para auxiliar o usuário na geração dos mapeamentos R2R por meio das Assertivas de Mapeamento.

## 1.3 Organização da Dissertação

Esta dissertação possui sete capítulos. Não sendo mais necessário tratar deste capítulo introdutório, os demais são delineados a seguir.

O Capítulo 2 – Fundamentação Teórica – apresenta uma síntese dos assuntos mais relevantes que servem de fundamentação para o entendimento dos demais capítulos desta dissertação. Ele expõe os principais tópicos relevantes para o desenvolvimento desta dissertação tais como: Web Semântica, Ontologia, RDF, RDFS, OWL (*Web Ontology Language*) e SPARQL.

O Capítulo 3 - Representação do Mapeamento - apresenta o formalismo usado para definir mapeamentos e o exemplo de motivação que será utilizado ao longo do trabalho.

O Capítulo 4 - Padrões de Mapeamento - mostra os padrões de mapeamento e como eles podem facilitar o mapeamento entre ontologias distintas. O capítulo 4, apresenta ainda Um *template* que permite representar os mapeamentos de uma maneira uniforme, uma sintaxe mais concisa, denominada *AM* e por último uma biblioteca contendo os padrões de mapeamento mais comuns.

O Capítulo 5 – RBA: R2R By Assertions – apresenta nossa ferramenta gráfica para geração dos mapeamentos R2R. Usamos o estudo de caso introduzido no Capítulo 3 para validação da ferramenta.

O Capítulo 6 – Trabalhos Relacionados – apresenta os trabalhos relacionados ao contexto de mapeamento entre ontologias. Neste capítulo é apresentada uma síntese das principais trabalhos referente ao alinhamento e mapeamento de ontologias.

Por fim, o Capítulo 7 – Conclusão – tece as considerações finais sobre o trabalho e apresenta possíveis trabalhos futuros para dar prosseguimento ao que obtivemos até o momento.

# 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados conceitos importantes para compreensão desta dissertação. A Seção 2.1 faz uma breve introdução sobre a Web Semântica. A Seção 2.2 apresenta algumas tecnologias da Web Semântica, especificamente discorremos sobre conceitos, como *ontologias*, *RDF*, *RDF-S*, *OWL* e *SPARQL*. Na Seção 2.3 é introduzida a linguagem de mapeamento R2R. Finalmente, a Seção 2.4 mostra as considerações finais deste capítulo.

## 2.1 Web Semântica

A Web, como a conhecemos hoje, está em contínuo processo de evolução e vem revolucionando a forma como criamos conteúdo e trocamos informações. Conforme observado em (CUNHA, 2002), a Web surgiu com o propósito de ter informações com significados bem definidos, de forma a facilitar a cooperação e a comunicação entre pessoas e agentes computacionais. Entretanto, os dados disponibilizados na Web tradicional utiliza uma estrutura e uma organização que mais facilita a comunicação entre humanos do que entre agentes computacionais.

Breitman e Casanova em (BREITMAN; CASANOVA, 2005) definem que a Web tradicional é sintática e tem apenas propósito representativo. Dessa forma, não é permitido extrair a semântica dos dados ou das informações sem antes aplicar algum esforço. Contrapondo essa Web Sintática, surge a Web Semântica a qual busca mecanismos que capturem o significado dos dados contidos nas páginas e criem um ambiente no qual os computadores possam processar e relacionar conteúdos provenientes de várias fontes.

De acordo com (BREITMAN; CASANOVA, 2005), é preciso processar o conteúdo semântico envolvido nos dados, considerando um determinado contexto, para, assim, possibilitar o entendimento e a interpretação pelos agentes computacionais. Para que isso se torne possível, é necessário embutir semântica na estrutura dos documentos disponíveis na Web (BREITMAN; CASANOVA, 2005).

Entre os principais objetivos da Web Semântica, está o aprimoramento dos serviços de busca e recuperação de dados na Web. Esses serviços visam principalmente melhorar a satisfação do usuário no momento da busca, retornando informações adequadas às necessidades do utilizador, pois a informação vem com um significado bem definido e não se encontra mais solta no emaranhado de conteúdo.

Outro objetivo da Web Semântica é potencializar a Web por meio da criação de

padrões e ferramentas que possibilitem atribuir significado ao conteúdo das páginas Web, bem como tornar possível o trabalho cooperativo entre usuários e programas.

Surge então, um conjunto de padrões que torna possível a identificação de dados na Web os quais possibilita a representação sintática, estrutural, semântica e lógica de informações referentes aos dados contidos na WEB. Esse conjunto de padrões formam as diversas camadas em que se divide a arquitetura da Web Semântica (apresentada na Figura 1).

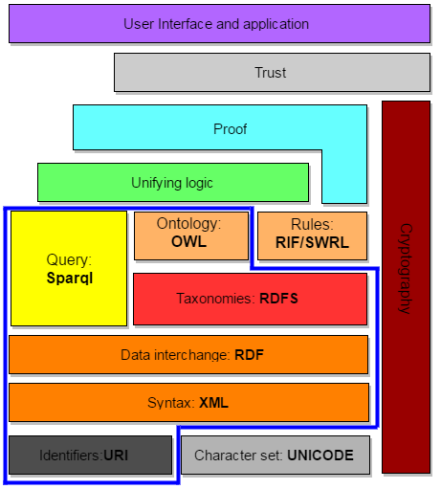



Figura 1: *A arquitetura da Web Semântica dividida em várias camadas (BERNERS-LEE, 2005).*

A Figura 1 apresenta a arquitetura da Web Semântica com os diferentes conceitos que a compõem organizados na forma de uma pilha (BAADER *et al.*, 2005). As tecnologias destacadas em cada camada dão suporte à estrutura da Web Semântica e têm o intuito tanto de defini-la, quanto de permitir a partilha global do conhecimento.

A camada base é constituída por dois conceitos básicos: *Unique Resource Identifiers* (URIs) e o *padrão UNICODE de codificação de caracteres* para garantir compatibilidade com todas as línguas mundialmente conhecidas. É importante dizer que existe uma abreviação mais recente, denominada IRI (*Internationalized Unicode-encoded*), que também é usada com o mesmo propósito.

A camada "XML" faz parte da arquitetura, pois RDF foi originalmente serializado em XML. Isso se deve porque XML se tornou um padrão amplamente difundido para troca de dados na Web. Acima da camada XML, está a camada RDF. RDF é um modelo simples de dados com uma semântica formal. É utilizado para padronizar a definição e a utilização de metadados, com intuito de facilitar o entendimento dos relacionamentos e a utilidade das informações dos dados. Esses metadados descrevem recursos da Web que nada mais são que quaisquer coisas tanto concretas quanto abstratas. Por exemplo, pessoas, instituições e até mesmo uma página

Web podem ser considerados recursos.

Acima da camada "RDF", estão as camadas RDF-S e OWL, as quais são linguagens que aumentam a expressividade do RDF, permitindo assim expressar dados na Web com mais semântica. RDF-S possui um maior poder de representação do que as linguagens XML ou RDF, permitindo organizar recursos de forma hierárquica; permite, ainda, definir classes, propriedades, relacionamentos e restrições. OWL é uma linguagem para definir e instanciar ontologias. As ontologias podem ser vistas como uma forma mais flexível de descrever um domínio sem nenhum compromisso com sua implementação (FENSEL, 2001).

SPARQL é uma linguagem de consulta que é utilizada para consultar dados no formato RDF. As demais camadas, (*unifying logic*, *Proof*, etc.) não são importantes para o entendimento do presente trabalho e, por isso, não são detalhadas aqui. De modo geral, elas são usadas para se descobrir novos dados a partir dos dados explicitamente definidos. Uma explicação mais completa sobre a arquitetura da Web Semântica pode ser encontrada em (BAADER *et al.*, 2005).

## 2.2 Tecnologias da Web Semântica

Esta seção apresenta as tecnologias da Web Semântica, que são mais relevantes para o presente trabalho: *ontologias*, *RDF*, *RDF-S*, *OWL* e *SPARQL*.

### *2.2.1 Ontologias*

Ontologia é o elemento da Web Semântica que possibilita descrever e representar diferentes domínios, de maneira utilizável pelo computador. Ontologias apresentam-se como modelo de relacionamento de entidades em um domínio específico do conhecimento. Seu principal objetivo é suprir a necessidade de um vocabulário compartilhado pelo qual as informações possam ser trocadas e também reutilizadas pelos usuários, sejam eles humanos ou agentes computacionais (DACONTA *et al.*, 2003).

Ontologias podem ainda ser descritas como uma técnica de organização de informações que vem recebendo especial atenção nos últimos anos, principalmente no que diz respeito à representação formal de conhecimento (GUARINO, 1995) (GUARINO, 1998). Geralmente criadas por especialistas, tendo sua estrutura baseada na descrição de conceitos e dos relacionamentos semânticos entre eles, as *"Ontologias geram uma especificação explícita de uma*

***conceitualização*** *e uma* ***descrição formal dos conceitos*** *e relacionamentos compartilhados de uma área de conhecimento"* (GRUBER, 1995). **Conceitualização** pode ser entendida como uma visão abstrata e simplificada do mundo que se deseja representar. **Descrição formal dos conceitos** indica que estes podem ser passíveis de um processamento automático (PINHEIRO, 2011).

Uma das razões para a popularidade das ontologias existe devido à grande promessa de compartilhamento e entendimento comum de algum domínio de conhecimento entre pessoas e computadores. Nesse sentido, ontologias têm sido desenvolvidas para facilitar o compartilhamento e a reutilização de informações (GRUBER, 1993). Ontologias possuem uma abordagem declarativa, a qual permite a modelagem de sistemas em alto nível de abstração e mais próximo do conhecimento do domínio a ser modelado.

Embora existam diferentes definições do termo Ontologia, algumas noções básicas quanto à sua estrutura são compartilhadas pela maioria das abordagens. A seguir são apresentados os principais elementos utilizados para definir uma ontologia (PINHEIRO, 2011)(NOY *et al.*, 2001):

- **Classe** (ou simplesmente *conceito*): Descreve grupos abstratos, conjuntos ou coleções de objetos. Por exemplo, uma classe *Artista* representa todos os *artistas* do meio musical. Uma classe pode ter subclasses, que representam conceitos mais específicos do que a superclasse (NOY *et al.*, 2001). Por exemplo, podemos dividir a classe de todos os *artistas* em *artista solo* e *banda*.
- **Propriedade** (também conhecida como *predicado*): São relações binárias, que podem ser usadas para estabelecer relacionamentos entre indivíduos ou entre indivíduos e valores de dados. Esses relacionamentos permitem afirmar fatos gerais sobre os membros das classes e podem também especificar fatos sobre indivíduos (WELTY *et al.*, 2004). Por exemplo, os álbuns e o nome de um artista. Ademais, uma relação pode ter sub-relações, as quais herdam as propriedades de outras relações.
- **Instância** (também chamada de *indivíduo*): É uma representação concreta de uma classe. Por exemplo, *Nando Reis* é uma instância da classe *Pessoa*.

Na prática, uma ontologia define um conjunto de termos que serão utilizados para formular consultas (ALMEIDA; BAX, 2003). Ontologias são fundamentais por ser uma forma viável de estruturar informações, regras de combinação entre os termos e seus relacionamentos. O uso de ontologias trazem vários benefícios (NOY *et al.*, 2001), sendo o de maior ênfase a

possibilidade de comunicação entre pessoas, agentes e sistemas, já que ontologias permitem, por exemplo, o reuso mapeamento de formalismos, o compartilhamento de conhecimentos, dentre outras finalidades.

### *2.2.2 Resource Description Framework (RDF)*

RDF (SCHREIBER; RAIMOND, 2015) é uma linguagem utilizada para representar informações na Web. O modelo RDF proporciona uma forma de descrição de recursos. RDF tem como base a ideia de que recursos são descritos por meio de declarações e que suas propriedades têm valores. Em RDF, as declarações podem ser modeladas como "triplas RDF". Uma tripla é dividida em três partes: (*Sujeito, Predicado, Objeto*), como exibido na Figura 2 (LIMA; CARVALHO, 2005). Uma outra maneira utilizada para moldar declarações em RDF é por meio de nós (sujeito e objeto) e arcos (predicado) em um grafo dirigido.

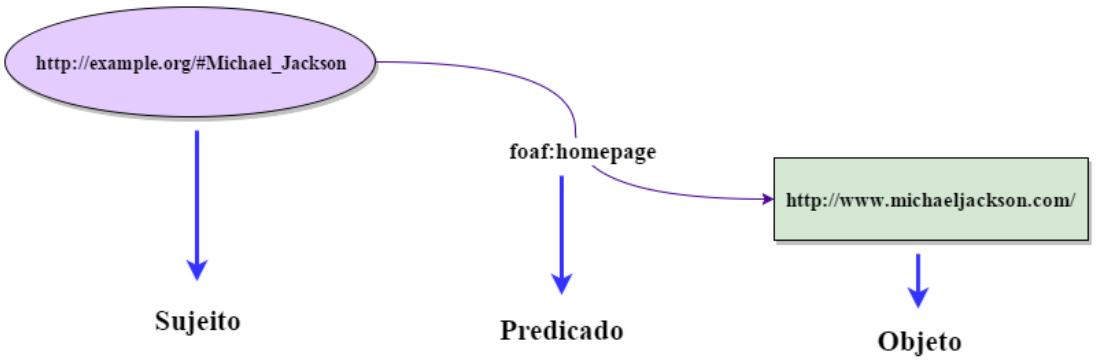



Figura 2: *Representação gráfica de uma tripla RDF*.

O modelo de dados básico de RDF consiste de três tipos de objetos:

- **Sujeito**: Identifica o objeto da declaração, por exemplo, uma página Web, a qual é vista como um recurso. Todo recurso deve ser capaz de ser identificado unicamente. Os recursos são sempre representados por um URI, os quais possibilitam a criação de identificadores para qualquer entidade imaginável;
- **Predicado** (Propriedade): Especifica um relacionamento entre o sujeito e um objeto em RDF. O predicado pode ser utilizado para descrever um recurso, e, geralmente, são nomeadas por um verbo. Propriedades permitem relacionar um recurso a dados ou a outros recursos;
- **Objeto** (Declaração): Denomina o recurso ou dado que se relaciona com o sujeito. É composto pela associação de um recurso específico, uma propriedade e o valor da propriedade para esse recurso. O valor de um objeto pode ser definido como um recurso ou um literal.

Segundo (KLYNE; CARROLL, 2004), RDF pode ser considerado um modelo de dados, recomendado pelo *World Wide Web Consortium*[1] (W3C), cuja a finalidade principal é embutir semântica aos dados na Web e fornecer interoperabilidade semântica entre aplicações que trocam informações processáveis por máquinas na Web. Dessa forma, RDF facilita a busca por recursos na web.

Existem várias representações sintáticas para o modelo RDF, e essas representações são conhecidas como serializações. Assim, dados RDF podem ser escritos utilizando diferentes formatos.

Alguns exemplos de serializações RDF incluem RDF/XML, N-Triples e Turtle. RDF/XML é uma sintaxe baseada em XML que representa as declarações de forma que possam ser mais facilmente processadas por máquinas. Já o formato *N-Triples* possui uma sintaxe muito básica, na qual existe apenas uma única linha por tripla no documento. Assim sendo as tuplas neste formato são rápidas para serem analisadas e altamente compreensíveis. *Turtle*, por sua vez, possui uma notação mais compacta do que *RDF/XML* e mais legível do que *N-Triples*. *Turtle* descreve grafos RDF em uma forma textual concentrada. As Figuras 3, 4 e 5 apresentam, respectivamente, exemplos de tuplas sobre o músico Nando reis, com sua data de nascimento e sua nacionalidade; as quais estão serializadas, respectivamente, usando RDF/XML, N-Triples e Turtle.

```
rdf/xml format:⟨?xml version="1.0" encoding="UTF-8"?⟩
⟨rdf:RDF
   xmlns:dbo="http://dbpedia.org/ontology/"
   xmlns:ex="http://example.org/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"⟩

⟨ rdf:Description rdf:about="http://example.org/Nando Reis"⟩
  ⟨rdf:type rdf:resource ="http://dbpedia.org/ontology/MusicalArtist"/⟩
  ⟨ dbo:birthDate⟩1963-01-12⟨/ dbo:birthDate⟩
  ⟨ dbo:nationality⟩ex:Brazil⟨/ dbo:nationality⟩
⟨/ rdf:Description⟩
⟨/ rdf:RDF⟩
```

Figura 3: Exemplo de um documento no formato *RDF/XML*.

```
<http://example.org/Nando Reis> <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
<http://dbpedia.org/ontology/MusicalArtist> .
<http://example.org/Nando Reis> <http://dbpedia.org/ontology/birthDate> "1963-01-12" .
<http://example.org/Nando Reis> <http://dbpedia.org/ontology/nationality> "ex:Brazil" .
```

Figura 4: Exemplo de um documento no formato *N-Triples*.

[1] https://www.w3.org/

```
@prefix dbo: <http://dbpedia.org/ontology/>.
@prefix ex: <http://example.org/> .

ex:Nando Reis a dbo:MusicalArtist ;
dbo:birthDate "1963-01-12" ;
dbo:nationality "ex:Brazil".
```

Figura 5: Exemplo de um documento no formato *Turtle*.

Apesar de fornecer uma semântica simplificada, com boa representação e que permite um tratamento eficaz dos metadados, RDF não faz suposições sobre a semântica dos recursos. Não existe nenhum construtor em RDF que especifique, por exemplo, que um recurso seja uma propriedade, ou seja, a idade de um artista, o nome de uma gravadora, etc.. Para suprir as necessidades existentes no modelo RDF e viabilizar o uso de um vocabulário maior e com restrições semânticas mais complexas, foi proposta uma extensão para RDF, chamada *Resource Description Framework Schema* (RDF-S).

### *2.2.3 RDF-S*

RDF-S (BRICKLEY; GUHA, 2004) é um vocabulário que estende RDF introduzindo uma camada que especifica algumas características que associam semântica a dados definidos em RDF. RDF-S tem construtores que possibilitam, por exemplo, especificar que determinadas URIs pertencem a uma determinada classe, por exemplo ("<http://example.org/#morte_e_vida_severina> rdf:type dbo:Album") indica que o recurso pertence à classe *Album*.

RDF-S usa a noção de classe (rdfs:Class) para especificar categorias que podem ser usadas para classificar os recursos. Assim, pode-se criar hierarquias de classes e subclasses (*rdfs:subClassOf*), ou ainda hierarquias de propriedades (rdfs:Property) e subpropriedades (rdfs:subPropertyOf). Em adição, RDF-S permite especificar restrições de tipos sobre os sujeitos e/ou os objetos de uma tripla por meio da especificação de domínios (*rdfs:domain*) e contradomínios (*rdfs:range*). A relação entre uma instância e sua classe é indicada por meio da propriedade *rdfs:type*. Também é importante destacar que a ideia de domínios e contradomínios relacionados à uma propriedade é diferente da ideia adotada, por exemplo, pelo modelo orientado a objetos (OO).

Para deixar mais clara a semântica que o RDF-S introduz, considere o exemplo apresentado na Figura 6.

Na Figura 6, *dbo:MusicalArtist*, *dbo:Artist* e *dbo:Country* são classes definidas na

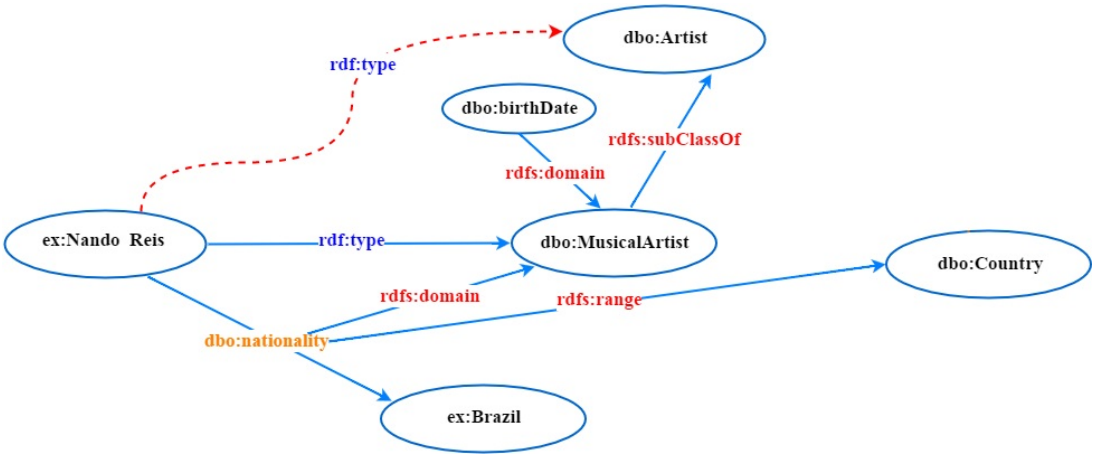



Figura 6: *Representação gráfica de um grafo utilizando conceitos RDF-S.*

*Dbpedia*, em que *dbo:MusicalArtist* é uma subclasse de *dbo:Artist*. *dbo:nationality* é uma propriedade cujo domínio é *dbo:MusicalArtist* e seu contradomínio é *dbo:Country*. Isto significa que esta propriedade pode ser usada com *dbo:MusicalArtist* e aceita como valores recursos de *dbo:Country*.

### *2.2.4 OWL*

*Ontology Web Language* (OWL) (BECHHOFER, 2009) é uma linguagem que estende RDF e RDF-S com um vocabulário mais abrangente provendo meios para tratar relacionamentos e restrições mais complexas entre objetos. A ideia principal da OWL é comportar a reprodução competente de ontologias, oferecendo ainda diversos construtores que permitem, entre outras coisas, a construção de classes complexas a partir de outras definições de classes, além do encadeamento de propriedades. Em OWL, existem três tipos de entidades como mostrado abaixo:

- *Instâncias* – representam recursos (também chamados de indivíduos).
- *Classes* – definem conjuntos de instâncias, de (indivíduos).
- *Propriedades* – representam relações binárias entre duas instâncias (*object property*) ou entre uma instância e um literal (*datatype property*).

Ontologias descritas usando OWL podem tornar o conhecimento implícito em conhecimento explícito, visto que OWL permite os sistemas criarem novas triplas com base em triplas já existentes. A linguagem possibilita também deduzir novos fatos com base em fatos já expostos. É importante destacar que uma ontologia OWL não define apenas um único modelo, mas sim um conjunto de restrições que definem um conjunto de modelos possíveis. Alguns exemplos de construtores OWL são:

- *owl:datatypeProperty*: uma propriedade cujo escopo é um conjunto de valores de tipo de dados;
- *owl:ObjectProperty*: uma propriedade cujo escopo é um conjunto de recursos identificados por URIs;
- *owl:disjointWith*: recursos pertencentes a uma classe não podem pertencer a outra.

A Figura 7 apresenta o fragmento de uma ontologia, enfatizando os diferentes termos utilizados para defini-la.

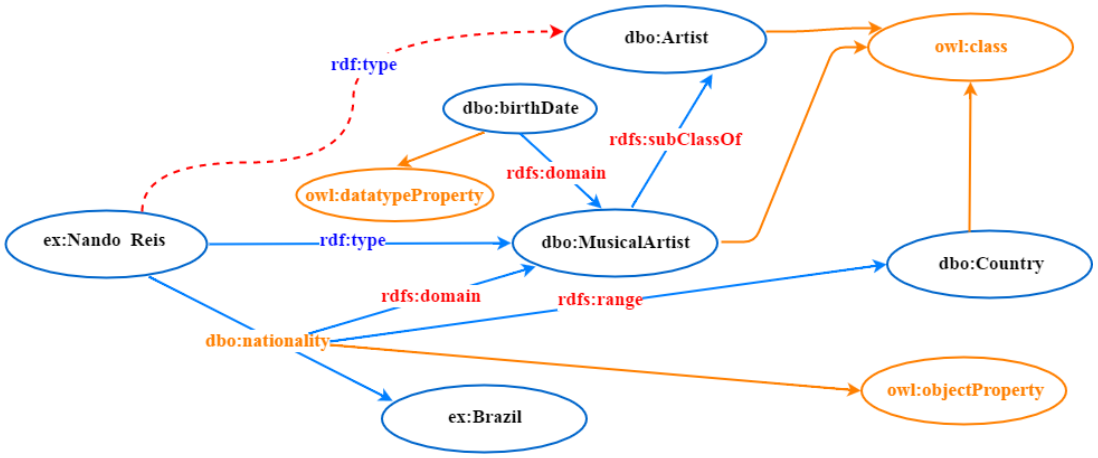



Figura 7: *Representação gráfica de um grafo utilizando conceitos OWL.*

Na Figura 7, as palavras e as flechas na cor laranja representam termos OWL. Observe que este grafo RDF é semanticamente mais rico do que o mostrado na Figura 6. Nós podemos, por exemplo, dizer explicitamente que *dbo:nationality* é uma propriedade de objeto, e *dbo:birthDate* é uma propriedade de tipo de dados.

A W3C sugere três sub-linguagens OWL: *OWL Lite*, *OWL DL* e *OWL Full*. Essas sub-linguagens possuem expressividade crescente e são projetadas para uso de comunidades específicas de usuários e tipos de aplicação para cada uma delas. A Figura 8 mostra uma comparação em termos de expressividade ou de "poder" entre as sub-linguagens. *OWL-Lite* é a menos expressiva das três, enquanto a *OWL-Full* é a mais expressiva; a expressividade da *OWL-DL* está entre a duas. Cada versão da OWL tornou-se um padrão para a descrição de ontologias (MENDONÇA, 2013).

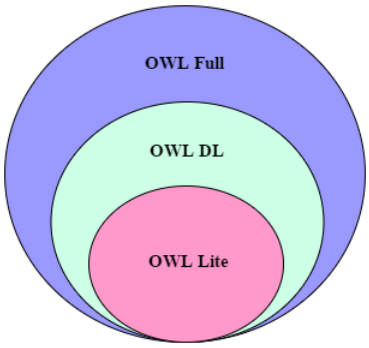



Figura 8: *Versões da linguagem OWL.*

### *2.2.5 Linguagem, Protocolo e Endpoint SPARQL*

Tanto bancos de dados relacionais quanto repositórios de triplas possuem uma linguagem para consultar os dados neles contidos. Bancos de dados relacionais usam a linguagem SQL, enquanto os repositórios de triplas (em inglês *triplestore*) usam a linguagem SPARQL.

De forma abreviada, *SPARQL Protocol and RDF Query Language* (SPARQL) é uma linguagem de consulta declarativa, bem como um protocolo da Web Semântica recomendado pelo (W3C).

Uma consulta SPARQL é construída com base em padrões de triplas (i.e, triplas RDF que possuem variáveis no lugar do sujeito, do predicado e/ou do objeto). Por exemplo, *<?s foaf:name ?n>* é um padrão de tripla com duas variáveis *?s* no lugar do sujeito e *?n* no lugar do objeto. Em SPARQL, nomes de variáveis são prefixados com “?” ou "$". Um conjunto padrão de triplas é chamado de padrão de grafo. SPARQL é baseado em consultas realizadas usando padrões de grafos. Uma consulta SPARQL consiste das seguintes seções:

- **Declarações de prefixos**, são utilizados para abreviar URIs. Exemplos de cláusulas SPARQL são ***prefix*** e ***base***.
- **Definição do dataset**, indica quais grafos RDF serão consultados. Exemplo de cláusulas SPARQL são: ***from*** e ***from named***.
- **Cláusula com resultado**, identifica a informação que deve ser retornada pela a consulta. Exemplo de cláusulas SPARQL são: ***select***, ***describe***, ***construct*** e ***ask***.
- **Padrões de grafos**, especificam os padrões de triplas para consultar no dataset indicado. Sintaxe: ***where {. . .}***.
- **Modificadores de consulta**, têm a função de limitar, ordenar e alterar os resultados de uma consulta. Exemplo de cláusulas SPARQL são : ***limit***, ***order by***, ***distinct***, ***offset*** e ***reduced***.

```
prefix foaf:⟨http://xmlns.com/foaf.0.1/⟩ .
prefix dbo:⟨http://dbpedia.org/ontology/⟩ .
select ?nome ?nascimento
from http://pt.dbpedia.org
where {?s foaf:name ?nome .
       ?s dbo:birthDate ?nascimento .}
limit 2
```

(a) Consulta SPARQL

| nome | nascimento |
|---|---|
| "Bruno Mars" | "1985-10-08" |
| "Nando Reis" | "1963-01-12" |

(b) Resultado da Consulta

Figura 9: Exemplo de uma Consulta SPARQL com um Possível Resultado.

A Figura 9a apresenta um exemplo de uma consulta SPARQL, a qual retorna os nomes e datas de nascimento de pessoas na ontologia *DBpedia* 9b. As principais cláusulas SPARQL utilizadas nesta consulta são:

- Prefixos e atalhos são bastante utilizados em consultas SPARQL para abreviar URIs. Por exemplo, se '*dc:*' é um prefixo para <http://purl.org/dc/elements/1.1/>, então '`dc:creator`' é uma notação abreviada da URI <http://purl.org/dc/elements/1.1/creator>. Um atalho frequentemente utilizado é a letra '`a`' ('artigo' em inglês), que serve para abreviar a URI '`rdf:type`', onde '`rdf:`' é um prefixo para <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. A sintaxe '<>' é utilizada para identificar o próprio URI do documento e onde ele está inserido. O símbolo ponto-e-vírgula ('`;`') é utilizado para delimitar uma tripla e definir que o sujeito da próxima tripla não precisa ser explicitamente definido, pois ele será o mesmo sujeito da tripla definida anteriormente (MAGALHÃES, 2012).
- **select**: contém os nomes das variáveis cujos valores serão retornados pela consulta. Na Figura 9a, a cláusula **select** especifica que os valores das variáveis *?nome* e *?nascimento* devem ser retornados.
- **from**: especifica o conjunto de dados RDF a ser consultado. Na Figura 9a, a cláusula **from** mostra que o conjunto de dados utilizado na consulta é a *DBpedia*.
- **where**: especifica os padrões de grafos e as condições que devem ser satisfeitas, para que uma tripla seja considerada no resultado da consulta. Por exemplo, a Figura 9a mostra três variáveis não ligadas: *?s*, *?nome* e *?nascimento*. Não há um URI específico no padrão grafo, em vez disso tem-se a variável *?s*. Assim, qualquer URI no sujeito do grafo pode ser combinado com *?s* e o resultado será mapeado para as variáveis *?nome* e *?nascimento*.
- **limit**: limita o número de linhas retornadas para 2.

## 2.3 Linguagem de Mapeamento R2R

Nesta seção, é apresentada a linguagem de mapeamento R2R. Destacamos suas principais propriedades, e em seguida, mostramos exemplos de mapeamentos entre ontologias utilizando essa linguagem de mapeamento. Esta seção é utilizada para motivar a aplicação da abordagem proposta neste trabalho.

R2R é uma linguagem declarativa, e baseada na linguagem SPARQL, para a publicação de mapeamentos entre diferentes vocabulários RDF. Cada mapeamento R2R refere-se a um mapeamento de classe (*r2r:classMapping*) ou a um mapeamento de propriedade (*r2r:propertyMapping*). A linguagem R2R é utilizada para recuperar dados de uma ontologia fonte e traduzi-los para um vocabulário de uma ontologia alvo.
Os principais elementos da linguagem de mapeamento R2R são:

- *Mapeamentos* - Definem as correspondências entre um termo no vocabulário alvo com termos de um vocabulário fonte.
- *Transformações* - Definem como os valores no padrão alvo são transformados. Por exemplo, usando *String* ou operações numéricas.
- *Modificadores* - Definem a estrutura alvo e o vocabulário que será mapeado.

### *2.3.1 Mapeamento R2R*

Mapeamentos R2R constituem unidades autônomas que representam correspondências entre termos de dois vocabulários diferentes. Cada mapeamento pode ser executado contra um conjunto de dados sem qualquer informação adicional. Todo mapeamento R2R é composto por cláusulas *r2r:sourcePattern* e *r2r:targetPattern*, similar a uma cláusula ***construct*** na linguagem SPARQL. A Tabela 1 lista as propriedades importantes de um mapeamento R2R:

### *2.3.2 Exemplos de Mapeamento R2R*

A seguir, destacamos alguns exemplos de mapeamentos R2R de classes e de propriedades.

A Figura 10 mostra o mapeamento R2R entre *mo:Label* e *myspo:recordLabel*. A variável de instância ?*SUBJ* é usada na cláusula *r2r:sourcePattern* para representar as instâncias que são o foco do mapeamento. As linhas 04 e 05 na Figura 10 especificam como obter todos os recursos da propriedade *myspo:recordLabel* da classe *myspo:MusicArtist*. A linha 05 especifica o formato da tripla em *r2r:targetPattern* (ou seja, a tripla que se refere a classe *mo:Label*). Quando

Tabela 1 – Lista das principais propriedades em um mapeamento R2R

| | |
|---|---|
| **r2r:prefixDenitions** | Pode ser utilizado para abreviar URIs dentro de um mapeamento. Os prefixos usados nos padrões fonte ou alvo devem ser definidos nesta cláusula. |
| **r2r:sourcePattern** | A cláusula *r2r:sourcePattern* compara com os dados da *Web* e ligando seus valores com um conjunto de variáveis. Nesse padrão, é possível incluir quase todas as expressões que existem na cláusula **where** SPARQL. |
| **r2r:targetPattern** | O *r2r:targetPattern* é utilizado para produzir triplas no vocabulário esperado, ou seja, o vocabulário alvo. |
| **r2r:transformation** | Define como os valores em *r2r:sourcePattern* são transformados em valores da ontologia alvo. |
| **r2r:mappingRef** | Usado em um mapeamento de propriedades para referir um mapeamento de classes (*r2r:ClassMapping*) definido anteriormente. Usado principalmente para reduzir a redundância. |

mapeamos *mo:Label* com *myspo:recordLabel*, um novo URI deve ser gerado. Esse novo URI tem como base o valor da propriedade *myspo:recordLabel*, a fim de garantir que um novo URI seja exclusivo. Assim, o sujeito da tripla *mo:Label* é o novo URI gerado na linha 07. O novo *u* URI é obtido usando as funções concat() e xpath-encode-for-uri().

```
01.  mp:  Label_to_recordLabel
02.  a r2r:ClassMapping;
03.    r2r:prefixDefinitions "mo:<...> .  myspo:<...>";
04.    r2r:sourcePattern "?SUBJ a myspo:MusicArtist;
05.    r2r:targetPattern "?u a mo:Label".
06.    r2r:transformation "?u= concat(?SUBJ, xpath-encode-for-uri(?r))".
```

Figura 10: Mapeamento R2R entre *mo:Label* e *myspo:recordLabel*.

A Figura 11 mostra um mapeamento R2R entre *moa:labelName* e *myspo:recordLabel*. Esse mapeamento especifica que cada tripla <?*SUBJ myspo:recordLabel* ?*r*> em *myspo:MusicArtist*, tal que o predicado *myspo:recordLabelType* = *"Major"* é verdadeiro, produz uma tripla <?*u moa:labelName* ?*r*> (linhas 04-07). O **r2r:mappingRef** faz referências para o *mp:Label_to_recordLabel* do mapeamento de classe. Esta cláusula é usada principalmente para reduzir a redundância. Nesse mapeamento, também precisamos gerar um URI que deve ser o mesmo gerado no mapeamento mostrado na Figura 10.

```
01.  mp:  labelName_to_recordLabel
02.  a r2r:PropertyMapping;
03.    r2r:prefixDefinitions "moa:  <...> .  myspo:<...>";
04.    r2r:mappingRef mp:Label_to_recordLabel ;
05.    r2r:sourcePattern "?SUBJ myspo:recordLabel ?r";
06.    r2r:targetPattern "?u moa:labelName ?r";
07.    r2r:transformation "?u = concat(?SUBJ, xpath-encode-for-uri(?r))".
```

Figura 11: Mapeamento R2R entre *moa:labelName* e *myspo:recordLabel*.

## 2.4 Discussão

Este capítulo apresentou uma síntese dos assuntos mais relevantes que servem como fundamentação para o entendimento dos demais capítulos desta dissertação. Apresentamos conceitos relacionados à Web Semântica, tais como ontologias, RDF, RDF-S e a linguagem SPARQL. Por último, introduzimos a linguagem R2R que será usada para mapear ontologias distintas.

# 3 REPRESENTAÇÃO DO MAPEAMENTO

Neste capítulo, é introduzido o formalismo para definir mapeamentos. Também é apresentado o exemplo de motivação que será utilizado ao longo deste trabalho. Por último, é apresentado as conclusões a respeito do formalismo proposto.

## 3.1 Conceitos Básicos e Notação Adotada

Nesta seção é apresentado o formalismo de mapeamento, baseado em regras e foi proposto inicialmente , para transformar o vocabulário das instâncias de uma ontologia fonte para o vocabulário de uma ontologia alvo. Tal formalismo é muito mais simples que as linguagens baseadas em regras, tais como SWRL[1], ou linguagens de mapeamento (p.e. R2R (BIZER; SCHULTZ, 2010a)), porém é suficiente para capturar expressivos mapeamentos. Além disso, o formalismo incorpora domínios *concretos* (Lutz, 2002) para capturar funções concretas, tais como "concatenação de *Strings*", necessárias em alguns tipos de mapeamentos, e predicados concretos, tais como "menor que", para especificar restrições. Alguns exemplos de regras de mapeamento serão apresentados brevemente neste capítulo.

Seja **V** um *vocabulário*, que é um conjunto de *classes* e *propriedades*. Uma *ontologia* é um par **O** = (**V**,Σ) tal que **V** é um vocabulário e Σ é um conjunto finito de fórmulas em **V**, ou seja, as *restrições* de **O**.

Seja $\mathbf{V_T}$ um *vocabulário alvo* e $\mathbf{O_S} = (\mathbf{V_S}, \Sigma_\mathbf{S})$ uma *ontologia fonte* com $\mathbf{V_S}$ e $\Sigma_\mathbf{S}$, sendo respectivamente, o vocabulário da fonte e o conjunto de restrições de $\mathbf{O_S}$. Seja $X$ um conjunto de variáveis. Seja $C$ um alfabeto de primeira ordem consistindo de um conjunto **F** de símbolos de função e um conjunto $P$ de símbolos de predicado, respectivamente chamados de *símbolos de função concreta* e *símbolos de predicados concretos*.

Os símbolos de função 0-ários são chamados de *constantes*, esses incluem os IRIs e valores de tipo de dados. É assumido que os símbolos em $C$ tem uma interpretação fixa. Por último, é suposto que $X$ e $C$ são mutuamente disjuntos e $C$ também é disjunto de $\mathbf{V_T}$ e $\mathbf{V_S}$.

Um *termo* é uma expressão recursivamente construída a partir de símbolos de função, constantes e variáveis, como de costume. Um *literal* é uma expressão de uma das formas:

- uma *classe literal* da forma $C(t)$, onde $C$ é uma classe de $\mathbf{V_T} \cup \mathbf{V_S}$ e $t$ é um termo;
- uma *propriedade literal* $P(t, u)$, onde $P$ é uma propriedade em $\mathbf{V_T} \cup \mathbf{V_S}$ e $t$ e $u$ são termos;

[1] https://www.w3.org/Submission/SWRL/

- $u{=}f(t_1,\ldots,t_n)$, onde $f$ é um símbolo de função n-ária em **F** e $u$, $t_1,\ldots,t_n$ são termos;
- $\mathbf{p}(t_1,\ldots,t_n)$, onde **p** é um símbolo de predicado n-ário em $P$ e $t_1,\ldots,t_n$ são termos.

Os literais que usam símbolos de função binária (ou predicado) concretos podem ser escritos em notação infixa, por conveniência sintática. Um padrão de *tripla* é uma classe ou propriedade literal. Dizemos que uma tripla ***t*** *relaciona-se* com um padrão de tripla p sse:

- p é uma classe literal da forma $C(x)$, onde $x$ é uma variável, e ***t*** é da forma ($s$, rdf:type, $C$);
- p é uma propriedade literal da forma $P(x, y)$, onde $x$ e $y$ são variáveis e ***t*** é da forma ($s$, $P$, $o$).

Note que uma tripla não relaciona um literal da forma $u{=}f(t_1,\ldots,t_n)$ ou $\mathbf{p}(t_1,\ldots,t_n)$, onde $f$ é um símbolo de função n-ária em **F** e **p** é um símbolo de predicado n-ário em $P$. O *corpo de uma regra* $B$ é uma lista de literais, separados por vírgula. Quando necessário, usa-se "$B[x_1,\ldots,x_k]$" para indicar que as variáveis $x_1,\ldots,x_k$ ocorrem em $B$. Diz-se que $B$ é *sobre* um vocabulário **V** sse todas as classes e propriedades que ocorrem em $B$ são de **V**.

Como uma conveniência de notação, o corpo de uma regra $B$ pode incluir: 1) alguns *caminhos de propriedade* SPARQL, seja em notação prefixa ou infixa e; 2) alguns operadores SPARQL *unários, binários ou ternários*, seja em notação prefixa ou infixa. A Tabela 2 lista os caminhos de propriedade permitidos em nessa proposta e sua tradução. Nesta tabela, $P$, $P_1,P_2,\ldots,P_k$ são propriedades e $x_1,x_2,\ldots,x_k$ são variáveis que não ocorrem no corpo da regra $B$. A Tabela 3 exemplifica os operadores SPARQL em que $x,x_1$ e $x_2$ são variáveis ocorrendo no corpo da regra $B$.

Tabela 2 – Expressões de caminho permitidos.

| **Tipo de Propriedade de Caminho** | **Notação** | **Tradução** |
|---|---|---|
| *Caminho inverso* | $P\hat{}(t_1, t_2)$ | $P(t_2, t_1)$ |
| | $t_1\ P\hat{}\ t_2$ | |
| *Caminho de sequência* | $P_1/P_2/\ldots P_k(t_1, t_2)$ | $P_1(t_1, x_2)$; $P_2(x_2, x_3)$; … $P_k(x_k, t_2)$ |
| | $t_1\ P_1/P_2/\ldots P_k\ t_2$ | |
| *Caminho de comprimento fixo ($k \geq 0$)* | $P\{\mathrm{k}\}(t_1, t_2)$ | $P(t_1, x_2)$; $P(x_2, x_3)$; … $P(x_k, t_2)$ |
| | $t_1\ P\{\mathrm{k}\}\ t_2$ | ($P$ repetidas k vezes) |

Tabela 3 – Exemplos de operadores permitidos em SPARQL.

| Operadores SPARQL | Significado |
|---|---|
| $x_1 \| \| x_2$ | "ou" lógico |
| $x_1 \&\& x_2$ | "e" lógico |
| bound($x$) | retorna verdadeiro se $x$ é "bound" para um valor |
| $!x$ | não $x$ |

Uma *regra de mapeamento* de $\mathbf{O_S} = (\mathbf{V_S}, \Sigma_\mathbf{S})$ para $\mathbf{V_T}$, ou simplesmente uma regra de $\mathbf{O_S} = (\mathbf{V_S}, \Sigma_\mathbf{S})$ para $\mathbf{V_T}$, é uma expressão de uma das seguintes formas:

- $C(x) \leftarrow B[x]$, é chamado de *mapeamento de classe*, onde $C$ é uma classe em $\mathbf{V_T}$ e $B[x]$ é o corpo de uma regra sobre $\mathbf{V_S}$;
- $P(x, y) \leftarrow B[x, y]$, é chamado de um *mapeamento de propriedade*, onde $P$ é uma propriedade em $\mathbf{V_T}$ e $B[x, y]$ é o corpo de uma regra sobre $\mathbf{V_S}$.

A expressão à esquerda da seta é chamada de *alvo* e a da direita, *fonte*. Um *mapeamento simples* é uma regra de mapeamento de uma das seguintes formas:

- $C_\mathbf{T}(x) \leftarrow C_\mathbf{S}(x)$, onde $C_\mathbf{T}$ é uma classe em $\mathbf{V_T}$ e $C_\mathbf{S}$ é uma classe em $\mathbf{V_S}$.
- $P_\mathbf{T}(x, y) \leftarrow C_\mathbf{S}(x); P_\mathbf{S}(x, y)$, onde $P_\mathbf{T}$ é uma propriedade em $\mathbf{V_T}$, $P_\mathbf{S}$ é uma propriedade em $\mathbf{V_S}$ e $C_\mathbf{S}$ é o domínio de $P_\mathbf{S}$, definido na ontologia fonte $\mathbf{O_S}$.

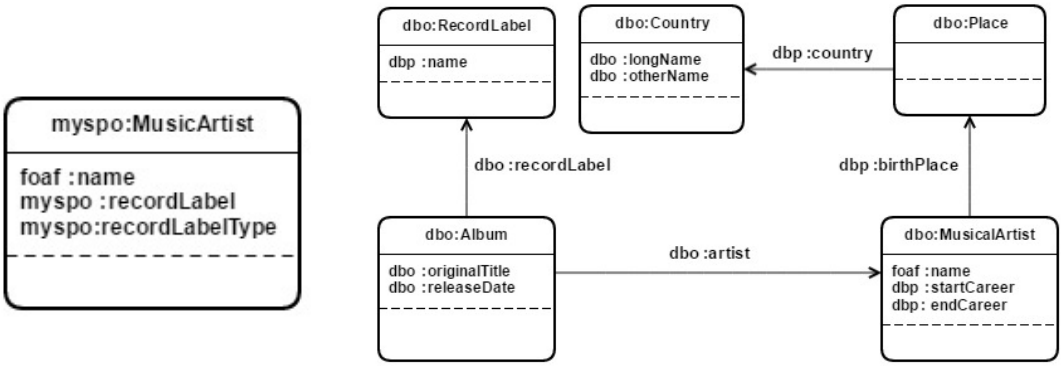
## 3.2 Exemplo de Motivação

Para facilitar o entendimento dos conceitos expostos, é apresentado um exemplo envolvendo o cenário musical. Para este exemplo são utilizadas três ontologias: uma ontologia alvo chamada MyMusic, ontologias fonte DBpedia[2] e MySpace[3]. Fragmentos das ontologias DBpedia (*dbo, dbp*) e MySpace (*myspo*) são apresentadas, respectivamente, nas Figuras 12a e 12b, enquanto MyMusic é mostrada na Figura 13. Todas as ontologias são apresentadas usando a notação de diagrama de classes UML. As classes são representadas como um retângulo com as propriedades de tipo de dados escritas em seu interior. As propriedades de objeto são representadas como arcos direcionados, em que a origem do arco é o domínio (no inglês *domain*) da propriedade e o destino do arco é o contradomínio da propriedade (em inglês *range*)

O fragmento da ontologia DBpedia apresentado na Figura 12b, provê informações relacionadas à música, tais como artistas e álbuns. As classes *dbo:RecordLabel*, *dbo:Country* e *dbo:Place* guardam informações sobre, respectivamente, a gravadora de um artista, seu país

[2] http://dbpedia.org/ontology/ e http://dbpedia.org/property/
[3] http://purl.org/ontology/myspace/

de origem e o local de nascimento do artista. Já as classes *dbo:MusicalArtist* e *dbo:Album* mantém, respectivamente, informações sobre o artista e seus álbuns. Os arcos entre as classes são rotulados com o nome da propriedade de objeto que as relaciona. Por exemplo, o arco entre *dbo:Album* e *dbo:MusicalArtist* está rotulado com a propriedade *dbo:artist* que relaciona um álbum ao artista que o gravou.




(a) Um fragmento simplificado da ontologia **MySpace**.

(b) Um fragmento simplificado da ontologia **Dbpedia**.

Figura 12: Ontologias Fonte

O fragmento da ontologia **MySpace** fornece uma representação RDF sobre informações de artistas que são usuários da rede social *MySpace*. **MyMusic** reutiliza termos de quatro vocabulários bem conhecidos: FOAF (**Friend of a friend**)[4], MO (**Music Ontology**)[5], DC (**Dublin Core**)[6] e **VCARD**[7]. Nós usamos o prefixo "moa" para novos termos definidos na ontologia **MyMusic**. Por exemplo, *moa:labelName* guarda o nome da gravadora e *moa:careerDuration* contém a duração da carreira de um artista.

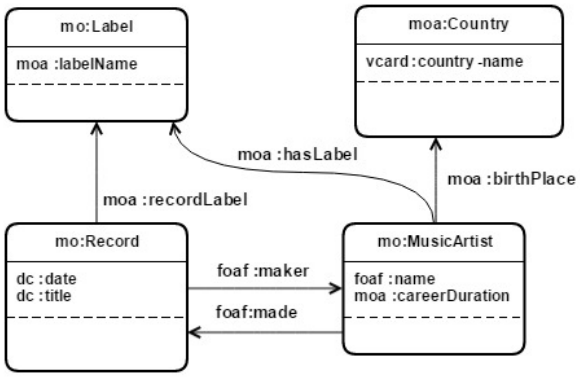



Figura 13: Ontologia Alvo. **MyMusic**.

[4] http://xmlns.com/foaf/0.1/
[5] http://purl.org/ontology/mo/
[6] http://purl.org/dc/elements/1.1/
[7] http://www.w3.org/2006/vcard/ns#

## 3.3 Exemplo de Regras de Mapeamento

Esta seção apresenta exemplos de regras de mapeamento usando o exemplo apresentado na Seção 3.2. A Tabela 4 mostra alguns exemplos de regras de mapeamentos. A regra de mapeamento *R*1 mapeia *mo:Record* para *dbo:Album*. *R*1 especifica um mapeamento complexo entre classes. *R*1 indica que cada tripla de *dbo:Album*, tal que o predicado *dbo:releaseDate* $>$ '2013-01-01' é satisfeito, produz uma tripla em *mo:Record*. A regra *R*2 especifica um mapeamento simples entre propriedades. *R*2 mapeia *dc:title*, cujo o domínio é *mo:Record*, para *dbo:originalTitle*, cujo o domínio é *dbo:Album*. *R*2 indica que cada tripla de *dbo:originalTitle* em *dbo:Album* produz uma tripla em *dc:title* cujo objeto é *dbo:originalTitle*.

Tabela 4 – Exemplo de Regras de Mapeamento.

| ID | Regra de Mapeamento |
|---|---|
| *R*1 | *mo:Record*(*s*) ← *dbo:Album*(*s*) ; *dbo:releaseDate*(*s*, *v*) $>$ '2013-01-01'^^xsd:date |
| *R*2 | *dc:title*(*s*, *v*) ← *dbo:Album*(*s*) ; *dbo:originalTitle*(*s*, *v*) |
| *R*3 | *mo:Label*(*u*) ← *myspo:MusicArtist*(*s*) ; *myspo:recordLabel*(*s*,*w*) ; concat(*s*, xpath:encode-for-uri(*w*), *u*) ; *myspo:recordLabelType*(*s*, *v*) = 'Maior' |
| *R*4 | *moa:labelName*(*u*, *w*) ← *myspo:MusicArtist*(*s*) ; *myspo:recordLabel*(*s*,*w*) ; concat(*s*, xpath:encode-for-uri(*w*), *u*) ; *myspo:recordLabelType*(*s*, *v*) = 'Major' |

A regra de mapeamento *R*3 mapeia *mo:Label* para *myspo:recordLabel*. *R*3 indica que cada tripla $\langle s$ *myspo:recordLabel* $w\rangle$, tal que o predicado *myspo:recordLabelType* = 'Major' é satisfeito, produz uma tripla $\langle u$ rdf:type *mo:Label*$\rangle$, onde o valor de *u* é obtido usando as funções *concat()* e *xpath:encode-for-uri()*. A função xpath:encode-for-uri() retorna o argumento *String* com certos caracteres de escape para permitir a *String* resultante ser usada como um segmento de caminho em um URI. A função *concat()* é usada aqui para gerar o novo URI *u* por concatenar o sujeito e o objeto da tripla de *myspo:recordLabel*. Isto assegura que os URIs das triplas produzidas pela regra *R*3 são únicos. A regra de mapeamento *R*4 mapeia *moa:labelName*, cujo domínio é *mo:Label*, para *myspo:recordLabel*, cujo domínio é *myspo:MusicArtist*. *R*4 indica que cada tripla $\langle s$ *myspo:recordLabel* $w\rangle$, tal que o predicado *myspo:recordLabelType* = 'Maior' é satisfeito, produz uma tripla $\langle u$ *moa:labelName* $w\rangle$, onde o valor de "*u*" é obtido da mesma maneira como na regra *R*3.

## 3.4 Discussão

Neste capítulo, apresentou-se o formalismo para especificação de mapeamentos entre ontologias. Este formalismo é uma forma conveniente para, manualmente, especificar os mapeamentos entre ontologias. Desta forma, as definições que foram descritas neste capítulo são a base para a criação dos mapeamentos. No corrente trabalho é apresentado um exemplo que ilustra a utilização deste formalismo. As ontologias fonte e a ontologia alvo apresentadas neste capítulo serão utilizadas ao longo deste trabalho como estudo de caso.

# 4 PADRÕES DE MAPEAMENTO

Neste capítulo, propomos o uso de Padrões de Mapeamentos como uma forma de facilitar a especificação do mapeamento entre ontologias. Tais padrões de mapeamento são essenciais para representar diversos problemas de incompatibilidades entre ontologias encontrados na literatura (KLEIN, 2001) pois, cada padrão de mapeamento representa uma solução genérica para um determinado problema de mapeamento. Especificamos os problemas de mapeamento por meio das Assertivas de Mapeamento (AM), as quais são uma representação mais abstrata e concisa das regras de mapeamento apresentadas no Capítulo 3. Na Seção 4.1, apresentamos o *template* que é utilizado para descrever os padrões de mapeamentos propostos. Na Seção 4.2 apresentamos uma biblioteca de padrões contendo os tipos mais comuns de mapeamentos entre ontologias. Por último, a Seção 4.3 apresenta as considerações finais do capítulo.

## 4.1 *Template* para Padrões de Mapeamento

Um *template* permite representar mapeamentos de uma maneira uniforme, usando um formato simples e genérico. Neste trabalho, os *templates* são utilizados para descrever problemas de mapeamento que ocorrem com alguma frequência. Os padrões de mapeamento apresentados no presente trabalho usam elementos clássicos da literatura de padrões de projeto (no inglês: *design patterns*) (Scharffe *et al.*, 2014). Em nossa abordagem, um padrão de mapeamento é constituído pelos seguintes elementos:

| | |
|---|---|
| Nome: | O nome do padrão |
| Problema | Uma descrição dos objetivos do padrão |
| Pré Condição: | Especificação das condições necessárias para aplicação do mapeamento |
| Solução: | Descrição da solução usando o formalismo de mapeamentos:<br>**- Regra de Mapeamento**,<br>**- AM**<br>**- Mapeamento R2R** |
| Exemplo | |
| Padrões relacionados | |

Os mapeamentos R2R e as regras de mapeamento que compõem os padrões de mapeamento já foram explanados nos Capítulos 2 e 3 respectivamente. AM é uma linguagem formal e declarativa usada para especificar mapeamentos entre ontologias de uma forma mais concisa e abstrata que as regras de mapeamento definidas no Capítulo 3. Existem propostas de AM (antes nomeadas como Assertiva de Correspodência (AC)) envolvendo outros modelos de dados. Por exemplo, mapeamentos entre esquemas XML (VIDAL *et al.*, 2001), e mapeamentos do relacional para RDF (VIDAL *et al.*, 2013). Neste trabalho, AM são utilizadas para especificar os tipos mais comuns de mapeamentos entre classes ou propriedades no modelos de dados RDF. As Assertivas de Mapeamentos podem ser definidas de três maneiras:

- Assertiva de Mapeamento de Classe (AMC), especifica um mapeamento de *classe* (ver definição no Capítulo 3);
- Assertiva de Mapeamento de Objeto (AMO), especifica um mapeamento de propriedade cujo o predicado alvo é uma *Propriedade de Objeto* (ver definição no Capítulo 3);
- Assertiva de Mapeamento de Tipo de Dados (AMD), especifica um mapeamento de propriedade cujo o predicado alvo é uma *Propriedade de Tipo de Dados* (ver definição no Capítulo 3).

## 4.2 Catálogo de Padrões de Mapeamento

Nesta seção, apresentamos nosso catálogo de padrões contendo os tipos mais comuns de mapeamentos entre ontologias. A Figura 14 fornece uma taxonomia para os padrões de mapeamento organizada por categorias.

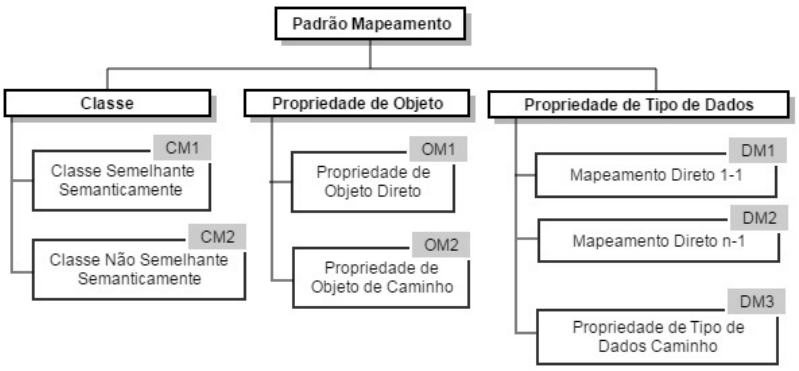



Figura 14: *Taxonomia dos Padrões de Mapeamento*.

Com base na taxonomia apresentada na Figura 14, definimos 7 padrões de mapeamentos e apresentamos estes padrões representativos de acordo com a biblioteca de padrões. Os

padrões estão de acordo com o *template* definido na Seção 4.1. A seguir mostramos a descrição completa de cada padrão de mapeamento.

### *4.2.1 Padrões de Mapeamento de Classe*

## Padrão MC1

- Nome: Padrão de Mapeamento de Classe Semelhantes Semanticamente
- Problema: Como especificar o mapeamento de instância de uma classe em uma ontologia fonte em instâncias de uma classe em uma ontologia alvo, que representam o mesmo objeto do mundo real?
- Pré-Condições:
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes em $\mathbf{V_S}$ e $\mathbf{V_T}$, respectivamente;
  - **f** é um predicado de seleção sobre instâncias de $C_{\mathbf{S}}$ (opcional).
- Solução:
  - **Regras de Mapeamento:** $C_{\mathbf{T}}(u) \leftarrow C_{\mathbf{S}}(u)$; $\mathbf{f}(u)$.
  - **AM:** $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}$ **/ f**
  - **Template de Mapeamento R2R:**

**Template T1**

# Mapeamento de Classe
# AMC $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}/\mathbf{f}$
mp:$\psi_C$
**a r2r:ClassMapping** ;
**r2r:prefixDefinitions** "prefixExp";
**r2r:sourcePattern** "?SUBJ a **S:**$C_{\mathbf{S}}$ sQuery" ;
**r2r:targetPattern** "?SUBJ a **T:**$C_{\mathbf{T}}$" .

- Exemplos
  - **Regras de Mapeamento:**

    R1: *mo:MusicArtist*($s$) $\leftarrow$ *dbo:MusicalArtist*($s$)

    R2: *mo:Record*($s$) $\leftarrow$ *dbo:Album*($s$) ; *dbo:releaseDate*($s$, $v$) > '2013-01-01'^^xsd:date
  - **Assertivas de Mapeamento:**

    $\psi_1$: *mo:MusicArtist* $\equiv$ *dbo:MusicalArtist*

    $\psi_2$: *mo:Record* $\equiv$ *dbo:Album* / *dbo:releaseDate* > "2013-01- 01"^^xsd:date
  - **Mapeamento R2R:**

```
# Mapeamento de Propriedade
mp:ψ2 a r2r:ClassMapping;
r2r:prefixDefinitions "mo:<http://purl.org/ontology/mo/> .
                       dbo:<http://dbpedia.org/ontology/>";
r2r:sourcePattern "?SUBJ a dbo:Album ; dbo:releaseDate ?t .
                   FILTER(?t > '2013-01-01'^^xsd:date)";
r2r:targetPattern "?SUBJ a mo:Record".
```

A AMC $\psi_1$ especifica que cada instância da classe *dbo:MusicalArtist* produz uma instância para a classe *mo:MusicArtist*. $\psi_2$ por sua vez especifica que cada instância da classe *dbo:Album*, tal que o filtro *dbo:releaseDate* $>$ "2013/01/01" é verdadeiro, produz instâncias para a classe *mo:Record*.

Os padrões de mapeamento possuem *templates* que ajudam na criação dos mapeamentos R2R. Cada *template* contém as variáveis *prefixExp* e *sQuery*. Estas variáveis auxiliam na geração dos mapeamentos R2R, pois:

- armazenam as informações necessárias para a criação dos mapeamentos R2R de acordo com os valores obtidos a partir das AM;
- as informações estão de acordo com as regras sintáticas estipuladas pela linguagem R2R.

A variável *prefixExp* armazena os prefixos de propriedades e classes presentes nos elementos de uma AM. A variável *sQuery* armazena todas as informações necessárias para construir a cláusula (*r2r:sourcePattern*) de um mapeamento R2R. O conteúdo da *sQuery* pode variar de acordo com o *template* utilizado. Assim, *sQuery* pode armazenar, tanto expressões de filtro, quanto expressões de caminhos. Por exemplo, quando trata-se do *template* T1 dentro do padrão MC1, a variável *sQuery* pode armazenar uma expressão de filtro como foi mostrado no mapeamento mp:$\psi_2$.

### Padrão MC2

- Nome: Padrão de Mapeamento de Classes Não Semelhantes Semanticamente
- Problema: Como especificar o mapeamento de instância de uma classe em uma ontologia fonte em instâncias de uma classe em uma ontologia alvo, que não representam o mesmo objeto do mundo real?
- Pré-Condições:
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes em $\mathbf{V_S}$ e $\mathbf{V_T}$, respectivamente;
  - $A_1,\ldots,A_n$ são propriedades de tipos de dados em $C_{\mathbf{S}}$.
- Solução:
  - **Regras de Mapeamento:** $C_{\mathbf{T}}(u) \leftarrow C_{\mathbf{S}}(s)$; generateUri[$\psi$]($s$,$u$); $\mathbf{f}(s)$.
  - **AM:** $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}[A_1,\ldots,A_n]$ / $\mathbf{f}$
  - **Template de Mapeamentos R2R:**

**Template T2**

# Mapeamento de Classe
# AMC $\psi$: $\boldsymbol{C_T} \equiv \boldsymbol{C_S}[\boldsymbol{A_1},\ldots,\boldsymbol{A_n}]$ / **f**
mp:$\psi_C$
**a r2r:ClassMapping** ;
**r2r:prefixDefinitions** "prefixExp";
**r2r:sourcePattern** "?SUBJ a **S:**$\boldsymbol{C_S}$ sQuery" ;
**r2r:targetPattern** "?s a **T:**$\boldsymbol{C_T}$";
**r2r:transformation** "?s = generateUri[$\psi_C$](?SUBJ)".

- Exemplos:
  - **Regras de Mapeamento**:

    R3: *mo:Label*($u$) ← *myspo:MusicArtist*($s$); generateUri[$\psi_3$]($s$,$u$)
  - **Assertivas de Mapeamento**:

    $\psi_3$: *mo:Label* ≡ *myspo:MusicArtist*[*myspo:recordLabel*]
  - **Mapeamento R2R**:

```
# Mapeamento de Classe
mp:ψ3
a r2r:ClassMapping ;
r2r:prefixDefinitions "mo:<...> .  myspo:<...>";
r2r:sourcePattern "?SUBJ a myspo:MusicalArtist";
r2r:targetPattern "?u a mo:Label";
r2r:transformation "?u = generateUri[ψ3](?SUBJ)".
```

A AMC $\psi_3$ especifica o mapeamento entre a classe *mo:Label* (Figura 13) e a propriedade *myspo:recordLabel* apresentada na Figura 12a. Neste caso, dizemos que *myspo:MusicArtist* [*myspo:recordLabel*] é uma classe embutida da classe *myspo:MusicArtist*. Os valores das variáveis utilizadas pelo *template* T2 no padrão de mapeamento MC2 são obtidos conforme descrito no padrão MC1.

### *4.2.2 Padrões de Mapeamento de Propriedades de Objeto*

#### Padrão MO1

- Nome: Padrão de Mapeamento de Propriedades de Objeto Direto de Classes Semelhantes Semanticamente
- Problema: Como especificar um mapeamento direto de propriedades de objeto em classes semelhantes semanticamente?
- Pré-Condições:
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes em $\mathbf{V_S}$ e $\mathbf{V_T}$, respectivamente;
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes semelhantes semanticamente. (AMC de $C_{\mathbf{T}}$ é da forma $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}$ / **f**);
  - $P_{\mathbf{T}}$ e $P_{\mathbf{S}}$ são propriedades de objetos em $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$, respectivamente. Os contradomínios de $P_{\mathbf{T}}$ e $P_{\mathbf{S}}$ são $C_{\mathbf{T}}$* e $C_{\mathbf{S}}$*;
  - $C_{\mathbf{T}}$* e $C_{\mathbf{S}}$* são classes semelhantes semanticamente.
- Solução:
  - **Regras de Mapeamento:**

    $P_{\mathbf{T}}(u,v) \leftarrow C_{\mathbf{S}}(u)$; $\mathbf{f}(u)$; $\varphi(u,s)$; $P_{\mathbf{S}}(s,v)$; $\mathbf{f}(v)$, onde o filtro **f** é opcional.
  - **AM:** $\psi$: $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/P_{\mathbf{S}}/\mathbf{f}$

    Restrições:

    Se a AMO é da forma $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/P_{\mathbf{S}}/\mathbf{f}$, então deve existir uma AMC que relaciona o domínio de

$P_{\mathbf{T}}$ com $C_{\mathbf{S}}$, e uma AMC que relaciona o contradomínio de $P_{\mathbf{T}}$ com o contradomínio de $P_{\mathbf{S}}$. Se a AMO é da forma $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/\mathbf{f}$, onde $\varphi$ é um caminho de $C_{\mathbf{S}}$ para $C_{\mathbf{R}}$, então deve existir uma AMC que relaciona o domínio de $P_{\mathbf{T}}$ com $C_{\mathbf{S}}$ e uma AMC que relaciona o contradomínio de $P_{\mathbf{T}}$ com $C_{\mathbf{R}}$.

– **Mapeamentos R2R:**

**Template T3**

# Mapeamento de Propriedade
# AMO $\psi$:$C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/P_{\mathbf{S}}$/ **f** mp:$\psi_P$
**a r2r:PropertyMapping** ;
**r2r:prefixDefinitions** "prefixExp";
**r2r:mappingRef** mp:$\psi_C$;
**r2r:sourcePattern** "?SUBJ sQuery";
**r2r:targetPattern** "?SUBJ **T:**$P_{\mathbf{T}}$ ?$P_{\mathbf{S}}$".

- Exemplo:
  - **Regras de Mapeamento:**

    R4: *moa:birthPlace*(*s*,*v*) ← *dbo:MusicalArtist*(*s*); *dbp:birthPlace / dbp:country*(*s*,*v*)
  - **Assertivas de Mapeamento:**

    $\psi_4$: *mo:MusicArtist* / *moa:birthPlace* ≡ *dbo:MusicalArtist* / $\varphi$, onde

    $\varphi$ = [*dbp:birthPlace*/*dbp:country*]
  - **Mapeamento R2R:**

```
# Mapeamento de Propriedade
mp:ψ4
a r2r:PropertyMapping;
r2r:prefixDefinitions "moa:<...> .  dbp:<...>";
r2r:mappingRef mp:ψ1;
r2r:sourcePattern "?SUBJ dbp:birthPlace ?t .  ?t dbp:country ?v";
r2r:targetPattern "?SUBJ moa:birthPlace ?v".
```

- Padrões Relacionados:

  Padrão de Mapeamento de Classe Semelhante Semanticamente (MC1).

A AMO $\psi_4$ especifica que cada caminho formado pelas triplas RDF obedecendo o padrão de tripla: <?*s dbp:birthPlace* ?*u*> <?*u dbp:country* ?*t*> produz triplas RDF com o padrão: <?*s moa:birthPlace* ?*t*>. O *template* T3 já foi apresentado no padrão MD1.

## Padrão MO2

- Nome: Padrão de Mapeamento de Propriedades de Objeto Direto de Classes Não Semelhantes Semanticamente
- Problema: Como especificar um mapeamento direto de propriedades de objeto em classes não semelhantes semanticamente?

- Pré-Condições:
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes em $\mathbf{V_S}$ e $\mathbf{V_T}$, respectivamente;
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes semelhantes semanticamente. (AMC de $C_{\mathbf{T}}$ é da forma $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}$ / **f**);
  - $P_{\mathbf{T}}$ é uma propriedade de objeto em $C_{\mathbf{T}}$, cujo contradomínio (range) é uma classe $C_{\mathbf{T}}$*.
  - $\varphi$ é um caminho de $C_{\mathbf{S}}$, cujo o contradomínio (range) é uma classe $C_{\mathbf{S}}$*
  - $C_{\mathbf{T}}$* e $C_{\mathbf{S}}$* são classes semelhantes semanticamente.
- Solução:
  - **Regras de Mapeamento:** $P_{\mathbf{T}}(s,u) \leftarrow C_{\mathbf{S}}(s)$; **f**$(s)$; generateUri[$\psi$]$(s,u)$, onde o filtro **f** é opcional.
  - **AM:** $\psi$: $C_{\mathbf{T}}$ $/P_{\mathbf{T}} \equiv C_{\mathbf{S}}[A_1,\ldots,A_n]$ / NULL

    Restrições:

    Deve existir uma AMC que relaciona o domínio de $P_{\mathbf{T}}$ com $C_{\mathbf{S}}$, e uma AMC que relaciona o contradomínio de $P_{\mathbf{T}}$ também com $C_{\mathbf{S}}$.
  - **Mapeamento R2R:**

**Template T5**

# Mapeamento de Propriedade
# AMO $\psi$: $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}[A_1 \ldots A_n]$/**NULL** mp:$\psi_P$
**a r2r:PropertyMapping** ;
**r2r:prefixDefinitions** "prefixExp";
**r2r:mappingRef** mp:$\psi_C$;
**r2r:sourcePattern** "?SUBJ a **T:**$C_{\mathbf{T}}$";
**r2r:targetPattern** "?SUBJ **T:**$P_{\mathbf{T}}$ ?s" ;
**r2r:transformation** "?s = generateUri[$\psi_R$](?SUBJ)".

- Exemplos:
  - **Regra de Mapeamento**

    R8: *moa:hasLabel*$(s,o)$ $\leftarrow$ *myspo:MusicArtist*$(s)$; generateUri[$\psi_8$]$(s,o)$
  - **Assertiva de Mapeamento**

    $\psi_8$: *mo:MusicArtist* / *moa:hasLabel* $\equiv$ *myspo:MusicArtist* [*myspo:recordLabel*] / NULL
  - **Mapeamento R2R**

```
# Mapeamento de Propriedade
mp:ψ8
a r2r:PropertyMapping;
  r2r:prefixDefinitions "mo:<...> .  moa:<...>";
  r2r:mappingRef mp:ψ3;
  r2r:sourcePattern "?SUBJ a mo:Label";
  r2r:targetPattern "?SUBJ moa:hasLabel ?r";
  r2r:transformation "?r = generateUri[ψ3](?SUBJ)".
```

- Padrões Relacionado:

  Padrão de Mapeamento de Classe Não Semelhante Semanticamente (MC1)

A AMO $\psi_8$ especifica que cada tripla RDF obedecendo o padrão de tripla: <?*s* a *mo:Label*> produz triplas da forma: <?*s moa:hasLabel* ?r>, onde ?r é o URI gerado para a classe embutida *myspo:MusicaArtist [myspo:recordLabel]*. As variáveis utilizadas no *template* T5 são similares às descritas para os *templates* T2 e T3.

### *4.2.3 Padrões de Mapeamento de Propriedades de Tipo de Dados*

## Padrão MD1

- Nome: Padrão de Mapeamento de Propriedades de Tipos de Dados Direto 1-1 de Classes Semelhantes Semanticamente
- Problema: Como especificar o mapeamento 1-1 de propriedades de tipos de dados de classes semelhantes semanticamente?
- Pré-Condições:
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes em $\mathbf{V_S}$ e $\mathbf{V_T}$, respectivamente;
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes semelhantes semanticamente. (AMC de $C_{\mathbf{T}}$ é da forma $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}$ / **f**);
  - $[P_1,\ldots,P_n]$ são propriedades de tipos de dados em $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$, respectivamente. Os contradomínios de $P_{\mathbf{T}}$ e $P_{\mathbf{S}}$ são $C_{\mathbf{T}}$* e $C_{\mathbf{S}}$*;
  - T é uma função de transformação que mapeia valores de $C_{\mathbf{T}}$* em valores do tipo $C_{\mathbf{S}}$*.
- Solução:
  - **Regras de mapeamento:**

    $P_{\mathbf{T}}(s,t) \leftarrow C_{\mathbf{S}}(s)$; $\mathbf{f}(s)$; $\varphi(s,o)$; $P_{\mathbf{S}}(o,v)$; $\mathbf{f}(v)$; $\mathtt{T}(v,t)$, onde o filtro **f** e a função T são opcionais.
  - **AM:** $\psi$: $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/P_{\mathbf{S}}/\mathbf{f}/\mathtt{T}$

    Restrições:

    Deve existir uma AMC que relaciona o domínio de $P_{\mathbf{T}}$ com o domínio de $P_{\mathbf{S}}$.
  - **Mapeamento R2R:**

| **T3** | **T6** |
|---|---|
| # Mapeamento de Propriedade<br># AMD/AMO $\psi$:$C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/P_{\mathbf{S}}$ / **f**<br>mp: $\psi_P$<br>**a r2r:PropertyMapping** ;<br>**r2r:prefixDefinitions** "prefixExp";<br>**r2r:mappingRef** mp: $\psi_C$;<br>**r2r:sourcePattern** "?SUBJ sQuery";<br>**r2r:targetPattern** "?SUBJ **T:**$P_{\mathbf{T}}$ ?$P_{\mathbf{S}}$". | # Mapeamento de Propriedade<br># AMD $\psi$: $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/P_{\mathbf{S}}$ / **f**/T<br>mp: $\psi_P$<br>**a r2r:PropertyMapping** ;<br>**r2r:prefixDefinitions** "prefixExp";<br>**r2r:mappingRef** mp: $\psi_C$ ;<br>**r2r:sourcePattern** "?SUBJ sQuery";<br>**r2r:targetPattern** "?SUBJ **T:**$P_{\mathbf{T}}$ ?v";<br>**r2r:transformation** "?v=FunctionExp" . |

- Exemplos:
  - **Regras de Mapeamento:**

    R5: *dc:title*(*s*,*v*) ← *dbo:Album*(*s*); *dbo:originalTitle* (*s*,*v*)

  - **Assertivas de Mapeamento:**

    $\psi_5$: *mo:Record /dc:title* $\equiv$ *dbo:Album/dbo:originalTitle*

  - **Mapeamento R2R:**

```
# Mapeamento de Propriedade
mp:ψ5
a r2r:PropertyMapping;
r2r:prefixDefinitions "dbo:<...> .  dc:<...>";
r2r:mappingRef mp:ψ2 ;
r2r:sourcePattern "?SUBJ dbo:originalTitle ?t";
r2r:targetPattern "?SUBJ dc:title ?t".
```

- Padrões Relacionados:

  Padrão de Mapeamento de Classe Semelhante Semanticamente (MC1)

A AMD $\psi_5$ especifica que toda tripla RDF no vocabulário fonte que obedece o padrão de tripla <?*s dbo:originalTitle* ?*t*> produz triplas RDF da forma: <?*s dc:title* ?*t*> no vocabulário alvo. O padrão MD1 é composto pelos *templates* T3 e T6. O *template* T3 utiliza as mesmas variáveis definidas nos padrões de mapeamento MC1 e MC2. Entretanto, neste padrão, a variável *sQuery* recebe valores obtidos a partir de uma AMD. Por exemplo, no *template* T3 *sQuery* pode receber além dos valores mostrado em MC1, o valor de uma expressão de caminho $\varphi$. Por sua vez, o *template* T6 também localizado no padrão de mapeamento MD1 utiliza além das variáveis já mencionadas, a variável FunctionExp. FunctionExp mantém a expressão usada para formar a cláusula *r2r:transformation*, que transforma os valores contidos na cláusula *2r:sourcePattern* antes que sejam carregados pela cláusula *r2r:targetPattern*.

### Padrão MD2

- Nome: Padrão de Mapeamento de Propriedades de Tipos de Dados Direto n-1 de Classes Semelhantes Semanticamente
- Problema: Como especificar o mapeamento n-1 de propriedades de tipos de dados de classes semelhantes semanticamente?
- Pré-Condições:
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes em $\mathbf{V_S}$ e $\mathbf{V_T}$, respectivamente;
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes semelhantes semanticamente. (AMC de $C_{\mathbf{T}}$ é da forma $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}$ / **f**);
  - $P_1,\ldots,P_n$ são propriedades de tipos de dados em $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$, respectivamente. Os contradomínios de $P_{\mathbf{T}}$ e $P_{\mathbf{S}}$ são $C_{\mathbf{T}}$* e $C_{\mathbf{S}}$*.
  - Té uma função de transformação que mapeia valores de $C_{\mathbf{T}}$* em valores do tipo $C_{\mathbf{S}}$*.

- Solução:
  - **Regras de Mapeamento:**

    $P_{\mathbf{T}}(s, t) \leftarrow C_{\mathbf{S}}(s)$; $\mathbf{f}(s)$; $\varphi(s, o)$; $P_{\mathbf{S}_1}(o, v_1)$, ..., $P_{\mathbf{S}_n}(o, v_n)$; concat($v_1, \ldots, v_n$,$v$); T($v$, $t$), onde o filtro **f** e a função T são opcionais.
  - **AM:** $\psi$: $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/\{P_{\mathbf{S}_1}, \ldots, P_{\mathbf{S}_n}\}/\mathtt{T}$

    Restrições:

    Deve existir uma AMC que relaciona o domínio de $P_{\mathbf{T}}$ com o domínio de $P_{\mathbf{S}_1}, \ldots, P_{\mathbf{S}_n}$.
  - **Template de Mapeamentos R2R:**

| **T7** | **T8** |
|---|---|
| # Mapeamento de Propriedade<br># AMD $\psi$: $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/\{P_{\mathbf{S}_1}, \ldots, P_{\mathbf{S}_n}\}$<br>mp: $\psi_P$<br>**a r2r:PropertyMapping** ;<br>**r2r:prefixDefinitions** "prefixExp";<br>**r2r:mappingRef** mp: $\psi_C$ ;<br>**r2r:sourcePattern** "?SUBJ sQuery";<br>**r2r:targetPattern** "?SUBJ **T:**$P_{\mathbf{T}}$ ?v";<br>**r2r:transformation** "?v=concat(?$P_{\mathbf{S}_1}, \ldots,$ ?$P_{\mathbf{S}_n}$)". | # Mapeamento de Propriedade<br># AMD $\psi$: $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/\varphi/\{P_{\mathbf{S}_1}, \ldots, P_{\mathbf{S}_n}\}$/T<br>mp: $\psi_P$<br>**a r2r:PropertyMapping** ;<br>**r2r:prefixDefinitions** "prefixExp";<br>**r2r:mappingRef** mp: $\psi_C$;<br>**r2r:sourcePattern** "?SUBJ sQuery";<br>**r2r:targetPattern** "?SUBJ **T:**$P_{\mathbf{T}}$ ?v";<br>**r2r:transformation** "?w=concat(?$P_{\mathbf{S}_1}, \ldots,$ ?$P_{\mathbf{S}_n}$)";<br>**r2r:transformation** "?v=FunctionExp". |

- Exemplos:
  - **Regras de Mapeamento**:

    R7: *moa:careerDuration*($s$, $v$)←*dbo:MusicalArtist*($s$); *dbp:startCareer*($s$, $v_1$);

    *dbp:endCareer*($s$, $v_2$); concat($v_1, v_2$, $v$)
  - **Assertivas de Mapeamento:**

    $\psi_7$: *mo:MusicArtist* / *moa:careerDuration* ≡ *dbo:MusicalArtist* / {*dbp:startCareer*, *dbp:endCareer*}
  - **Mapeamento R2R:**

```
# Mapeamento de Propriedade
mp:ψ7
a r2r:PropertyMapping ;
r2r:prefixDefinitions "moa:<...> .  dbp:<...>";
r2r:mappingRef mp:ψ1;
r2r:sourcePattern "?SUBJ dbp:startCareer ?startCareer;
                   dbp:endCareer ?endCareer";
r2r:targetPattern "?SUBJ moa:careerDuration ?v";
r2r:transformation "?v= concat(?startCareer,'-', ?endCareer)".
```

- **Padrões Relacionados:**

  Padrão de Mapeamento de Classes Semelhantes Semanticamente (MC1)

A $\psi_7$ especifica que cada conjunto de triplas RDF obedecendo os padrões de triplas (?SUBJ dbp:startCareer ?startCareer; dbp:endCareer ?endCareer) produzem triplas da forma: <?*t moa:careerDuration* ?n>, utilizando a função *concat(?startCareer, ?endCareer)*. A variável

?$v$ é o resultado da concatenação de ?startCareer e ?endCareer. Os *templates* T7 e T8 no padrão MD2 utilizam as mesmas variáveis descritas para os *templates* T3 e T6 no padrão de mapeamento MD1.

### Padrão MD3

- Nome: Padrão de Mapeamento Propriedade de Tipo de Dados com Caminho
- Problema: Como especificar o mapeamento de caminho de propriedades de tipos de dados de classes semelhantes semanticamente?
- Pré-Condições:
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes em $\mathbf{V_S}$ e $\mathbf{V_T}$, respectivamente;
  - $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ são classes semelhantes semanticamente. (AMC de $C_{\mathbf{T}}$ é da forma $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}$ / **f**);
  - $P_{\mathbf{T}}$ é uma propriedade de tipos de dados em $C_{\mathbf{T}}$, cujo contradomínio (range) é uma classe $C_{\mathbf{T}}$*;
  - $\varphi$ é um caminho de $C_{\mathbf{S}}$, cujo o contradomínio (range) é uma classe $C_{\mathbf{S}}$*;
  - $C_{\mathbf{T}}$* e $C_{\mathbf{S}}$* são classes semelhantes semanticamente.
- Solução:
  - **Regras de Mapeamento:** $P_{\mathbf{T}}(u,v) \leftarrow C_{\mathbf{S}}(s)$; $\mathbf{f}(s)$; $P_{\mathbf{S}}(s,v)$; generateUri[$\psi$]($s$,$u$), onde o filtro **f** é opcional.
  - **AM:** $\psi$**:** $\boldsymbol{C_{\mathbf{T}}/P_{\mathbf{T}}} \equiv \boldsymbol{C_{\mathbf{S}}}[\boldsymbol{A}_1 \ldots \boldsymbol{A}_n]/\boldsymbol{A_i}$**/ f**

    Restrições:

    Deve existir uma AMC que relaciona o domínio de $P_{\mathbf{T}}$ com $C_{\mathbf{S}}[A_1,\ldots,A_n]$.
  - **Mapeamento R2R:**

**Template T4**

```
# Mapeamento de Propriedade
# AMD ψ: C_T/P_T ≡ C_S[A_1 ... A_n]/A_i/ f
mp: ψ_P
a r2r:PropertyMapping ;
  r2r:prefixDefinitions "prefixExp";
 r2r:mappingRef mp:ψ_C ;
 r2r:sourcePattern "?SUBJ sQuery";
 r2r:targetPattern "?s T:P_T ?A_i" ;
 r2r:transformation "?s = generateUri[ψ_C](?SUBJ)".
```

- Exemplo:
  - **Regra de Mapeamento**

    R6: *moa:labelName*($u$,$w$) ← *myspo:MusicArtist*($s$); *myspo:recordLabel*($s$,$w$);

    generateUri[$\psi_3$]($s$,$u$)
  - **Assertiva de Mapeamento**

    $\psi_6$: *mo:Label /moa:labelName* ≡ *myspo:MusicArtist*[*myspo:recordLabel*] */myspo:recordLabel*

– **Mapeamento R2R**

```
# Mapeamento de Propriedade
mp:ψ6
a r2r:PropertyMapping ;
  r2r:prefixDefinitions "myspo:<...> .  moa:<...>";
  r2r:mappingRef mp:ψ3;
  r2r:sourcePattern "?SUBJ myspo:recordLabel ?r";
  r2r:targetPattern "?u moa:labelName ?r";
  r2r:transformation "?u = generateUri[ψ3](?SUBJ)".
```

- Padrões Relacionados:

  Padrão de Mapeamento de Classes Semelhantes Semanticamente (MC2).

A AMD $\psi_6$ especifica que cada tripla RDF obedecendo o padrão de tripla: <?*s* *myspo:recordLabel* ?r> produz triplas da forma: <?u *moa:labelName* ?r> no vocabulário da ontologia alvo. A variável ?*u* em $\psi_6$ é o URI que será gerado para a classe embutida *myspo:MusicArtist*[*myspo:recordLabel*] usando como base o valor do URI em ?*s*. As variáveis utilizadas no *template* T4 deste padrão são similares às descritas para os *templates* T3 e T7 nos padrões MD1 e MD2.

## 4.3 Discussão

Neste capítulo, foi apresentado um conjunto de padrões de mapeamento, os quais abordam problemas e soluções genéricas para vários tipos de mapeamentos entre ontologias. Ademais é mostrado, por meio de exemplos, que esses padrões podem ser utilizados para definir tipos específicos de mapeamento. Por exemplo, um padrão de mapeamento entre classes pode ser utilizado para gerar mapeamentos entre classes nos quais todas as instâncias de uma classe são mapeadas ou apenas algumas instâncias são mapeadas (i.e., o mapeamento pode incluir um filtro ou não). De modo semelhante, é mostrado que um mapeamento pode incluir (ou não) caminhos e funções de transformação.

# 5 RBA: R2R BY ASSERTIONS

Neste capítulo, apresentamos a ferramenta *R2R BY ASSERTIONS* (de forma abreviada RBA), que simplifica a tarefa de gerar mapeamentos R2R por utilizar os padrões de mapeamento propostos no Capítulo 4. Na Seção 5.1 apresentamos a arquitetura da ferramenta R2R By Assertions e descrevemos seus principais componentes. Na Seção 5.2, introduzimos os algoritmos que viabilizam a geração dos mapeamentos R2R de forma automática. A Seção 5.3, aborda a aplicação da ferramenta RBA utilizando para isso o estudo de caso apresentado no Capítulo 3. Por fim, a Seção 5.4 apresenta as considerações finais do capítulo.

## 5.1 Arquitetura da Ferramenta R2R By Assertions

A RBA, cuja arquitetura é apresentada na Figura 15, é composta por um conjunto de módulos, nomeadamente: *GUI*, *GRM*, *GR2R* e o *Mecanismo de mapeamento R2R*. Cada módulo é responsável pela implementação de uma etapa ou parte de uma etapa do processo de criação dos mapeamento R2R. Estes módulos estão interligados por meio de um processo, e na maioria das vezes a saída de um módulo é entrada para um próximo a ser executado.

O processo de criação automática dos mapeamentos R2R tem início com a definição dos esquemas alvo e fonte no módulo *GUI* (Figura 15, número 1). A saída do módulo *GUI* é um conjunto de assertivas de mapeamento, que são enviadas para os módulos *GRM* e *GR2R*. O módulo *GRM* (Figura 15, número 3) é responsável por gerar às regras de mapeamento. O módulo *GR2R* (Figura 15, número 2), por sua, vez é encarregado de gerar os mapeamentos R2R. Uma vez gerados os mapeamentos R2R, estes podem ser executados no módulo *Mecanismo de mapeamento R2R*. Esse mecanismo executa os mapeamentos gerados pelo módulo *GR2R* e envia os resultados para um banco de triplas RDF local. Embora todos os módulos façam parte da mesma arquitetura e estejam interligados, vale ressaltar que os módulos *GUI*, *GRM* e *GR2R* são executados diretamente pela RBA, enquanto o módulo *Mecanismo de mapeamento R2R* é executado pelo *framework* R2R (BIZER; SCHULTZ, 2010b).

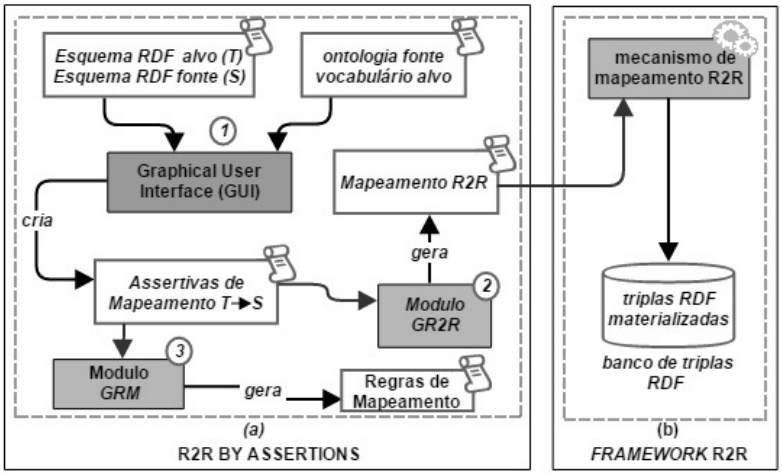



Figura 15: *Arquitetura da ferramenta R2R By Assertions*.

Para uma melhor compreensão, apresentamos a seguir a descrição de cada módulo na ordem em que devem ser utilizados, ou seja, desde a criação das assertivas de mapeamento até a execução dos mapeamentos R2R.

### *5.1.1 Módulos da Arquitetura RBA*

- **Graphical User Interface (GUI)**: Neste módulo, são configurados os esquemas alvo e fonte. Após esse passo inicial, o esquema da ontologia alvo (T) (onde T vem de alvo, em inglês *Target*) é apresentado por meio de uma estrutura de árvore. O esquema da ontologia fonte (S) (onde S vem de fonte, em inglês *Source*) também é exibido na estrutura de árvore. Entretanto, a árvore da ontologia fonte só é apresentada quando uma classe ou uma propriedade no esquema da ontologia alvo T é selecionada, de modo a poder definir as associações entre classe/propriedade da ontologia alvo e classe/propriedade de uma ontologia fonte. As associações criadas especificam as Assertivas de Mapeamento (AMs) entre o esquema RDF alvo e o esquema RDF fonte, as quais, são assim, criadas de forma gráfica usando o módulo *GUI*.
- **Gerador das Regras de mapeamento (GRM)**: Este módulo é o responsável por gerar as regras de mapeamento (ver Capítulo 3) associadas a cada AM gerada pelo módulo GUI. Assim, a saída deste módulo é uma *regra de mapeamento* na forma $C(\mathrm{x}) \leftarrow B[x]$ ou $P(\mathrm{x,y}) \leftarrow B[x,y]$.
- **Gerador dos Mapeamentos R2R (GR2R)**: Este módulo é responsável por gerar os mapeamentos R2R a partir de um conjunto de assertivas de mapeamento definidas pelo usuário no módulo *GUI*. A tarefa do módulo *GR2R* é percorrer todo o conjunto de assertivas

utilizando os Algoritmos 2, 3 e 4 apresentados detalhadamente na Seção 5.2. Para cada assertiva, é gerado um mapeamento R2R com base nos padrões de mapeamento definidos no Capítulo 4. A tarefa de gerar os mapeamentos R2R torna-se fácil com o auxílio das assertivas de mapeamento que satisfazem as restrições impostas pelos padrões de mapeamento, nos casos em que existem restrições, de acordo com o que foi definido na biblioteca de padrões no Capítulo 4.

- **Mecanismo de mapeamento R2R**: Este módulo é o responsável por gerar um conjunto de triplas RDF, as quais são instâncias da ontologia alvo, com base nos mapeamentos R2R gerados pelo módulo GR2R.

## 5.2 Algoritmos para Geração de Mapeamentos R2R

Nesta seção, apresentamos os algoritmos responsáveis por gerar automaticamente os mapeamentos R2R, utilizando para isso AMs pertencentes à um conjunto **M** de AM que satisfazem as restrições impostas pelos padrões de mapeamento propostos no Capítulo 4. Para cada AMC em **M**, o Algoritmo *G_R2RclassMapping* (Algoritmo 2) gera os mapeamentos de classe na linguagem R2R. Em seguida, para cada AMO em **M**, o Algoritmo *G_OMA_R2RpropMapping* (Algoritmo 3) gera os mapeamentos de propiedade de objeto em R2R. Por último, para cada AMD em **M**, o Algoritmo *G_DMA_R2RpropMapping* (Algoritmo 4) gera os mapeamentos de propriedade de tipo de dados em R2R.

**Algoritmo 1:** Gerando mapeamento R2R a partir de AM.

```
1 for classe C_T em O_T do
2 |   G_R2RclassMapping(AMC) AMCs de C_T;
3 |   for propriedade de objeto P_T cujo o domínio é C_T do
4 |   |   G_OMA_R2RpropMapping(AMO, AMC) AMO de P_T;
5 |   end
6 |   for propriedade de tipo de dados P_T cujo o domínio é C_T do
7 |   |   G_DMA_R2RpropMapping(AMD, AMC) AMD de P_T;
8 |   end
9 end
```

### *5.2.1 Algoritmo para Gerar Mapeamento de Classe em R2R*

O Algoritmo *G_R2RclassMapping* (Algoritmo 2) recebe como entrada um conjunto AMC de **M**. Para cada AMC, o algoritmo obtém os prefixos das classes que irão compor o mapeamento R2R, utilizando para isso a função *getPrefixes()*. Em seguida, o algoritmo verifica se a AMC é um padrão MC Regular (linha 5) ou um padrão MC Embutida (linha 10). Em qualquer um dos casos, o algoritmo verifica se um filtro **f** foi definido. Em caso afirmativo, a expressão de filtro **f** deve ser convertida para a sintaxe de filtro da linguagem R2R. Esta tarefa é realizada utilizando a função *FilterExp($\psi_C$)*. O resultado de *FilterExp()* é acrescentado à variável *sQuery*, descrita no Capítulo 4. Caso a AMC seja um padrão MC Regular, o *template* T1 é utilizado para gerar o mapeamento da classe $C_{\mathbf{T}}$, caso contrário é utilizado o *template* T2. Ambos os *templates* T1 e T2 foram apresentados no Capítulo 4.

**Algoritmo 2:** G_R2RclassMapping().

```
1  Conjunto de AMC de C_T contidos em M.
2  for AMC ψ_C de M do
3      sQuery = NULL;
4      prefixExp = getPrefixes(ψ_C) Obtém os prefixos presentes na AMC;
5      if ψ_C é da forma ψ_C: C_T ≡ C_S/f then
6          if f≠ NULL then
7              sQuery = FilterExp(ψ_C);
8          end
9          use o template T1;
10     else
11         if f≠ NULL then
12             sQuery = FilterExp(ψ_C);
13         end
14         use o template T2;
15     end
16 end
```

A Figura 16 apresenta um mapeamento R2R entre as classes *mo:Record* e *dbo:Album*. Esse mapeamento foi gerado utilizando o Algoritmo *G_R2RclassMapping*, tendo como parâmetro de entrada a AMC $\psi_2$ (apresentada no Capítulo 4). Visto que $\psi_2$ é do tipo padrão MC Regular, o algoritmo utilizou o *template* T1 apresentado no Capítulo 4.

```
01.  mp:ψ2
02.  a r2r:ClassMapping;
03.   r2r:prefixDefinitions ‘‘mo:  <http://purl.org/ontology/mo/>.
                              dbo:  <http://dbpedia.org/ontology/>’’;
04.   r2r:sourcePattern ‘‘?SUBJ a dbo:Album; dbo:releaseDate ?t’’.
                         FILTER(?t > ‘2013-01-01’^^xsd:date)";
05.   r2r:targetPattern "?SUBJ a mo:Record".
```

Figura 16: Mapeamento R2R $\psi_2$ gerado usando o *template* T1.

A cláusula ***r2r:prefixDefinitions*** (linha 3) na Figura 16 corresponde a cláusula (*r2r:prefixDefinition* “prefixExp”;) no *template* T3. A expressão contida nesta cláusula foi obtida a partir dos prefixos de $\psi_4$ e informações contidas nas ontologias **MyMusic** e **DBpedia** utilizando a função getPrefix($\psi_4$).

### *5.2.2 Algoritmo para Gerar um Mapeamento de Propriedade de Objeto em R2R*

O Algoritmo *G_OMA_R2RpropMapping* (Algoritmo 3) recebe como entrada um conjunto AMO de **M**. Para cada AMO, o algoritmo obtém os prefixos que irão compor o mapeamento, utilizando para isso a função *getPrefixes()*. Em seguida, o algoritmo verifica se a AMO é um padrão AMO Regular (linha 6) ou um padrão AMO Embutida (linha 14). Para o caso onde a AMO é um padrão MO Regular o algoritmo verifica se existe uma expressão de caminho $\varphi$. Em caso afirmativo, a expressão de caminho $\varphi$ é convertida para a sintaxe de caminho da linguagem R2R usando a função de caminho $\varphi$ e acrescentada a variável *sQuery*. No passo seguinte, o algoritmo verifica se um filtro **f** foi definido. Em caso afirmativo, a expressão de filtro **f** deve ser convertida para a sintaxe de filtro da linguagem R2R. Isto é realizado utilizando a função *FilterExp($\psi_C$)*. O resultado de *FilterExp()* é acrescentado a variável *sQuery*. Se a AMO é um padrão MO Regular, então o *template* T3 (linha 13) é invocado pelo algoritmo. Caso contrário, o *template* T5 (linha 15) é invocado. Ambos os *templates* T3 e T5 foram apresentados no Capítulo 4.

**Algoritmo 3:** G_OMA_R2RpropMapping().

```
1  Conjunto de AMO de P_T e as AMC
2  for AMO ψ_P de M do
3      sQuery = NULL;
4      prefixExp = getPrefixes(ψ_P) Obtém os prefixos presentes na AMO;
5      ψ_C = foundCMA(ψ_P) obtém o domínio da AMO P_S;
6      if ψ_P é da forma ψ_p: C_T/P_T ≡ C_S/φ/P_S/ f then
7          if φ ≠ NULL then
8              sQuery = Path(φ) + "; S:P_S  ?P_S";
9          else
10             sQuery = "S:P_S  ?P_S";
11         if f ≠ NULL then
12             sQuery = sQuery + FilterExp(ψ_P)
13         use o template T3 ;
14     else
15         use o template T5 ;
```

A Figura 17 apresenta o mapeamento R2R entre as propriedades *moa:birthPlace* e *dbp:country*. Esse mapeamento foi gerado utilizando o Algoritmo *G_OMA_R2RpropMapping* tendo como parâmetro a AMC $\psi_4$ (apresentada no Capítulo 4). Visto que, $\psi_4$ é um padrão MO Regular, o Algoritmo 3 utilizou o *template* T3.

```
01.  mp:  ψ4
02.  a r2r:PropertyMapping;
03.    r2r:prefixDefinitions "moa:  <...> .  dbp:  <...>";
04.    r2r:mappingRef mp:ψ1 ;
05.    r2r:sourcePattern "?SUBJ dbp:birthPlace ?birthPlace .
                          ?birthPlace dbp:country ?country";
06.    r2r:targetPattern "?SUBJ moa:birthPlace ?country".
```

Figura 17: Mapeamento R2R $\psi_4$ gerado usando o *template* T3.

A cláusula ***r2r:prefixDefinitions*** (linha 3) na Figura 17 corresponde a cláusula (*r2r:prefixDefinition* "prefixExp";) no *template* T3. A expressão contida nesta cláusula foi obtida a partir de $\psi_4$ e informações contidas nas ontologias **MyMusic** e **DBpedia**. Para isso foi utilizado a função getPrefix($\psi_4$). A cláusula ***r2r:mappingRef*** (linha 4) na Figura 17 equivale a (*r2r:mappingRef* mp:$\psi_C$) em T3. Esta expressão foi obtida a partir de uma AMC definida anteriormente, no caso a AMC $\psi_1$. A cláusula ***r2r:sourcePattern*** (linha 5) na Figura 17 é equivale a cláusula (*r2r:sourcePattern* "?SUBJ a s:Cs; s:Ps sQuery") no *template* T3. Esta expressão

foi obtida a partir do lado direito da AMO $\psi_4$, sendo que a expressão "*dbp:birthPlace* ?*d* . ?*d dbp:country* ?*country*" é a saída da função *Path*($\varphi$). Finalmente a cláusula ***r2r:targetPattern*** (linha 6) na Figura 17 corresponde a (*r2r:targetPattern* "?SUBJ a T:Ct; T:Pt") em T3. A expressão contida nesta cláusula foi obtida a partir do lado esquerdo da AMO $\psi_4$, sendo que a propriedade $\mathbf{T{:}}\boldsymbol{P}_{\mathbf{T}}$ no *template* T3 é substituída pela propriedade *moa:birthPlace*.

### *5.2.3 Algoritmo para Gerar um Mapeamento de Propriedade de Tipo de Dados em R2R*

O Algoritmo *G_DMA_R2RpropMapping* (Algoritmo 4), recebe como entrada um conjunto AMD de **M**. Para cada AMD do conjunto **M**, o algoritmo obtém os prefixos que irão compor o mapeamento, usando para isso a função *getPrefixes()*. Em seguida, o algoritmo verifica se a AMD é uma AMD Regular (linha 5), um padrão AMD Embutida (linha 20) ou um padrão MD Múltipla (linha 21). Para os casos onde a AMD é um padrão MD Regular ou padrão MD Múltipla o algoritmo verifica se existe uma expressão de caminho $\varphi$. Em caso afirmativo, a expressão de caminho $\varphi$ é convertida para a sintaxe de caminho da linguagem R2R usando a função de caminho $\varphi$ e acrescentada a variável *sQuery*. No passo seguinte, o algoritmo verifica se um filtro **f** foi definido. Em caso afirmativo, a expressão de filtro **f** deve ser convertida para a sintaxe de filtro da linguagem R2R. Isto é realizado utilizando a função *FilterExp*($\psi_C$). O resultado de *FilterExp()* é acrescentado a variável *sQuery*. Em seguida o algoritmo verifica se há uma função T. Em caso afirmativo, a expressão em T é convertida para a sintaxe da linguagem R2R, mas com atenção aos nomes das variáveis usadas em sQuery. O *template* selecionado muda de acordo com a existência ou não da função `T`. Caso a AMD seja um padrão MD Regular existe uma função de transformação, o Algoritmo 4 utiliza o *template* T6, caso contrário, usa o *template* T3 é utilizado. Quando a AMD é um padrão AMD Embutida o *template* invocado é o T4. Se à AMD é um padrão AMD Múltipla e existe uma função de transformação, então o Algoritmo 4 utiliza o *template* T8, caso contrario, usa o *template* T7.

**Algoritmo 4:** G_DMA_R2RpropMapping().

```
1  for AMD ψ_P de M do
2      sQuery = NULL;
3      prefixExp = getPrefixes(ψ_P) Obtém os prefixos presentes na AMD;
4      ψ_C = foundCMA(ψ_P) Obtém o domínio da AMD P_S;
5      if ψ_P é da forma ψ_P:C_T/P_T ≡ C_S/φ/P_S/f/T then
6          if φ ≠ NULL then
7              sQuery = Path(φ) + "; S:P_S  ?P_S";
8          else
9              sQuery = "S:P_S  ?P_S";
10         end
11         if f ≠ NULL then
12             sQuery = sQuery + FilterExp(ψ_P);
13         end
14         if T ≠ NULL then
15             FunctionExp = Function(T);
16             use o template T6;
17         else
18             use o template T3;
19         end
20     else
21         if ψ_P é da forma ψ_P:C_T/P_T ≡ C_S[A_1,...,A_n]/A_i then
22             sQuery = "S:A_i  ?A_i";
23             use o template T4;
24         else
25             if φ≠ NULL then
26                 sQuery = Path(φ) + "; S:P_S1  ?P_S1;...;S:P_Sn  ?P_Sn";
27             else
28                 sQuery = "S:P_S1  ?P_S1;...;S:P_Sn  ?P_Sn";
29             end
30         end
31         if T≠ NULL then
32             FunctionExp = Function(T);
33             use template T8;
34         else
35             use template T7;
36         end
37     end
38 end
```

A Figura 18 apresenta o mapeamento R2R entre a propriedade *moa:careerDuration* mantida na ontologia alvo e as propriedades *dbp:startCareer* e *dbo:endCareer* que fazem parte da ontologia fonte. Esse mapeamento foi gerado pelo Algoritmo *G_DMA_R2RpropMapping* tendo como parâmetro $\psi_7$ (apresentada no Capítulo 4). Visto que $\psi_7$ é um padrão MD Múltipla,

e não tem uma função de transformação, o algoritmo utilizou o *template* T7 apresentado no Capítulo 4.

```
01.  mp:  ψ7
02.  a r2r:PropertyMapping;
03.    r2r:prefixDefinitions "moa:  <...> .  dbp:  <...>";
04.    r2r:mappingRef mp:ψ1 ;
05.    r2r:sourcePattern "?SUBJ dbp:startCareer ?startCareer ;
                          dbp:endCareer ?endCareer";
06.    r2r:targetPattern "?SUBJ moa:careerDuration ?v";
07.    r2r:transformation "?v= concat(?startCareer, ?endCareer)".
```

Figura 18: Mapeamento R2R $\psi_7$ gerado usando o *template* T7.

A cláusula ***r2r:prefixDefinitions*** (linha 3) na Figura 18 corresponde a cláusula (*r2r:prefixDefinition* "prefixExp";) no *template* T7. A expressão contida nesta cláusula foi obtida a partir de $\psi_7$ e as informações contidas nas ontologias **MyMusic** e **DBpedia** utilizando a função getPrefix($\psi_7$). A cláusula ***r2r:mappingRef*** (linha 4) na Figura 17 equivale a (*r2r:mappingRef* mp:$\psi_C$) em T3. Esta expressão foi obtida a partir de uma AMC definida anteriormente, no caso a AMC $\psi_1$. A cláusula *r2r:sourcePattern* (linha 5) na Figura 18 corresponde a cláusula (*r2r:sourcePattern* "?SUBJ a s:Cs; s:Ps sQuery") no *template* T7. Esta expressão foi obtida a partir do lado direito da AMD $\psi_7$ (linha 28 do Algoritmo 4). A cláusula *r2r:targetPattern* (linha 6) na Figura 18 equivale a (r2r:targetPattern "?SUBJ a t:Ct; t:Pt") em T7. Esta expressão foi obtida a partir do lado esquerdo da AMD $\psi_7$, sendo que **T:$P_{\mathbf{T}}$** no *template* T7 foi substituída pela propriedade *moa:careerDuration*. Por último, a cláusula *r2r:transformation* (linha 7) na Figura 18 corresponde a *r2r:transformation* "?v = concat($P_{\mathbf{S}_1}, \ldots, P_{\mathbf{S}_n}$)" no *template* T7. Os valores das propriedades em *r2r:transformation* foram obtidos a partir do lado direito da AMD $\psi_7$.

## 5.3 Implementação do Estudo de Caso

Nesta seção, abordamos a aplicação da ferramenta RBA ao estudo de caso introduzido no Capítulo 3. Apresentamos passo a passo a interação do usuário com a ferramenta, desde a configuração das ontologias e configurações R2R, passando pela geração dos mapeamentos R2R e regras de mapeamento até o envio dos resultados para um banco de triplas RDF local.

### *5.3.1 Configuração Inicial para Criação de um Mapeamento*

Uma nova sessão na ferramenta têm início utilizando o módulo *GUI*, onde são definidas as ontologias fonte e alvo. Para ontologia alvo, usamos a ontologia *My Music* apresentada na (Figura 13), a qual está definida no arquivo *MyMusic_ontology*. Como ontologia fonte, utilizamos um fragmento da ontologia *DBpedia*, apresentada na (Figura 12b), a qual está definida no arquivo DBpedia_ontology. Os esquemas de ambas as ontologias estão armazenados localmente, entretanto, a ferramenta também permite que as ontologias sejam carregadas por meio de uma URL, como representado na Figura 19.

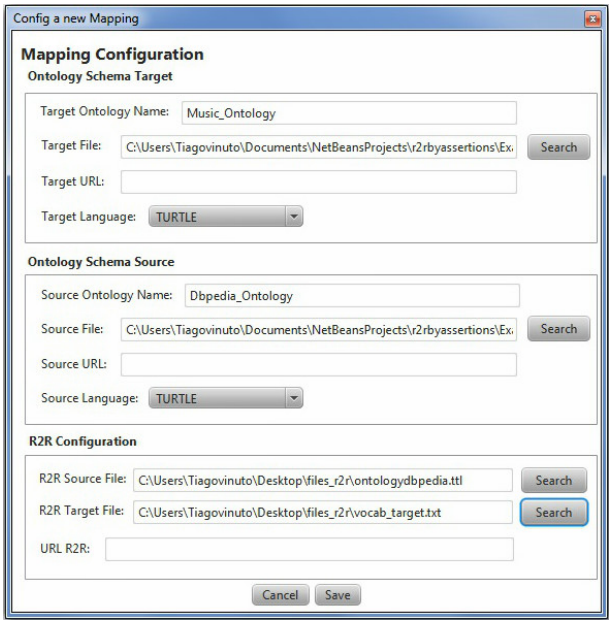



Figura 19: *Configurações das ontologias na RBA e Parametros R2R.*

Além de indicar qual a ontologia alvo, qual ontologia fonte e onde estão localizados os arquivos também é necessário escolher a linguagem na qual as ontologias estão descritas. Por exemplo TURTLE, RDF/XML e N-TRIPLES, entre outras. Nesse exemplo, *MyMusic_ontology* e *DBpedia_ontology* foram escritas usando a linguagem TURTLE.

O próximo passo é definir as *configurações R2R*, que também são mostradas na Figura 19. Para essas configurações são requisitados dois parâmetros: (i) um arquivo com triplas RDF (*R2R source file*), as quais são instâncias da ontologia fonte, em nosso exemplo *DBpedia_ontology* e (ii) um arquivo onde será armazenado a nossa ontologia alvo (*R2R target file*). As triplas RDF contidas no arquivo fonte das configurações R2R devem ser definidas de acordo com o esquema da ontologia fonte, previamente selecionada nas configurações de mapeamento (*Ontology Schema Source*). Ambos os arquivos são armazenados localmente.

Concluído o resultado da configuração inicial, o componente GUI apresenta à esquerda da tela inicial uma árvore, que representa o esquema alvo (T) conforme visto na Figura 20.

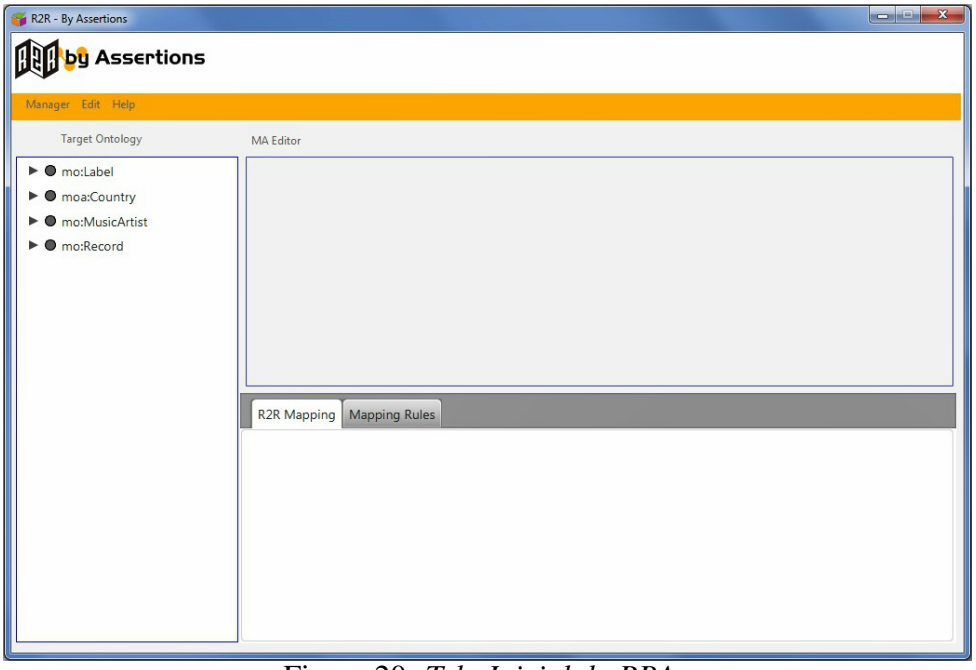



Figura 20: *Tela Inicial da RBA* .

### *5.3.2 Criação das Assertivas de Mapeamento*

Nesta seção, apresentamos como a ferramenta *RBA* facilita a tarefa de criar as assertivas de mapeamento. Essa tarefa é realizada em 3 etapas: (i) seleção de um elemento na árvore T que representa a ontologia alvo; (ii) associação entre o elemento da ontologia alvo selecionado no item (i) com um elemento da ontologia fonte, essa associação dá origem a uma assertiva de mapeamento e (iii) inclusão da assertiva de mapeamento criada na lista de assertivas de mapeamento associada ao item da ontologia alvo selecionado no item (i). A Figura 21 exibe a criação de uma assertiva de mapeamento de classe quando é selecionado na árvore T a classe mo:Record (Figura 21a à esquerda) e na árvore S é selecionado a classe dbo:Album (Figura 21b).

A árvore S é exibida quando o botão “Load Source” é pressionado e a escolha do elemento na árvore S termina quando o botão “Finish” é pressionado. A AMC sendo criada, pode ser visualizada na caixa de texto, localizada na parte inferior do editor de AM. Quando o botão “*Add*” localizado ao lado da caixa de texto é selecionado, a AMC é incluída na lista de AMCs de mo:Record.

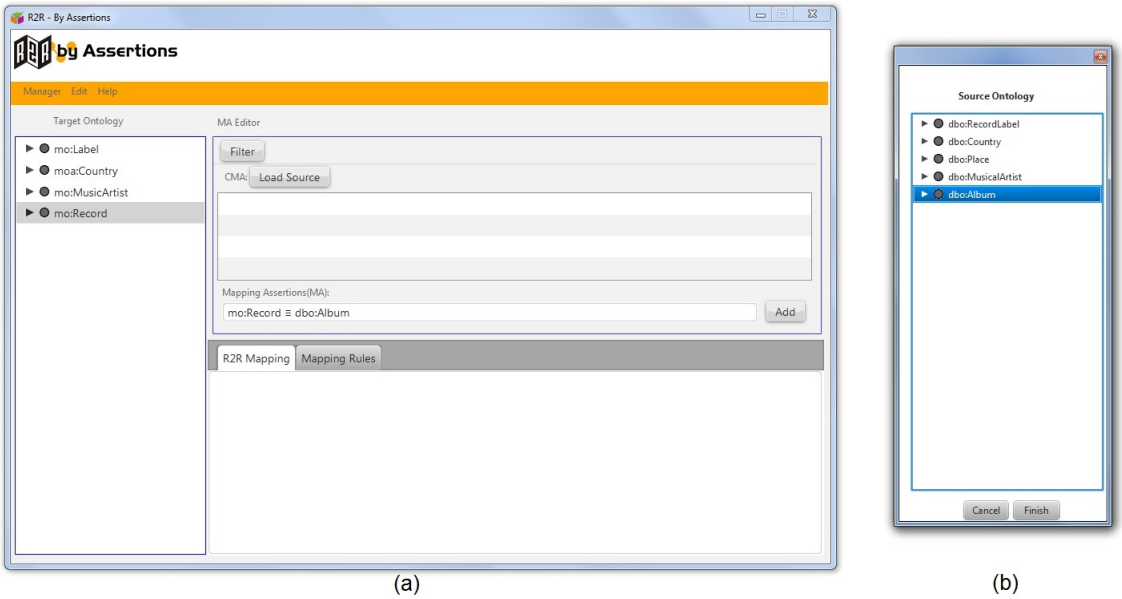



Figura 21: Seleção de classes nas ontologias Alvo e Fonte

É possível criar um filtro de seleção em uma AMC. A criação de um filtro dá-se início quando pressionamos o botão *Filter*, localizado no canto superior do *MA_Editor*. Aparece, então uma janela composta por duas seções: *Filter Condition* e *Filter[Query/Condition]* Figura 22. *Filter Condition* disponibiliza os vários elementos que podem constituir um filtro, enquanto Filter[Query/Condition] contém uma caixa de texto que exibe o filtro em construção. Quando o botão "Ok" é pressionado a edição do filtro termina e este é incluído na definição da AMC sendo criada. A Figura 23 mostra a lista de assertivas de mapeamento atualizada com a inclusão da assertiva com filtro.

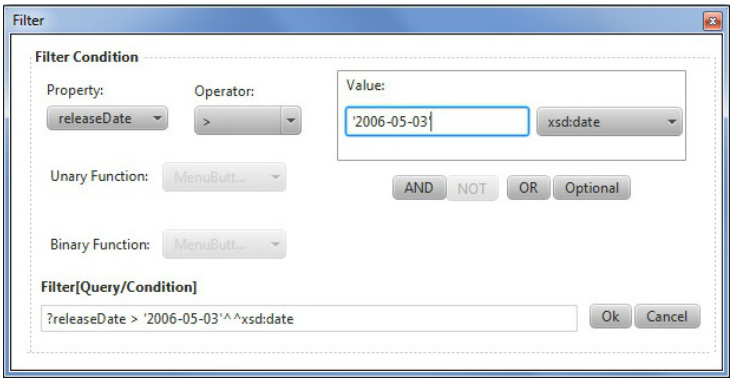



Figura 22: *Inclusão de um filtro à assertiva de mapeamento.*

Esta atualização ocorre após a seleção da opção "*Add*". Desta forma, é possível criar de forma fácil e intuitiva, as assertivas exibida no Capítulo 4.

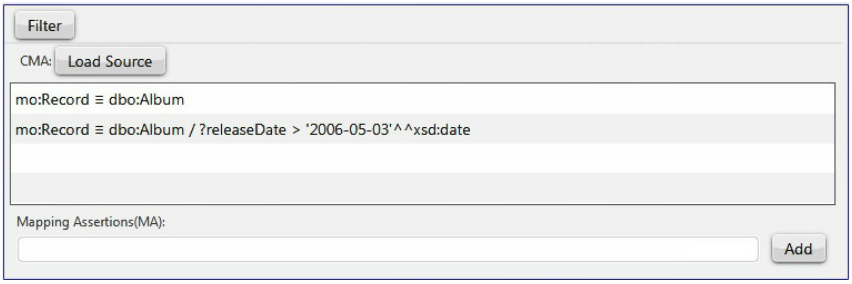



Figura 23: *Adição da assertiva criada a lista assertivas*.

A Figura 24 exibe a criação da AMO $\psi_4$ entre a propriedade *moa:birthPlace*, cujo o domínio é *mo:MusicArtist* (árvore T), e a propriedade de objeto *mo:country*, cujo domínio é *dbo:Place* (árvore S). O processo de criação da AMO é similar ao descrito para a construção de uma AMC. Ele tem início com a seleção da propriedade de objeto *moa:birthPlace* na *árvore* T. Em seguida, na *árvore* S devemos navegar pelos elementos a partir da classe *dbo:MusicalArtist* passando por um caminho contendo a propriedade (*dbp:country*) até chegar à classe *dbo:Place* e, logo depois, selecionar a propriedade de objeto *mo:country*. Por fim, pressionando o botão (*Finish*) o processo de escolha dos itens na *árvore S* é finalizado. A assertiva de mapeamento será então acrescentada à caixa de texto localizada no *MA_Editor* e, em seguida, quando pressionamos o botão "Add", a AMO é inserida na lista de assertivas de mapeamento da propriedade de objeto. Vale ressaltar que, caso não tenha sido criada uma assertiva de mapeamento de classes com o domínio da propriedade *moa:birthPlace*, a interface apresentará uma mensagem de erro. Como apresentado na Figura 25.

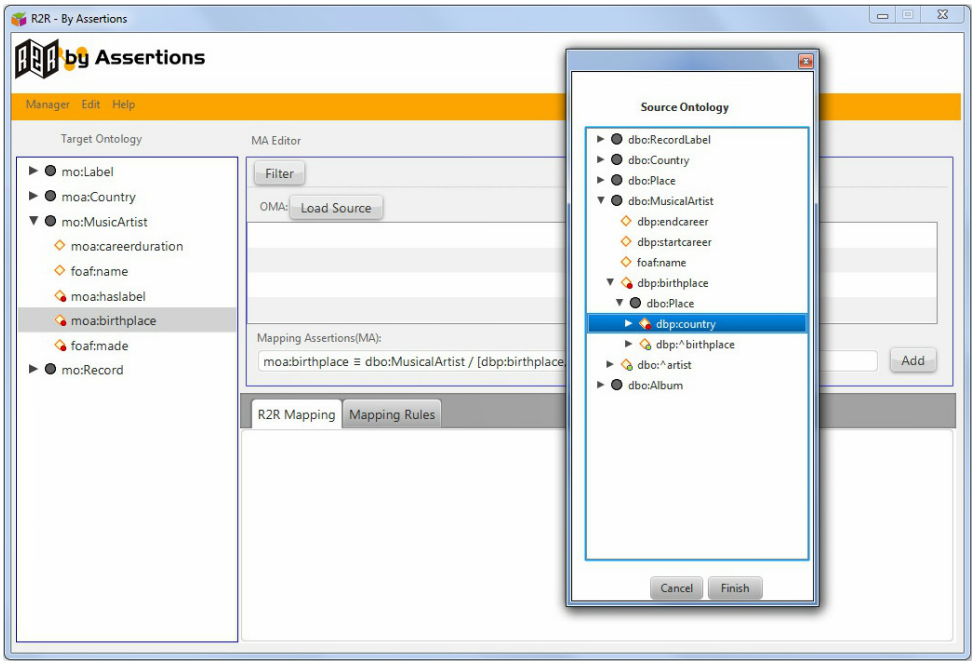



Figura 24: *Seleção de uma propriedade de objeto na árvore fonte*.


Figura 25: *Mensagem de erro ao adicionar uma propriedade de objeto*.

Na etapa de construção de uma AMO também é possível adicionar um filtro. Entretanto, caso nenhum filtro seja adicionado, a AMO poderá ser inserida na lista de assertivas de mapeamento pressionando o botão (*Add*).A Figura 26, exibe AMO adicionada a lista relacionada a propriedade *moa:birthPlace*.

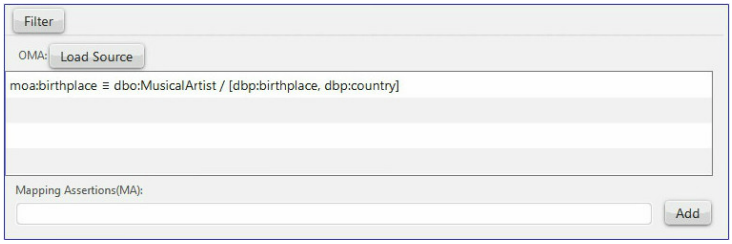



Figura 26: *Assertiva de objeto adicionada a lista de assertivas*.

Uma vez criada uma assertiva de mapeamento, é possível criar/visualizar o mapeamento R2R (Figura 27) ou a regra de mapeamento (Figura 28) relacionado com cada assertiva mapeamento. No primeiro caso, basta selecionar a aba “R2R Mapping”. Para criar uma regra de mapeamento basta selecionar a aba “Mapping Rule”

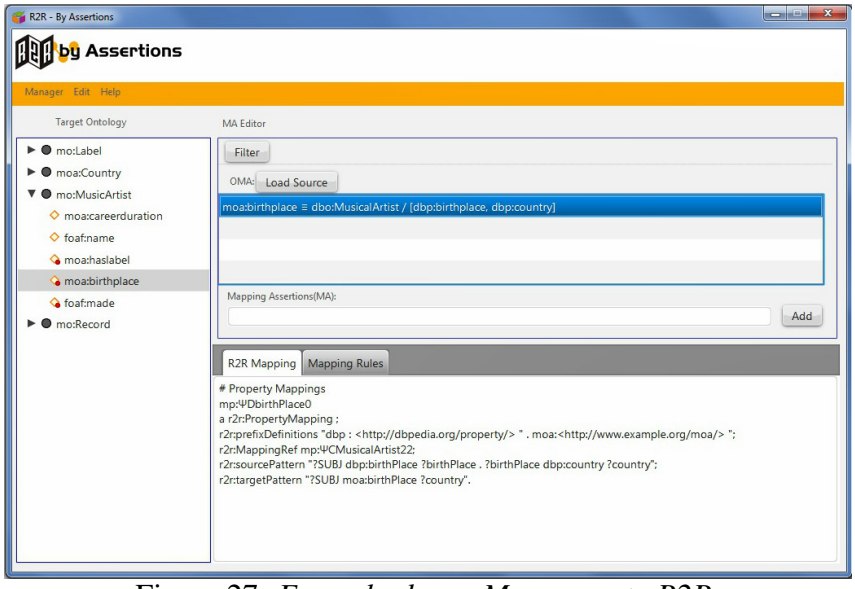



Figura 27: *Exemplo de um Mapeamento R2R*.

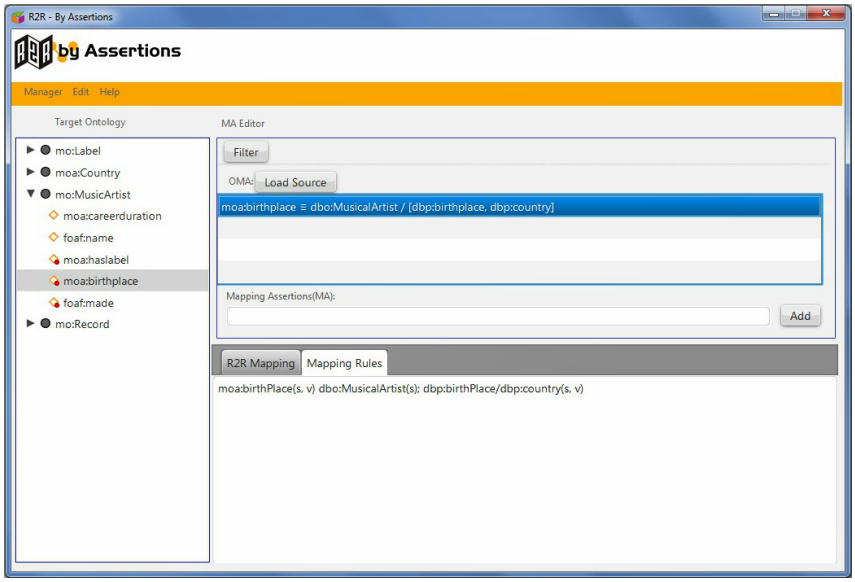



Figura 28: *Exemplo de uma Regra de mapeamento*.

### *5.3.3 Geração dos Mapeamentos R2R e Regras de Mapeamento*

A lista com todas as assertivas de mapeamento podem ser acessadas por meio do menu *Manager* localizado na parte superior da interface e, em seguida, no item *Assertions List*. A janela com essa lista é apresentada na Figura 29.

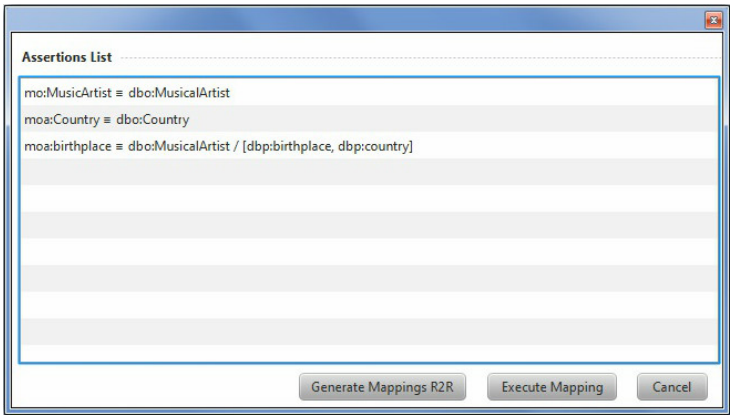



Figura 29: *Lista com todas as assertivas de mapeamento criadas*.

Para gerar um arquivo contendo todos os mapeamentos R2R com base nas assertivas definidas pelo usuário e obter os dados traduzidos para o vocabulário alvo. Devemos abrir a janela "Assertions List" (Menu Manager) e pressionar o botão *Generate Mapping R2R* e *Execute Mapping* também localizado na janela "Assertions List". Neste passo, o componente GR2R executa os Algoritmos 2, 3 e 4 mostrados na Seção 5.2, gerando o mapeamento mostrado na Figura 30. Este mapeamento é a saída final do processo da RBA.

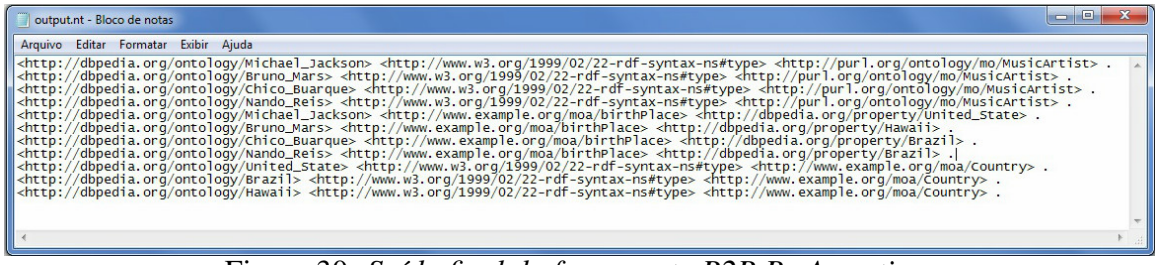



Figura 30: *Saída final da ferramenta R2R By Assertion.*

## 5.4 Discussão

Neste capítulo, apresentamos as principais funcionalidades da ferramenta *R2R By Assertion*, descrevendo sua arquitetura, bem como o funcionamento de cada um dos seus componentes. Introduzimos os algoritmos que viabilizaram a geração dos mapeamentos R2R de forma automática. Por último, Abordamos a aplicação da ferramenta RBA por meio do estudo de caso do Capítulo 3 que retrata um cenário musical. A ferramenta tem como função auxiliar na criação das assertivas de mapeamento entre duas ontologias e, a partir destas, gerar automaticamente os mapeamentos R2R. Como funcionalidade adicional, adicionamos a api do R2R na RBA de modo que podemos gerar as triplas R2R para o vocabulário alvo sem que seja necessário sair da RBA.

# 6 TRABALHOS RELACIONADOS

Uma das motivações para o estudo de ontologias é proporcionar uma visão homogênea sobre fontes de dados heterogêneas. No entanto, problemas relacionados a heterogeneidade de ontologias ainda persistem. Isso ocorre porque é possível encontrar diversas ontologias definidas para um mesmo domínio, cada qual com suas particularidades de modelagem. Essa diversidade torna-se um problema quando é necessário consultar dados armazenados em ontologias diferentes, mas que dizem respeito a um mesmo domínio. Este capítulo apresenta pesquisas relacionadas com alinhamento (*matching*) Seção 6.1 e mapeamento de ontologias Seção 6.2, que visam solucionar alguns problemas relacionados a heterogeneidade de ontologias. Também apresenta outros trabalhos que utilizam padrões de projeto como uma maneira eficiente de auxiliar na resolução desses problemas Seção 6.3.

## 6.1 Alinhamento de Ontologias

Alinhamento de ontologias (*ontology alignment*) é o processo de determinar correspondências entre termos (classes e propriedades) de duas ontologias. Usualmente os trabalhos nesta área propõem estratégias de modo que a geração dessas correspondências sejam semiautomáticas. Atualmente, existem muitas pesquisas relacionadas com alinhamento de ontologias (veja (SHVAIKO; EUZENAT, 2013) para um *survey*). Podemos citar a título de exemplo os trabalhos de (AUMUELLER *et al.*, 2005; LEME *et al.*, 2010; SELIGMAN *et al.*, 2010).

Em (AUMUELLER *et al.*, 2005), os autores utilizam a ferramenta **COMA++** para identificar quais itens são relevantes para realizar o alinhamento entre as ontologias. Esta tarefa é realizada combinando vários algoritmos, os quais por meio de iterações calculam a similaridade dos elementos (classes e propriedades) selecionados. A ferramenta proposta em (AUMUELLER *et al.*, 2005) oferece ainda uma infra-estrutura abrangente para resolver grandes problemas de alinhamentos do mundo real. **COMA++** tem como principais características: fácil utilização, personalizável para esquemas e instâncias de ontologias e excelente capacidade de avaliar diferentes algoritmos de alinhamento.

A estratégia adotada por (LEME *et al.*, 2010) propõe não só gerar os alinhamentos com base em funções de similaridade, mas também adota heurísticas para a realização das correspondências. (LEME *et al.*, 2010) propõe ainda técnicas de alinhamento híbridas baseadas em valores de instância e em informações do esquema, tais como tipos de dados, cardinalidade e

relações.

(SELIGMAN *et al.*, 2010) utiliza o alinhamento de ontologias para ajudar os usuários a descobrir e visualizar esquemas relevantes. Na abordagem apresentada por (SELIGMAN *et al.*, 2010) as correspondências relevantes são classificadas por meio de métricas, com base em técnicas de alinhamento de ontologias, levando sempre em consideração a qualidade das correspondências.

A nossa abordagem é diferente das estratégias apresentadas por (AUMUELLER *et al.*, 2005; LEME *et al.*, 2010; SELIGMAN *et al.*, 2010), uma vez que o foco da nossa abordagem está na transformação de instâncias e não na geração automática das correspondências entre elementos de ontologias diferentes. Embora também seja relevante determinar quais termos estão relacionados nos diferentes esquemas, nos preocupa fundamentalmente como estes termos estão relacionados. Assim, os trabalhos tais como os citados aqui, podem ser úteis numa primeira abordagem para encontrar correspondências entre as ontologias, mas precisam de mais informações para poderem ser usados no contexto do nosso trabalho.

## 6.2 Mapeamento de Ontologias

Mapeamentos são expressões que indicam como dados de um esquema podem ser transformados em dados descritos por meio de outro esquema. Diferente do alinhamento de ontologias, onde tem-se como resultado as duas ontologias separadas, mas com ligações entre seus termos equivalentes, no mapeamento de ontologias, tem-se como resultado uma única estrutura contendo expressões que fazem as ligações entre os termos de um modelo. Alguns trabalhos despontam no sentido de mapear ontologias entre eles está o (Colocar referências) *framework LDIF* (SCHULTZ *et al.*, 2012) destacado em (BIZER *et al.*, 2012) por apresentar um bom desempenho.

O *framework LDIF* define um *workflow* para integração de dados em *Linked Data*. Uma das etapas desse *workflow*, aborda a descoberta, a combinação e publicação dos mapeamentos de diferentes ontologias. O usuário tem que criar, de forma manual, declarações na linguagem de mapeamento R2R de modo a relacionar duas ontologias. A definição manual de mapeamentos R2R não é trivial e está sujeita a erros, principalmente quando mapeamentos complexos são necessários (Por exemplo mapear uma classe para uma propriedade). Além disso, a R2R é uma linguagem expressiva, mas em geral não é intuitiva para o usuário. Embora o LDIF seja o trabalho que mais está relacionado com o nosso, ele não trata do problema de criação dos

mapeamentos. Assim, a nossa abordagem pode ser vista como um complemento para o *LDIF*.

## 6.3 Padrões de Projeto em Ontologias

Padrões de Projeto em ontologias foram utilizados primeiramente na Engenharia de Ontologias, com o intuito de solucionar problemas tais como, relações inversas e relações compostas (BLOMQVIST; SANDKUHL, 2005). Existem poucos trabalhos de padrões de projeto na área de alinhamento de ontologias. Como exemplo podemos citar (SEQUEDA *et al.*, 2012), (SCHARFFE *et al.*, 2014), (SVÁB-ZAMAZAL *et al.*, 2009)). Até onde nós sabemos há apenas um trabalho na área de mapeamento de ontologias que baseiam-se em padrões de projeto.

Em (SEQUEDA *et al.*, 2012), são propostos padrões de projeto para reutilizar os mapeamentos R2RML mais comuns entre o modelo relacional e o RDF. A estrutura de seus padrões é muito semelhante a apresentada em nosso trabalho, sendo que a especificação do problema que cada padrão deve tratar é expressa por meio de uma pergunta ao invés de usar uma notação mais formal como a nossa. Por exemplo, o padrão *One to One Table Mapping* definido em (SEQUEDA *et al.*, 2012) trata do problema de *"Como várias tabelas podem ser mapeadas para uma classe?"*. Embora a pergunta seja intuitiva, ela não descreve detalhes da problemática envolvida.

(SVÁB-ZAMAZAL *et al.*, 2009) propõe um serviço de transformação de ontologias baseado em padrões. Neste trabalho, fragmentos de duas ontologias são alinhadas e transformadas em uma nova ontologia usando padrões de correspondência (também chamados padrões de transformação. O trabalho de (SVÁB-ZAMAZAL *et al.*, 2009) usa os padrões em um contexto diferente do nosso. No corrente trabalho, estamos interessados em transformar instâncias de uma ontologia fonte para instâncias de uma ontologia alvo. Os nossos padrões de mapeamento são utilizados para determinar como ocorre essa transformação.

(SCHARFFE *et al.*, 2014) introduzem padrões de alinhamento de ontologias como um meio para formalizar soluções para tratar incompatibilidades entre ontologias. Mais uma vez, este cenário (o mesmo apresentado, em (SVÁB-ZAMAZAL *et al.*, 2009)) é diferente do cenário apresentado neste documento. Em (SCHARFFE *et al.*, 2014), a relação de correspondências entre as entidades é, em geral, mais forte que a nossa, porque a relação de correspondência deve ser válida em ambas as direções do mapeamento (ou seja da fonte para o alvo e vice-versa). Em nossa abordagem, o mapeamento ocorre apenas em um sentido (a partir de uma ontologia fonte para uma ontologia alvo). Além disso, existem padrões em (SCHARFFE *et al.*, 2014) que não se

aplicam ao nosso enfoque desde que eles apenas indicam a existência de um relacionamento entre os termos mapeados, mas não contém semântica suficiente para ser utilizado na transformação dos dados de uma ontologia em outra. Por exemplo, padrões indicando que duas classes são disjuntas. Alguns dos padrões de mapeamento propostos no corrente trabalho são baseados nos padrões de alinhamento apresentados em (SCHARFFE *et al.*, 2014). Em adição, propomos novos padrões de mapeamentos como é o caso dos padrões de mapeamento para classes embutidas com filtros. A nossa abordagem também difere da apresentada por (SCHARFFE *et al.*, 2014) em dois aspectos: 1) usamos uma notação formal quando mostramos soluções para os problemas abordados pelos padrões; e 2) apresentamos, as restrições para utilizar o padrões nos casos onde elas existam.

A Tabela 7 apresenta a correspondência entre os padrões propostos por (SCHARFFE *et al.*, 2014) e os padrões apresentados nesta dissertação (Capítulo 4). A primeira coluna da tabela refere-se aos padrões de (SCHARFFE *et al.*, 2014). A segunda coluna contém o nome do padrão proposto no corrente trabalho, a terceira coluna contém um dos formalismos que usamos para representar o padrão.

Tabela 5 – Padrões de Alinhamento x Padrões de Mapeamento

| **Padrões de Alinhamento** | **Padrões de Mapeamento** | |
|---|---|---|
| **Nome** | **Nome** | **Formalismo** |
| - | Classe Não Semelhante Semanticamente | $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}$[P1,..., Pn]/**f** |
| Equivalente Class | Classe Semelhante Semanticamente | $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}/\mathbf{f}$ |
| Class Intersection | - | - |
| Class Union | Classe Semelhante Semanticamente (x2) | $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}1$, e $\psi$: $C_{\mathbf{T}} \equiv C_{\mathbf{S}}2$ |
| Class by Attribute Value | Propriedade de Tipo Dados 1-1 | $\psi$: $C_{\mathbf{T}}/P_{\mathbf{T}} \equiv C_{\mathbf{S}}/P_{\mathbf{S}}/\mathbf{f}/\mathrm{T}$ |
| SubClass Correspondence defined by relation domain | - | - |

O padrão de ***Classe Não Semelhante Semanticamente*** é utilizado para mapear instâncias de uma classe embutida $C_{\mathbf{S}}[A_1,\ldots,A_n]$ em $\mathbf{O_S}$ para instâncias de uma classe $C_{\mathbf{T}}$ em $\mathbf{O_T}$ (ver definição de classe embutida no Capítulo 4). Os padrões de projeto desenvolvidos por (SCHARFFE *et al.*, 2014) não trata esse problema. O padrão de ***Classe Semelhante Semanticamente*** é similar ao padrão *Equivalente class*. A diferença em que o nosso padrão determina que o conjunto de indivíduos de $C_{\mathbf{S}}$ devem está em $C_{\mathbf{T}}$ (entretanto, $C_{\mathbf{T}}$ pode ter outros

elementos para além dos de $C_{\mathbf{S}}$), enquanto o padrão *Equivalente Class* determina que o conjunto de indivíduos em $C_{\mathbf{T}}$ e $C_{\mathbf{S}}$ é exatamente o mesmo.

No padrão *Class Intersection*, uma classe em uma ontologia alvo é obtida a partir da interseção de duas classes na ontologia fonte. Atualmente as AM apresentadas neste trabalho não tratam esta situação. Poderíamos tratar esse caso usando filtro, mas para isso teríamos que estender a definição de filtro para conter propriedades de classes diferentes. No padrão *Class Union* ocorre a união de duas classes numa mesma ontologia fonte conseguimos lidar com essa situação ao mapear duas classes distintas na ontologia fonte ($C_{\mathbf{S}}1$, e $C_{\mathbf{S}}2$) em uma classe $C_{\mathbf{T}}$ na ontologia alvo. Entretanto, nesse casos instâncias duplicadas eventualmente podem aparecer em $C_{\mathbf{T}}$.

No padrão *Class by Attribute Value* uma instância em $C_{\mathbf{T}}$ equivale exatamente a outra instância em $C_{\mathbf{S}}$. Esse caso é tratado neste trabalho sendo ao similar ao padrão *Propriedade de Tipo de Dados 1-1*, ainda para este caso é opcional acrescentar uma seleção (filtro) sobre $C_{\mathbf{S}}$. No padrão *SubClass Correspondence Defined by Relation Domain* uma classe em uma ontologia é equivalente à subclasse de uma classe em uma segunda. Não temos um padrão semelhante para esse caso.

Quanto a proposta apresentada em (BIZER *et al.*, 2012) trata-se de um *benchmarking* para avaliação de tecnologias que realizam o mapeamento entre ontologias. Para a execução do *benchmarking*, os autores construíram manualmente mapeamentos utilizando *datasets* existentes na LOD[1], por eles denominados de "mapeamentos executáveis". A partir desses mapeamentos, os autores observaram a existência de problemas que aparecem com frequência no mundo de dados ligados. Como consequência dessa observação, os autores propuseram padrões de mapeamentos para os problemas observados. Para uma visão mais ampla dos padrões RBA em relação aos padrões apresentados no trabalho de Scharffe, o leitor pode dirigir-se ao *apêndice A* deste trabalho.

O trabalho em (BIZER *et al.*, 2012) assemelha-se ao nosso no que diz respeito à utilização de padrões de mapeamento. Entretanto, em (BIZER *et al.*, 2012) os padrões consistem apenas na definição dos problemas de mapeamento usuais entre ontologias da LOD. Uma vez que o foco não é na transformação dos dados entre ontologias, não são apresentados soluções para esses problemas. Além disso, em (BIZER *et al.*, 2012) os mapeamentos são gerados manualmente. A nossa proposta visa criar mapeamentos automaticamente, entre outros motivos

[1] https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

Tabela 6 – Padrões de Mapeamento RBA x Padrões de Mapeamento Bizer

| **Padrão de Mapeamento RBA** | **Padrão de Mapeamento Bizer (BIZER *et al.*, 2012)** |
|---|---|
| Classes Semelhantes Semanticamente | X |
| Classes Não Semelhantes Semanticamente | X |
| – | Rename Class based on Property (RCP) |
| Propriedade de Objeto Direto de Classes Semelhantes Semanticamente | X |
| Propriedade de Objeto de Caminho de Classes Semelhantes Semanticamente | X |
| Propriedade de Objeto de Caminho de Classes Não Semelhantes Semanticamente | – |
| – | Resourcesify (Rsc) |
| Propriedades de Tipos de Dados Direto 1-1 de Classes Semelhantes Semanticamente | X |
| Propriedades de Tipos de Dados Direto n-1 de Classes Semelhantes Semanticamente | X |
| Propriedades de Tipos de Dados de Caminho | – |
| – | Add Language Tag (ALT) |
| – | Remove Language Tag (RLT) |
| – | Aggregate (Agg) |
| – | URI to Value (UtV) |

porque quando escrevemos os mapeamentos à mão estamos sujeitos a diversos tipos de erros tais como (erros de escrita ou até mesmo erros semânticos). Na nossa proposta desenvolvemos uma ferramenta para auxiliar a construção dos mapeamentos de forma semiautomática. Para isso, utilizamos assertivas de mapeamento, as quais fazem parte dos padrões de mapeamento.

Apesar da diferença de foco entre os dois trabalhos, os padrões identificados em (BIZER *et al.*, 2012) podem dar-nos uma certeza maior de que os padrões que definimos cobrem os problemas usuais encontrados na LOD. Assim, após a análise dos padrões introduzidos em (BIZER *et al.*, 2012), verificamos que alguns problemas identificados em (BIZER *et al.*, 2012) não são tratados por nossos padrões. É o caso do padrão *Agregate* o qual descreve o problema da contagem do número de instâncias de propriedades entre as ontologias alvo e fonte.

A Tabela 6 apresenta a correspondência entre os padrões apresentados nesta dissertação (Capítulo 4) e os padrões propostos por (BIZER *et al.*, 2012). A primeira coluna da tabela refere-se aos padrões RBA e a segunda apresenta os padrões propostos por (BIZER *et al.*, 2012). As linhas marcadas com a letra "X" significa que, para aquele padrão RBA existe um padrão similar proposto por (BIZER *et al.*, 2012), caso contrário, a linha é marcada com "–" indicando que para o padrão RBA não existe nenhum padrão similar.

## 6.4 Discussão

Este capítulo apresentou alguns trabalhos que tratam de problemas relacionados com a heterogeneidade entre ontologias, mais especificamente aqueles relacionados com alinhamento e mapeamento de ontologias. Os trabalhos destacados neste capítulo estão de alguma forma relacionados com o desenvolvimento de formalismos para representar correspondência. Além disso eles propõem utilizar padrões como uma forma de auxiliar na construção, seja de mapeamentos ou no processo de alinhamento de ontologias. Com este estudo, foram obtidos subsídios para formular contribuições relevantes no contexto de mapeamentos entre ontologias.

# 7 CONCLUSÃO E TRABALHOS FUTUROS

## 7.1 Considerações Finais

Motivados pela necessidade de facilitar a geração de mapeamentos R2R, foi proposto uma abordagem com base em padrões de mapeamento para gerar de forma semiautomática mapeamentos R2R. A estratégia utiliza Assertivas de Mapeamento como um formalismo mais conciso e de mais alto nível para especificar mapeamentos entre vocabulários RDF. Ressaltamos que, nessa abordagem, as AM podem auxiliar na especificação tanto mapeamentos mais simples, tais como: de classes semelhantes semanticamente e propriedades de tipo de dados diretas, quanto mapeamentos complexos tais como de classes não semelhantes semanticamente.

A estratégia de usar Assertivas foi introduzida em trabalhos anteriores como em (VIDAL *et al.*, 2005) para investigar visões XML e em (PEQUENO *et al.*, 2014) para facilitar geração de mapeamentos entre bancos de dados relacionais e RDF. Portanto, seria natural adotar a mesma abordagem para o problema de criar mapeamentos entre ontologias.

Usando AM, as quais fazem parte da solução para os problemas de mapeamento definidos nos padrões propostos no Capitulo 4, é mostrado como os mapeamentos R2R podem ser gerados de maneira semiautomática para traduzir o vocabulário de uma ontologia fonte para o vocabulário de uma ontologia de alvo.

Para dar suporte à abordagem proposta, foi desenvolvida uma ferramenta gráfica denominada **R2R By Assertions**, apresentada com detalhes no Capítulo 5. O funcionamento dessa ferramenta foi demonstrado passo a passo com a aplicação do estudo de caso introduzido no Capítulo 3.

Uma versão preliminar da proposta discutida nesta dissertação foi publicada em uma conferência de âmbito nacional: *Automatic Generation of R2R Mappings from Correspondence Assertions* (SBBD, 2015). No entanto, espera-se publicações mais aprofundadas sobre a parte teórica e sobre a ferramenta *R2R By Assertions* em conferências e jornais importantes e de âmbito internacional.

## 7.2 Trabalhos Futuros

Muitas ideias surgiram durante o desenvolvimento deste trabalho. Algumas foram deixadas de lado por não terem se mostrado viáveis. Outras foram postas em prática e fazem

parte desta dissertação. No entanto, ainda existem muitas possibilidades para dar continuidade ao trabalho realizado, elencadas a seguir:

- Implementação de todos os padrões de mapeamento apresentados na biblioteca de padrões mostrado no Capítulo 4 na ferramenta RBA, tais como as Padrão de Mapeamento de Propriedade de Tipo de Dados Direto 1-1 de Classes Não Semelhantes Semanticamente e Padrão de Mapeamento de Propriedade de Objeto Direto de Classes Não Semelhantes Semanticamente.
- Estender a ferramenta RBA com novas funcionalidades, tais como: (i) permitir a edição e remoção de AMs, e (ii) permitir outros tipos de serialização de ontologias tais como N-TRIPLES, RDF/XML.
- Embora a ferramenta *R2R-By Assertions* contemple a maioria dos casos de mapeamento tais como: o mapeamento de classe e propriedades, algumas possibilidades de implementação como filtros e funções de transformação ainda precisão ser colocadas em prática. Desta forma, essa tarefa fica como uma extensão do nosso trabalho.
- Por último, iremos estender nossa abordagem para incluir os padrões **Deresourcesify (DRsc)**, **Add Language Tag (ALT)**,**Remove Language Tag (RLT)** e **Change Datatype (CD)** apresentados em (BIZER *et al.*, 2012), uma vez que a abordagem proposta nesse trabalho não cobre estes casos.

# REFERÊNCIAS


ALMEIDA, M. B.; BAX, M. P. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. **Ciência da Informação, Brasília**, SciELO Brasil, v. 32, n. 3, p. 7–20, 2003.

AUMUELLER, D.; DO, H.-H.; MASSMANN, S.; RAHM, E. Schema and ontology matching with coma++. In: ACM. **Proceedings of the 2005 ACM SIGMOD international conference on Management of data**. [S.l.], 2005. p. 906–908.

BAADER, F.; HORROCKS, I.; SATTLER, U. Description logics as ontology languages for the semantic web. In: **Mechanizing Mathematical Reasoning**. [S.l.]: Springer, 2005. p. 228–248.

BECHHOFER, S. Owl: Web ontology language. In: **Encyclopedia of Database Systems**. [S.l.]: Springer, 2009. p. 2008–2009.

BERNERS-LEE, T. **Linked Data - Design Issues**. 2005. <https://www.w3.org/2005/Talks/0517-boit-tbl/>.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. *et al.* The semantic web. **Scientific american**, New York, NY, USA:, v. 284, n. 5, p. 28–37, 2001.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked Data - The Story So Far. **Int. J. Semantic Web Inf. Syst.**, v. 5, n. 3, p. 1–22, 2009.

BIZER, C.; JENTZSCH, A.; CYGANIAK, R. **State of the LOD Cloud**. 2011. <http://www4.wiwiss.fu-berlin.de/lodcloud/state/>.

BIZER, C.; SCHULTZ, A. The r2r framework: Publishing and discovering mappings on the web. In: HARTIG, O.; HARTH, A.; SEQUEDA, J. (Ed.). **COLD**. [S.l.]: CEUR-WS.org, 2010. (CEUR Workshop Proceedings, v. 665).

BIZER, C.; SCHULTZ, A. The r2r framework: Publishing and discovering mappings on the web. In: CEUR-WS. ORG. **Proceedings of the First International Conference on Consuming Linked Data-Volume 665**. [S.l.], 2010. p. 97–108.

BIZER, C.; SCHULTZ, A.; RUIZ, D.; RIVERO, C. R. Benchmarking the performance of linked data translation systems. 2012.

BLOMQVIST, E.; SANDKUHL, K. Patterns in ontology engineering: Classification of ontology patterns. In: CITESEER. **ICEIS (3)**. [S.l.], 2005. p. 413–416.

BREITMAN, K. K.; CASANOVA, M. A. Desenvolvimento de ontologias para engenharia de software e banco de dados: um tutorial prático. **Tutorial Apresentado no XIX Simpósio Brasileiro de Engenharia de Software. Uberlândia–MG**, 2005.

BRICKLEY, D.; GUHA, R. V. {RDF vocabulary description language 1.0: RDF schema}. 2004.

CUNHA, L. M. S. Web semântica: estudo preliminar. **Embrapa Informática Agropecuária. Documentos**, Campinas: Embrapa Informática Agropecuária., 2002.

DACONTA, M. C.; OBRST, L. J.; SMITH, K. T. **The semantic web: a guide to the future of XML, web services, and knowledge management**. [S.l.]: John Wiley & Sons, 2003.

FENSEL, D. **Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, SpringerVerlag Berlin and Heidelberg GmbH & Co**. [S.l.]: KG, 2001.

GRUBER, T. What is an ontology. **WWW Site http://www-ksl. stanford. edu/kst/whatis-an-ontology. html (accessed on 07-09-2004)**, 1993.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing? **International journal of human-computer studies**, Elsevier, v. 43, n. 5, p. 907–928, 1995.

GUARINO, N. Formal ontology, conceptual analysis and knowledge representation. **International journal of human-computer studies**, Elsevier, v. 43, n. 5, p. 625–640, 1995.

GUARINO, N. Formal ontology and information systems. In: **Proceedings of FOIS**. [S.l.: s.n.], 1998. v. 98, n. 1998, p. 81–97.

JACOB, E. K. Ontologies and the semantic web. **Bulletin of the American Society for Information Science and Technology**, Wiley Online Library, v. 29, n. 4, p. 19–22, 2003.

JUN, T.; TAN, S.; ZHENG, L. Linked dataset mapping language: R2r. **Journal of Library Science in China**, v. 3, p. 013, 2012.

KLEIN, M. Combining and relating ontologies: an analysis of problems and solutions. In: **IJCAI-2001 Workshop on Ontologies and Information Sharing**. Seattle, WA: [s.n.], 2001. p. 53–62.

KLYNE, G.; CARROLL, J. J. Resource description framework (rdf): Concepts and abstract syntax. w3c recommendation, 2004. **World Wide Web Consortium, http://w3c. org/TR/rdf-concepts**, 2004.

LEME, L. A. P. P.; CASANOVA, M. A.; BREITMAN, K. K.; FURTADO, A. L. Owl schema matching. **Journal of the Brazilian Computer Society**, Springer, v. 16, n. 1, p. 21–34, 2010.

LIMA, J. C. de; CARVALHO, C. L. de. **Resource description framework (rdf)**. [S.l.], 2005.

LOPES, F. L. **Acesso a Dados a partir de Ontologias utilizando Mapeamentos Heterogêneos e Programação em Lógica**. Tese (Doutorado) — MSc. Thesis, UFC, Fortaleza, Brazil, 2010.

Lutz, C. Description logics with concrete domains—a survey. In: **Advances in Modal Logic 2002 (AiML 2002)**. Toulouse, France: [s.n.], 2002.

MAGALHÃES, R. **Um Ambiente para Processamento de Consultas Federadas em Linked Data Mashups**. Tese (Doutorado) — MS thesis, Universidade Federal do Ceará, 2012.

MENDONÇA, R. R. de. **Uma abordagem para coleta e publicação de dados de proveniência no contexto de Linked Data**. Dissertação (Mestrado em Informática) — Programa de Pós-Graduação em Informática, Instituto de Matemática, Instituto Tércio Pacitti, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Dezembro 2013.

NOY, N. F.; MCGUINNESS, D. L. *et al.* **Ontology development 101: A guide to creating your first ontology**. [S.l.]: Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, 2001.

PEQUENO, V. M.; VIDAL, V. M.; CASANOVA, M. A.; NETO, L. E. T.; GALHARDAS, H. Specifying complex correspondences between relational schemas and rdf models for generating customized r2rml mappings. In: ACM. **Proceedings of the 18th International Database Engineering & Applications Symposium**. [S.l.], 2014. p. 96–104.

PINHEIRO, J. C. **Processamento de consulta em um framework baseado em um mediador para integração de dados no padrão de Linked Data**. Dissertação (Mestrado em Ciência da Computação) — Departamento de Ciência da Computação, Mestrado e Doutorado em Ciência da Computação, Universidade Federal do Ceará, Fortaleza/CE, Setembro 2011.

PRUD'HOMMEAUX, E.; SEABORNE, A. *et al.* Sparql query language for rdf. **W3C recommendation**, v. 15, 2008.

RIVERO, C. R.; HERNÁNDEZ, I.; RUIZ, D.; CORCHUELO, R. Mosto: Generating sparql executable mappings between ontologies. In: SPRINGER. **International Conference on Conceptual Modeling**. [S.l.], 2011. p. 345–348.

Scharffe, F.; Zamazal, O.; Fensel, D. Ontology alignment design patterns. **Knowl. Inf. Syst.**, v. 40, n. 1, p. 1–28, jul. 2014.

SCHARFFE, F.; ZAMAZAL, O.; FENSEL, D. Ontology alignment design patterns. **Knowledge and Information Systems**, Springer, v. 40, n. 1, p. 1–28, 2014.

SCHREIBER, G.; RAIMOND, Y. **RDF 1.1 Primer-W3C Working Group Note 24 june 2014, W3C Working Group Note**. 2015.

SCHULTZ, A.; MATTEINI, A.; ISELE, R.; MENDES, P. N.; BIZER, C.; BECKER, C. Ldif-a framework for large-scale linked data integration. In: **21st International World Wide Web Conference (WWW 2012), Developers Track, Lyon, France**. [S.l.: s.n.], 2012.

SELIGMAN, L.; MORK, P.; HALEVY, A.; SMITH, K.; CAREY, M. J.; CHEN, K.; WOLF, C.; MADHAVAN, J.; KANNAN, A.; BURDICK, D. Openii: an open source information integration toolkit. In: ACM. **Proceedings of the 2010 ACM SIGMOD International Conference on Management of data**. [S.l.], 2010. p. 1057–1060.

SEQUEDA, J.; PRIYATNA, F.; VILLAZÓN-TERRAZAS, B. Relational database to rdf mapping patterns. In: CEUR-WS. ORG. **Proceedings of the 3rd International Conference on Ontology Patterns-Volume 929**. [S.l.], 2012. p. 97–108.

SHVAIKO, P.; EUZENAT, J. Ontology matching: state of the art and future challenges. **IEEE Transactions on knowledge and data engineering**, IEEE, v. 25, n. 1, p. 158–176, 2013.

SVÁB-ZAMAZAL, O.; SVÁTEK, V.; SCHARFFE, F. Pattern-based ontology transformation service. In: NO COMMERCIAL EDITOR. **Proc. 1st IK3C international conference on knowledge engineering and ontology development (KEOD)**. [S.l.], 2009. p. 210–223.

VIDAL, V.; ARAUJO, V.; CASANOVA, M. Towards automatic generation of rules for incremental maintenance of xml views of relational data. In: SPRINGER. **International Conference on Web Information Systems Engineering**. [S.l.], 2005. p. 189–202.

VIDAL, V. M. P.; CASANOVA, M. A.; CARDOSO, D. S. Incremental maintenance of RDF views of relational data. In: **Proc. The 12th International Conference on Ontologies, DataBases, and Applications of Semantics**. Graz, Austria: [s.n.], 2013.

VIDAL, V. M. P.; LÓSCIO, B. F.; SALGADO, A. C. Using correspondence assertions for specifying the semantics of XML-based mediators. In: **Workshop on Information Integration on the Web**. [S.l.: s.n.], 2001. p. 3–11.

WELTY, C.; MCGUINNESS, D. L.; SMITH, M. K. Owl web ontology language guide. **W3C recommendation, W3C (February 2004) http://www. w3. org/TR/2004/REC-owl-guide-20040210**, 2004.

YU, C.; POPA, L. Constraint-based xml query rewriting for data integration. In: ACM. **Proceedings of the 2004 ACM SIGMOD international conference on Management of data**. [S.l.], 2004. p. 371–382.

# APÊNDICE A – PADRÕES RBA X PADRÕES DE PROJETO SCHARFFE

Tabela 7 – tabela1

| **Padrão de Mapeamento RBA** | **Padrão de Mapeamento Bizer (SCHARFFE *et al.*, 2014)** |
|---|---|
| Classes Semelhantes Semanticamente | X |
| Classes Não Semelhantes Semanticamente | – |
| Propriedade de Objeto Direto de Classes Semelhantes Semanticamente | X |
| Propriedade de Objeto Direto de Classes Não Semelhantes Semanticamente | – |
| Propriedade de Objeto de Caminho de Classes Semelhantes Semanticamente | X |
| Propriedade de Objeto de Caminho de ClassesNão Semelhantes Semanticamente | – |
| Propriedades de Tipos de Dados Direto 1-1 de Classes Semelhantes Semanticamente | X |
| Propriedades de Tipos de Dados Direto 1-1 de Classes Semelhantes Semanticamente | X |
| Propriedades de Tipos de Dados Direto n-1 de Classes Semelhantes Semanticamente | X |
| Propriedades de Tipos de Dados de Caminho | – |
| – | Subclass Correspondence |
| – | Class Intersection Correspondence |
| Classes Semelhantes Semanticamente (x2) | Class Union Correspondence |
| – | Disjoint Class Correspondence |
| – | Class By Property Value Correspondence |
| – | Class Correspondence by Path Attribute Value |
| – | Class By Attribute Occurence Correspondence |
| | Class Correspondence by Path Attribute Occurence |
| – | Class By Attribute Type Correspondence |
| – | Class Relation Correspondence |
| – | Class Instance Correspondence |
| – | Subclass By Domain Correspondence |
| – | Subclass Outside Domain Correspondence |
| – | Subclass By Range Correspondence |
| – | Subclass Outside Range Correspondence |
| – | Attribute domain restriction Correspondence |
| – | Property - Relation Correspondence |
| – | Disjoint Attributes Correspondence |
| – | Equivalent Property Correspondence |
| – | Subproperty Correspondence |

Tabela 8 – Continuação da tabela anterior

| **Padrão de Mapeamento RBA** | **Padrão de Mapeamento Bizer (SCHARFFE *et al.*, 2014)** |
|---|---|
| – | Property Correspondence by Value |
| – | Property Value Transformation |
| – | Equivalent Relation Correspondence |
| – | Subrelation Correspondence Negated Relation Correspondence |
| – | Relation Transitive Closure Correspondence |
| – | Inverse Relation Correspondence |
| – | Relation Symmetric Closure Correspondence |
| – | Inverse Subrelation Correspondence |
| – | Relation by Domain Restriction |
| – | Relation by Range Restriction |
| – | Equivalent Individual Correspondence |
| – | Disjoint Instances Correspondence |
| – | Subclass Correspondence By Attribute Value Intersection |
| – | Subclass By Attribute Disjunction Correspondence |
| – | Inverse Relation Transitive Closure Correspondence |
| – | Class By Attribute Range Partition |
| – | Multiple Classes By Attribute Range Partition |
| – | Aggregation Pattern |
| – | Property Partition by Value Pattern |
| – | RCA-A Concat Pattern |