**UNIVERSIDADE FEDERAL DO CEARÁ**

**CENTRO DE CIÊNCIAS**

**DEPARTAMENTO DE COMPUTAÇÃO**

**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**NARCISO MOURA ARRUDA JÚNIOR**

# MANUTENÇÃO INCREMENTAL DE VISÕES DE *MASHUP* DE DADOS INTERLIGADOS

**FORTALEZA**

**2015**

NARCISO MOURA ARRUDA JÚNIOR

# MANUTENÇÃO INCREMENTAL DE VISÕES DE *MASHUP* DE DADOS INTERLIGADOS

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação do Departamento de Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação. Área de Concentração: Banco de Dados.

Orientador: Prof. Drª. Vânia Maria Ponte Vidal

FORTALEZA

2015

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)

---

A819m Arruda Júnior, Narciso Moura.
Manutenção Incremental de Visões de Mashup de Dados Interligados / Narciso Moura Arruda Júnior. – 2015.
63 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2015.
Orientação: Profa. Dra. Vânia Maria Ponte Vidal.

1. Aplicação de Mashup de Dados. 2. Interligação de Datasets RDF. 3. Dados Interligados. 4. Manutenção de Visão Mashup. I. Título.

CDD 005

---

NARCISO MOURA ARRUDA JÚNIOR

# MANUTENÇÃO INCREMENTAL DE VISÕES DE *MASHUP* DE DADOS INTERLIGADOS

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação do Departamento de Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação. Área de Concentração: Banco de Dados.

Aprovada em ____/_____/_____

BANCA EXAMINADORA

_____

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

_____

Prof. Dr. Marco Antonio Casanova
Pontifícia Universidade Católica do Rio de Janeiro (PUC RIO)

_____

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

_____

Prof. Dr. Javam de Castro Machado
Universidade Federal do Ceará (UFC)

Dedico este trabalho a minha família, que foi meu porto seguro perante as dificuldades deste percurso.

## AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer, a Deus, pois sem Ele nada seria possível, pois eles são os pilares que nos sustentam. E, por todas as oportunidades dadas.

Agradeço á minha mãe Célia Rodrigues, minha avó Maria de Lourdes e minhas irmãs Ivelise e Iveline, que estiveram sempre me apoiando e me dando forças para que eu continuasse na luta durante essa etapa da minha vida.

Muito obrigado também a minha namorada, Beatriz Freitas, que compartilhou comigo esse momento, foi paciente em minhas ausências e me deu apoio nesse trabalho.

Agradeço a minha orientadora Vânia Vidal e meu coorientador Antonio Casanova por terem me orientado, e me dando todo suporte necessário para a conclusão desse trabalho, e de forma brilhante terem contribuído para o meu desenvolvimento.

Ao professor Javam de Castro Machado, por ter aceitado o convite para participar da banca examinadora deste trabalho.

Agradecimento à pesquisadora Valeria Magalhães Pequeno pela ajuda com os mapeamentos R2R.

Agradecimento aos amigos Regis Pires Magalhães pela ajuda na escrita, ao Roberval Mariano pelas discussões sobre visões de *Mashup* e fusão de dados, ao Diego Sá Cardoso pelos esclarecimentos sobre manutenção incremental e ao Tiago Vinuto pela ajuda com os mapeamentos R2R.

Agradeço a todos que compõem o MDCC, em especial aos professores José Maria, José Antônio Macedo e João Paulo, que sempre estiveram dispostos a ajudar, contribuindo com o meu trabalho e desenvolvimento pessoal. Os meus amigos Julian Rodrigues, Carleandro Noleto, Claudio Carvalho, Thiago Alves, Andressa Bezerra e Samara Martins, que estiveram juntos comigo nessa batalha, compartilhando seus conhecimentos e experiência, tornando essa caminha mais agradável.

Agradecimento aos amigos do DI, da PUC Rio, Giseli Lopes, Alexander Mera, José Eduardo, Percy Salas e Elisa Menendez, que foram essenciais na busca de conhecimento, e que me apoiaram e me ajudaram nas minhas pesquisas.

A todos que não foram citados, mas que contribuíram para esse trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio financeiro que possibilitou a realização deste trabalho.

*“A essência do conhecimento consiste em aplicá-lo, uma vez possuído.”* (Confúcio)

## RESUMO

A iniciativa de Dados Interligados estimula a publicação de bases de dados previamente isoladas como um conjunto de dados RDF interligados, criando assim um espaço de dados de escala global, conhecido como a Web de Dados. Aplicações de *Mashup* de Dados Interligados, que consomem dados de múltiplas fontes de dados interligados na Web de Dados, são confrontadas com o desafio de obter uma visão homogeneizada deste espaço global de dados, chamada de visão de *Mashup* de Dados Interligados. Este trabalho propõe um framework baseado em ontologia para especificar formalmente visões de *Mashup* de Dados Interligados, e uma estratégia para manutenção incremental dessas visões, baseada em suas especificações.



**Palavras-chave**: Aplicação de *Mashup* de Dados. Interligação de Datasets RDF. Dados Interligados. Manutenção de Visão *Mashup*.

## ABSTRACT

The Linked Data initiative promotes the publication of previously isolated databases as interlinked RDF datasets, thereby creating a global scale data space, known as the Web of Data. Linked Data Mashup applications, which consume data from the multiple Linked Data sources in the Web of Data, are confronted with the challenge of obtaining a homogenized view of this global data space, called a Linked Data Mashup view. This work proposes an ontology-based framework for specifying formally Linked Data Mashup views, and a strategy for incremental maintenance of such views, based on their specifications.



**Keywords**: Data Mashup application. RDF dataset interlinking. Linked Data. View Maintenance.

## LISTA DE FIGURAS

## LISTA DE TABELAS

**SUMÁRIO**

# 1 INTRODUÇÃO

## 1.1 Motivação

A iniciativa de *Dados Interligados* (BERNERS-LEE, 2006) trouxe novas oportunidades para construir a próxima geração de aplicações de *Mashup* com Semântica (ENDRES, 2013). Ao expor conjuntos de dados anteriormente isolados na forma de um grafo de dados, que podem ser interligados e integrados com outros conjuntos de dados. *Dados Interligados* permitem a criação de um espaço de dados interligados de escala global, conhecida como a Web de Dados. O sucesso da iniciativa de Dados Interligados deve-se principalmente à adoção de padrões conhecidos da Web, tais como padrões de infraestrutura da Web (*URIs* e *HTTP*), padrões da Web Semântica (*RDF* e *RDFs*) e vocabulários. Graças a isso, uma quantidade sem precedente de Fontes de Dados Interligados foi recentemente produzidas e continua crescendo rapidamente.

Um *mashup* de dados interligados é uma aplicação (web) que oferece novas funcionalidades combinando, agregando, e transformando dados disponíveis na web de dados (HANH, TAI, *et al.*, 2014), (SCHULTZ, MATTEINI, *et al.*, 2012). Alimentado por ferramentas e tecnologias tendo sido desenvolvidas pela comunidade de Web Semântica, existem várias aplicações de domínio construídas de aplicações com *mashup* de Dados Interligados (HANH, TAI, *et al.*, 2014). Um simples exemplo de *Mashup* de Dados Interligados é *BBC Music* (KOBILAROV, SCOTT, *et al.*, 2009) que integra dados de duas fontes de Dados Interligados, *DBpedia* (BIZER, LEHMANN, *et al.*, 2009) e *MusicBrainz* (SWARTZ, 2002).

## 1.2 Caracterização do Problema

Aplicações de *Mashup* de Dados Interligados (Aplicações MDI) são confrontadas com o desafio de obter uma visão homogeneizada desse espaço de dados global. Chamamos essa visão de visão de *Mashup* de Dados Interligados (visão *MDI*). A criação de uma visão *MDI* é uma tarefa complexa que envolve quatro grandes desafios:

1) seleção das fontes de Dados Interligados que são relevantes para a aplicação;
2) extração e tradução de dados diferentes, possivelmente de fontes de Dados Interligados heterogêneas para um vocabulário comum;
3) identificação de ligações entre recursos em diferentes fontes de Dados Interligados;
4) combinação e fusão de múltiplas representações do mesmo objeto no mundo real para uma única representação e resolução de inconsistências para melhorar a qualidade dos dados.

Para ser útil, uma visão MDI deve ser atualizada continuamente para refletir dinamicamente as atualizações das fontes de dados. Basicamente, existem duas estratégias para manutenção de uma visão materializada: a *Re-materialização*, que recalcula os dados da visão em tempos pré-estabelecidos, e a *manutenção incremental* que periodicamente modifica parte dos dados da visão para refletir atualizações do banco de dados. Tem sido demonstrado que a manutenção incremental geralmente supera o recalculo da visão completa (ABITEBOUL, MCHUGH, *et al.*, 1998), (DIMITROVA, EL-SAYED e RUNDENSTEINER, 2003), (GRIFFIN e LIBKIN, 1998), (KUNO e RUNDENSTEINER, 1998).

## 1.3 Contribuições

Neste trabalho, investigamos o problema da manutenção incremental de visões MDI. Em primeiro lugar, propomos um framework baseado em ontologia para especificar formalmente visões MDI. Nesse framework, uma visão MDI é especificada com a ajuda de visões exportadas, visões de conjunto de ligações *sameAs*, métricas de avaliação de qualidade, regras de fusão de dados e função de normalização. A especificação da visão MDI é usada para materializar automaticamente a visão de *mashup*. Então, propomos uma estratégia que usa a especificação da visão MDI para manter incrementalmente a materialização da visão de *mashup*.

A estratégia de manutenção incremental de visões MDI é a maior contribuição deste trabalho. A estratégia é endereçada a problemas de lidar com a combinação e fusão de múltiplas representações do mesmo objeto do mundo real onde as fontes de Dados Interligados são atualizadas. Como será discutido no capítulo de trabalhos relacionados, este

problema recebeu pouca atenção e ainda coloca novos desafios devido à própria natureza dos dados do *mashup*.

## 1.4 Organização da Dissertação

Esta dissertação possui seis capítulos. O Capítulo 1 é esse capítulo introdutório, os demais são:

Capítulo 2 – Fundamentação Teórica – contém os resumos dos assuntos mais relevantes para nosso trabalho e que serão necessários para o entendimento da dissertação.

Capítulo 3 – Trabalhos Relacionados – apresenta os principais trabalhos que abordam o tema criação de *Mashup* e Manutenção incremental.

Capítulo 4 – Framework Baseado em Ontologia para Especificar visões MDI – descreve o estudo de caso que será usado ao longo da dissertação. Propõe um Framework para criação de *mashup* materializados. Nesse capítulo, também serão definidos os requisitos necessários para materialização das visões exportadas, conjunto de ligações *sameAs*, e por fim a materialização da visão de *mashup* MDI.

Capítulo 5 – Manutenção Incremental de Visões MDI – descreve o procedimento *Effect*, utilizado para manter atualizada a visão MDI. Apresenta o Teorema 5.1 e o Teorema 5.2, para demostrar que a atualização é feita corretamente.

E por fim, o Capítulo 5 – Conclusão – apresenta possíveis trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

## 2.1 Introdução

Neste capítulo, apresentamos os conceitos básicos utilizados nessa dissertação. Entre os assuntos abordados estão Dados Interligados e *mashup* de Dados Interligados.

## 2.2 *Linked Data* (Dados Interligados)

A Web atual deixou de ser apenas um espaço global de documentos interligados e está se tornando um enorme espaço global de dados vinculados que cobrem os mais variados domínios (HEATH e BIZER, 2011). Esta nova Web, denominada Web de Dados, visa preparar o caminho para a Web Semântica funcional, onde haverá a disponibilidade de uma grande quantidade de dados interligados em formato RDF (MAGALHÃES, 2012). O conjunto com as melhores práticas para publicação de dados e ligação de dados estruturados na Web é conhecido como Dados Interligados. A adoção das melhores práticas de Dados Interligados levou a Web a se tornar um espaço global de dados ligados de diversos domínios, tais como saúde, música, publicações científicas, televisão, dados estatísticos etc.

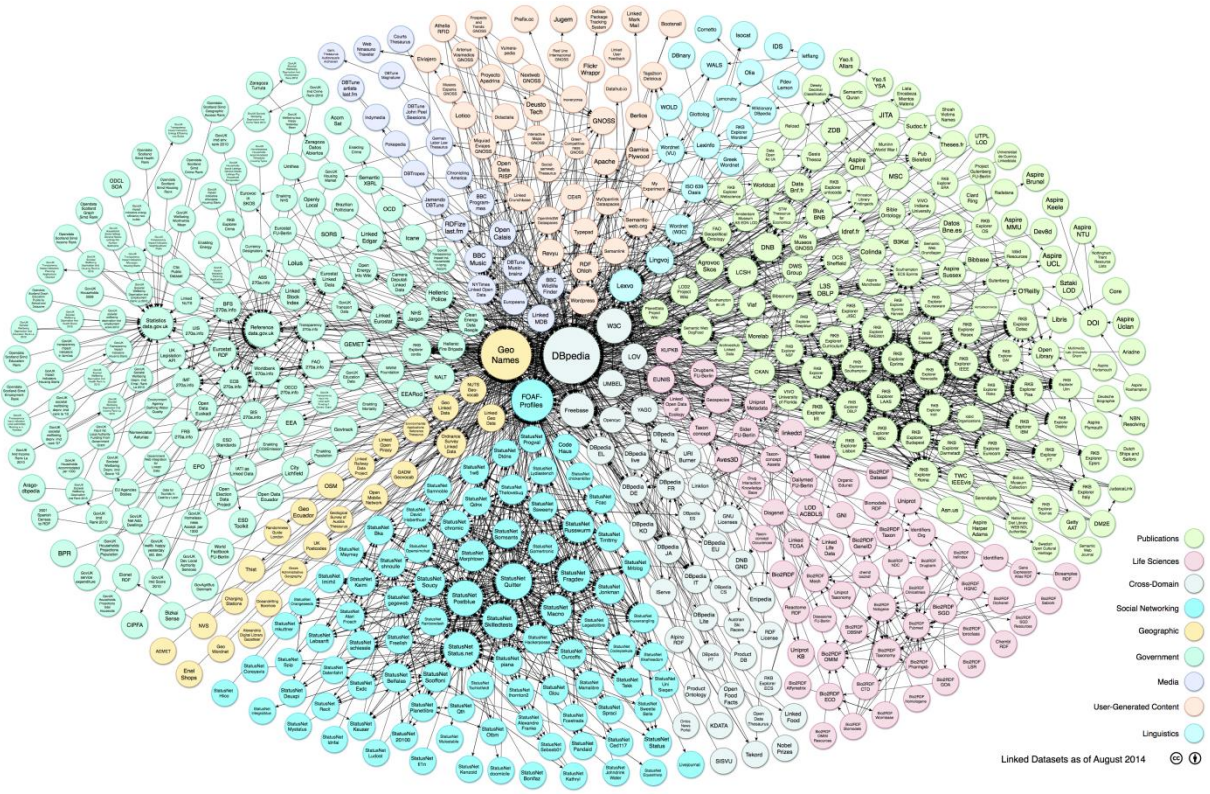
Tecnicamente, Dados Interligados refere-se a dados publicados na Web, de tal forma que é interpretável por máquina, o seu significado é explicitamente definido, ele está ligado a outros conjuntos de dados externos, e pode ser ligado a partir de conjuntos de dados externos (BIZER, HEATH e BERNERS, 2009).

Os Princípios de Dados Interligados foram propostos por (BERNERS-LEE, 2006) e são enumerados a seguir:

1) usar URIs como nomes para coisas;
2) usar URIs HTTP para que as pessoas possam consultar esses nomes;
3) quando alguém consulta um URI, fornece informação útil, usando os padrões (RDF, SPARQL);
4) incluir ligações para outras URIs, a fim que possam permitir a descoberta de mais coisas.

O exemplo mais conhecido da adoção e aplicação dos princípios Dados Interligados é o projeto *Linking Open Data*[1] fundado em janeiro de 2007 e apoiado pela *W3C Semantic Web Education* e *Outreach Group*[2]. O objetivo principal desse projeto é o desenvolvimento da web de dados, obtendo um conjunto de dados que estão disponíveis sobre licença aberta e convertendo-os para o formato RDF e publicar na Web usando os princípios dos Dados Interligados (BIZER, HEATH e BERNERS, 2009).

Figura 1 — Nuvem Linking Open Data




Fonte: Elaborada por Richard Cyganiak e Anja Jentzsch. Atualizado em 30/08/2014.

A Figura 1 mostra um conjunto de dados que foram publicados e interligados pelo projeto até agora. Contendo, 570 fontes de dados que são conectadas por 2909 conjuntos de ligações.

Para facilitar o entendimento da Web de Dados, faremos uma comparação com a Web de Documentos que já conhecemos. A Web de Dados pode ser acessada a partir de navegadores RDF, assim como os navegadores HTML são usados para acessar a Web de Documentos. Enquanto na Web de Documentos usamos links HTML para navegar entre diferentes páginas, na Web de Dados os links RDF são usados para acessar dados de outras fontes. Portanto, os links de hipertexto são capazes de conectar os documentos, assim como

[1] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
[2] http://www.w3.org/2001/sw/sweo/

os links RDF interligam os dados (MAGALHÃES, 2012). A Tabela 1 ilustra algumas comparações entre a Web de Documentos e a Web de Dados.

Tabela 1 — Comparações entre a Web de Documentos e a Web de Dados

| Web de Documentos | Web de Dados |
|---|---|
| Navegadores HTML | Navegadores RDF |
| Links HTML conectando documentos | Links RDF interligando dados |
| Mecanismo de identificação – URIs | Mecanismo de identificação – URIs |
| Mecanismo de acesso – HTTP | Mecanismo de acesso – HTTP |
| Formato de conteúdo – HTML | Modelo de dados – RDF |
| — | Linguagem de consulta – SPARQL |

Fonte: Dissertação (MAGALHÃES, 2012).

A Web de Dados é baseada em um conjunto de padrões, como: um mecanismo de identificação universal e único URI (ou IRI); um mecanismo de acesso universal HTTP; um modelo padrão para representação dos dados, o RDF e uma linguagem de consulta para acesso aos dados, a linguagem SPARQL. A seguir esses padrões serão apresentados:

a) ***URI* (*Uniform Resource Identifier*)** – Identificador Uniforme de Recursos (URI – *Uniform Resource Identifier*) é uma sequência compacta de caracteres que identifica um recurso físico ou abstrato (BERNERS-LEE, FIELDING e MASINTER, 2005). Foi definido em uma ação conjunta da *World Wide Web Consortium*[3] e *Internet Engineering Task Force*[4]. *URIs* são usadas na Web de Dados para identificar objetos e conceitos, permitindo que eles sejam desreferenciados, obtendo informações ao seu respeito. Assim, uma *URI* desreferenciada resulta em uma descrição RDF do recurso identificado. Por exemplo, a URI http://www.w3.org/People/Berners-Lee/card#i, identifica o pesquisador Tim Bernes-Lee.

b) ***IRI* (*Internationalized Resource Identifier*)** – Foi definido em 2005 pela *Internet Engineering Task Force* com um novo padrão de internet para estender o padrão URI. A principal diferença dos dois protocolos é que o URI é limitado a um subconjunto dos caracteres ASCII (60 caracteres), enquanto o IRI contém caracteres do *Universal Character Set*, incluído chinês, japonês e outros caracteres.

c) **HTTP (*HyperText Transfer Protocol*)** – ou protocolo que transferência de Hipertexto (em português), é o mecanismo padrão de acesso a dados e

[3] http://www.w3.org/Consortium/
[4] https://www.ietf.org/about/

documentos na web de documentos e web de dados. É um protocolo de comunicação entre sistema de informação, e permite a transferência de dados entre as redes de computadores, principalmente na Internet.

d) **RDF (*Resource Description Framework*)** – É um framework para representar informações na Web (CYGANIAK, WOOD e LANTHALER, 2014). Em RDF, um recurso pode se relacionar com outro recurso ou com dados, onde são estruturados em forma de triplas (Sujeito, Predicado, Objeto). O sujeito representa o recurso sobre o qual se quer escrever uma sentença, o predicado é uma relação binaria que relaciona o recurso com outro recurso ou dados, uma propriedade também é um recurso e deve possuir um identificador único. O objeto representa o recurso ou objeto que se relaciona com o sujeito. Tabela 2 contém um exemplo de duas triplas relacionadas ao recurso *<http://www.ufc.br/narcisoarruda>*, o predicado *foaf:name* relaciona o sujeito a uma literal e o predicado *vex:orientadoPor* relaciona o sujeito com outro recurso.

Tabela 2 — Triplas RDF

| Sujeito | Predicado | Objeto |
|---|---|---|
| *<http://www.ufc.br/narcisoarruda>* | *foaf:name* | "Narciso Moura Arruda Jr" |
| *<http://www.ufc.br/narcisoarruda>* | *vex*[5]*:orientadoPor* | *<http://www.ufc.br/vaniavidal>* |

Fonte: Elaborada pelo autor.

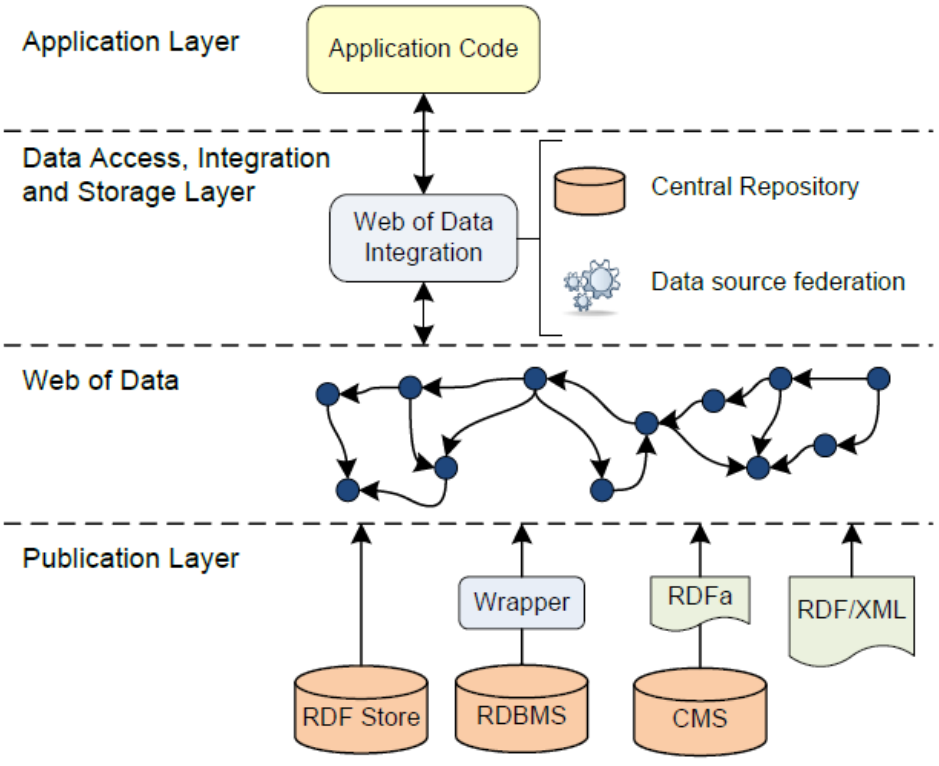
e) **SPARQL** – SPARQL é a linguagem de consulta padrão recomendada pela W3C, para recuperação dos dados contidos em grafos RDF. Apesar de existirem outras linguagens de consulta (SeRQL[6], RQL[7], etc.) mais antigas e com maior poder de expressividade que o SPARQL, elas foram projetadas para domínios específicos ou são interpretadas apenas por algumas ferramentas, o que resulta em uma baixa interoperabilidade. O SPARQL funciona como linguagem de consulta, assim como o SQL funciona para bancos de dados relacionais. Semelhante ao SQL, o SPARQL possui uma estrutura semelhante Select-From-Where.

[5] Vocabulário fictício usado nesse exemplo
[6] http://www.w3.org/2001/sw/wiki/SeRQL
[7] https://github.com/persvr/rql

## 2.3 *Mashup* de Dados Interligados (*Linked Data Mashup*)

*Mashups* de Dados Interligados (MDI) são aplicações construídas para realizar integração de dados no contexto de Dados Interligados (MAGALHÃES, 2012). Normalmente, podemos distinguir *Mashups* de Dados Interligados em duas arquiteturas padrões (HARTIG; ZHAO, 2010), que estão representadas na Figura 2. A primeira arquitetura é para um enfoque materializado, e consiste em coletar e centralizar os dados coletados das fontes. As consultas são executadas nesses dados centralizados. Esse enfoque tem a desvantagem de ter um gasto adicional de espaço em disco, e como os dados são armazenados localmente, tendem a ficar desatualizados. Os principais desafios são a criação e a manutenção da visão de integração. A segunda arquitetura é para um enfoque Virtual, que consiste em executar consultas federadas em fontes de dados definidas (MAGALHÃES, 2012). O principal desafio é a construção de planos de execução sobre várias fontes de dados com desempenho satisfatório. Porém, as consultas resultantes sempre usam os dados atualizados. As vantagens são atualização dos dados e não há necessidade de espaço adicional de armazenamento, além do espaço reservado aos dados originais.

Figura 2 — Arquitetura Geral dos *Mashups* de Dados Interligados




Fonte: Dissertação (MAGALHÃES, 2012).

Os dois maiores problemas encontrados na construção de aplicações de dados interligados é heterogeneidade dos vocabulários e IRI pseudônima (SCHULTZ, MATTEINI, *et al.*, 2012). Uma pequena parcela das fontes de Dados Interligados utiliza termos de vocabulários amplamente aceitos para descrever tipos comuns de entidades, como pessoas, organizações, publicações e músicas. Para as mais especializadas, entidades de domínio específico, tais como gene, descrição de linhas de metrô e dados científicos, nenhum vocabulário foi difundido ainda. As fontes de dados nestes domínios usam termos proprietários (BIZER, JENTZSCH e CYGANIAK, 2011). Um segundo problema é a identidade de ligações. Algumas fontes de dados possuem conjuntos de ligações *owl:sameAs* para a mesma entidade em outras fontes de dados. Muitas outras fontes não se preocupam em identificar essas ligações (BIZER, JENTZSCH e CYGANIAK, 2011).

Em contraste com a heterogeneidade da Web, é interessante para a aplicação ter todos dados que descrevem uma classe de entidades, representados pelo mesmo vocabulário. Ao invés de ser confrontada com *IRIs* pseudônimas, que se referem a dados que podem ou não descrever a mesma entidade, aplicações de Dados Interligados preferencialmente descrevem a mesma entidade com o mesmo IRI, para facilitar muitas tarefas, incluído consultas, agregações e visualizações (SCHULTZ, MATTEINI, *et al.*, 2012).

De modo a facilitar a utilização dos dados Web no contexto das aplicações, é aconselhável, converter dados para um único vocabulário alvo (mapeamento do vocabulário) e substituir IRI que representam o mesmo objeto do mundo real, por uma única IRI alvo (resolução de identidade), antes de fazer qualquer processamento mais sofisticado.

O processo de integração de dados ou *mashup* de dados, normalmente é dividido em três etapas: integração de esquema, detecção de duplicação e fusão (BLEIHOLDER e NAUMANN, 2008). A seguir descrevemos as etapas:

a) a primeira etapa consiste no alinhamento dos esquemas usados para descrever diferentes fontes de dados, a fim de facilitar a identificação e a fusão dos recursos nas próximas etapas. Isto pode ser conseguindo pela integração de esquemas ou mapeamento de esquemas. No primeiro caso, um esquema ideal é definido com base nos esquemas individuais com objetivo de integrá-los. No segundo caso, o objetivo é criar um mapeamento do esquema das fontes para esquema alvo. O principal objetivo dessa etapa é a reduzir o espaço de busca do problema, ou seja, diminuir a quantidade de comparações entre os recursos de diferentes datasets, passando a comparar apenas recursos que são de elementos iguais ou similares do esquema;

b) a segunda etapa consiste na detecção de duplicação, objetivo de identificar e ligar recursos que correspondem ao mesmo objeto do mundo real. Isto é conhecido também como *record linkage*, *deduplication*, *object identification*, *reference reconciliation*, *link discovery*, *interlinking*, etc. A saída para esse problema normalmente é um identificador unificado que descreve os indivíduos duplicados ou ligações entre as entidades (exemplo *owl:sameAs*);

c) a terceira etapa consiste em realizar a fusão dados. A fusão é o processo de combinar as propriedades de dois ou mais recursos que representam o mesmo objeto no mundo real, para produzir uma descrição universal para o objeto. Nessa etapa as entidades vinculadas são fundidas para a produção de uma entidade mais rica, mais completa e mais correta. É onde se resolver os potenciais conflitos e irregularidades de propriedades de entidades correspondentes, tais como diferentes valores ou domínios para a mesma propriedade, a falta de valores ou propriedades, as diferenças de qualidade dos conjuntos de dados, etc. O maior desafio desta tarefa é aplicar de forma eficiente a estratégia mais adequada de resolução de conflito, para as propriedades a serem usadas.

## 2.4 Conclusão

Nesse capítulo, apresentamos os assuntos mais relevantes e necessários para o entendimento dos capítulos a seguir. Entre os assuntos que abordamos estão os conceitos de Dados Interligados e *Mashup* de Dados Interligados.

# 3 TRABALHOS RELACIONADOS

## 3.1 Introdução

Pelo nosso conhecimento, a manutenção incremental de visões MDI ainda não foi abordada em nenhum framework. Neste Capítulo, falaremos dos principais trabalhos relacionados ao nosso tema, e mostraremos por que eles não são adequados a resolver o nosso problema.

## 3.2 Incremental Maintenance of RDF Views of Relational Data (VIDAL, CASANOVA e CARDOSO, 2013)

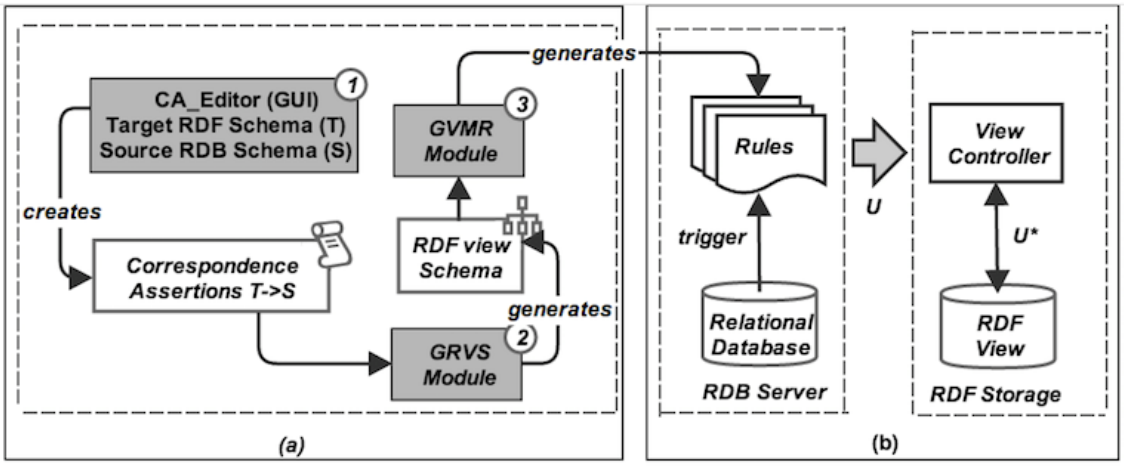
Esse trabalho propõe um framework baseado em regras, para a manutenção incremental de visões RDF definidas sobre dados relacionais. Nesse framework é possível gerar automaticamente as regras, com base no mapeamento entre o esquema relacional e uma ontologia alvo, para manter atualizada a visão RDF.

A estratégia de manutenção incremental é baseada em regras, para definição de uma visão RDF sobre dados relacionais. A estratégia tem dois passos principais:

a) *mapping generation step* (etapa de geração de mapeamento): Depende do designer para especificação de mapeamento entre o esquema relacional e uma ontologia alvo e resulta em uma especificação de como representar conceito de esquema relacional em termos classes e propriedades RDF de escolha do designer. O mapeamento induz uma visão RDF que é exportada a partir da fonte de dados;

b) *view maintenance rules generation step* (etapa de geração de regras de manutenção): Gera automaticamente as regras necessárias para manutenção incremental da visão RDF a partir das regras de mapeamento.

Figura 3 — Componentes do Framework




Fonte: Artigo (VIDAL, CASANOVA, *et al.*, 2014).

Figura 3 descreve os principais componentes do Framework. Resumidamente, o administrador de um banco de dados relacional usa *Rubya* (***Ru**les **by a**ssertion*), para criar uma visão RDF e definir um conjunto de regras usando *Rubya* – Figura 3 (a). Estas regras são responsáveis por:

(i) calcular as declarações de manutenção necessárias para manter a visão materializada **V** com respeito a atualizações da base;

(ii) enviar as declarações de manutenção da visão para a *view controller* – Figura 3 (b).

A *view controller* da visão RDF tem as seguintes funcionalidades:

(i) receber as atualizações de manutenção da visão do servidor RDB;

(ii) aplicar as atualizações para a visão em conformidade.

A solução dada para manter atualizada a visão utiliza manutenção incremental, mas o problema trata de visões geradas de base de dados relacionais. Logo, não seria possível tratar nosso problema com essa abordagem.

## 3.3 LDIF (Linked Data Integration Framework)

LDIF é um framework para criação de visões MDI materializadas. A integração do LDIF consiste nos seguintes passos:

a) coletar Dados: Um módulo de importação replica um conjunto de dados localmente por meio de download de arquivos, crawling ou SPARQL;

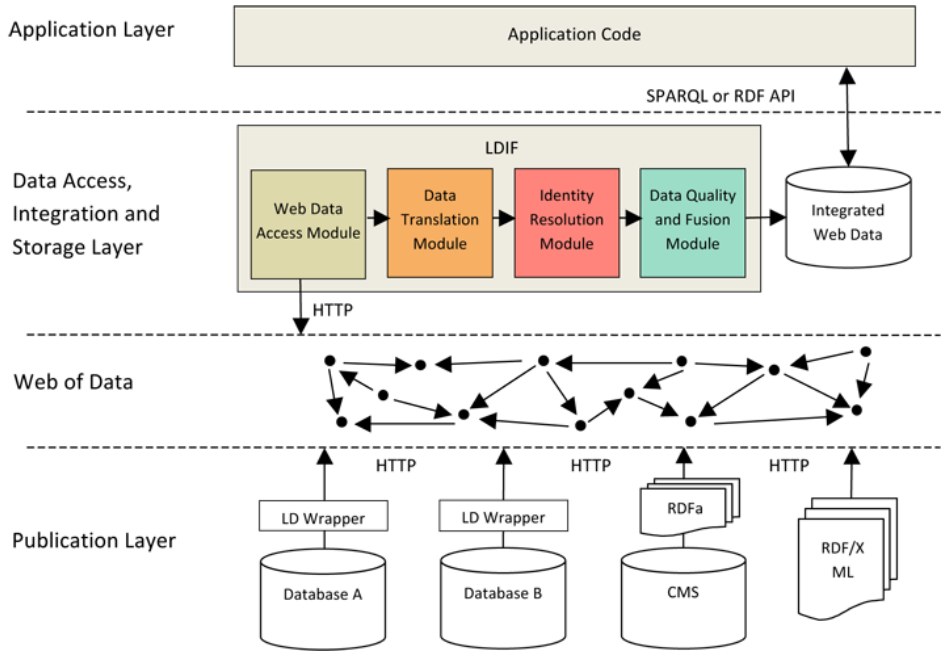
b) mapeamento de Esquema: Uma linguagem expressiva de mapeamento permite traduzir os dados dos vários vocabulários que são usados na Web para um único vocabulário de destino consistente;

c) resolver Identidades: Um componente de resolução de identidade descobre URIs diferentes para mesmo objeto no mundo real na entrada e os altera por uma única URI alvo;

d) avaliação de Qualidade e Fusão de Dados: Um componente de limpeza de dados filtra dados de acordo com as diferentes avaliações de qualidade e prevê a fusão de dados de acordo com diferentes métodos de resolução de conflitos.

e) saída: LDIF retorna os dados integrados e podem ser gravados em arquivo ou para um QuadStore[8].

Figura 4 — Papel do LDIF dentro da arquitetura de uma aplicação de Dados Ligados




Fonte: Página Web http://ldif.wbsg.de/ (SCHULTZ, MATTEINI, *et al.*, 2012).

A Figura 4 destaca as etapas do processo de integração de dados que são suportados pelo LDIF.

[8] É um banco de dados RDF que armazena triplas RDF com um elemento adicional de informação. Exemplos: OpenLink Virtuoso, Fuseki e OpenRDF Sesame

## 3.4 FAGI

FAGI (GIANNOPOULUOS, SKOUTAS, *et al.*, 2014) é um framework projetado para suportar amplas funcionalidades para fusão de dados RDF geoespaciais. O framework apresenta duas ferramentas, *FAGI-tr* para traduzir representações RDF de dados geoespaciais, e *FAGI-gis* que fornece a infraestrutura para processamentos e transformações geométricas para fusões de propriedades geométricas de dados de entidades ligadas.

*FAGI for transformations (FAGI-tr)*, fornece um framework para:

a) carregar uma fonte e um dataset geoespacial RDF alvo;
b) identificar vocabulários para representar dados geoespaciais RDF;
c) selecionar, de ambos datasets, as representações a serem consideras para tratamento.
d) selecionar um vocabulário alvo e traduzir todas as triplas geoespaciais de ambos datasets para o respectivo formato;
e) resultando em dois datasets (alinhados com relação a seus vocabulários geoespaciais) para processamentos futuros.

*FAGI-geospatial processing (FAGI-gis)*, fornece um framework para:

a) indexar propriedades geoespaciais;
b) calcular similaridade entre geometrias;
c) aplicar estratégia de fusão geométrica, tanto por combinação ou por transformações geométricas, implementa um conjunto inicial de funções que podem ser facilmente estendidas.

FAGI é um *Framework* para criação de mashup de Dados Interligados de dados geoespaciais. Ele não trata do formalismo da criação da visão de MDI, além de não tratar da atualização dos dados. O que torna necessário o reprocessamento de todos os dados.

## 3.5 Conclusão

O problema de manutenção de visão incremental tem sido extensivamente estudado na literatura. No entanto, na maior parte da literatura, a visão é definida através de uma única fonte de dados, por exemplo, para visões relacionais (CERI e WIDOM, 1991), para

visões orientadas a objeto (KUNO e RUNDENSTEINER, 1998), para visões semiestruturadas (ABITEBOUL, MCHUGH, *et al.*, 1998), para visões XML (DIMITROVA, EL-SAYED e RUNDENSTEINER, 2003), (VIDAL, LEMOS, *et al.*, 2008) e para visões RDF (VIDAL, CASANOVA e CARDOSO, 2013). Nenhuma das técnicas propostas pode ser diretamente aplicada para visões de integrações de dados.

A manutenção incremental de visão de integração de dados é vista em (GRIFFIN e LIBKIN, 1998), (GUPTA e MUMICK, 2000). O foco nesses trabalhos é a visão relacional. Em (GUPTA e MUMICK, 2000), foi desenvolvido um algoritmo para a manutenção incremental de outerjoin e visões relacionadas. Em (GRIFFIN e LIBKIN, 1998), algoritmos de propagação de alterações algébricas foram desenvolvidos para a manutenção da visão outerjoin. Apesar de suas importantes contribuições, nenhuma dessas técnicas pode ser aplicada para visão MDI, uma vez que a natureza das fontes dados ligados coloca novos desafios para lidar com o problema de fusão de dados.

Recentemente, foram propostos dois frameworks para a criação de visões MDI. O framework ODCleanStore (KNAP, J., *et al.*, 2012) oferece fusão de Dados Interligados, lidando com inconsistências. LDIF – Linked Data Integration Framework (SCHULTZ, MATTEINI, *et al.*, 2012) oferece uma linguagem de mapeamento, lida com remapeamento de URIs e usa grafos nomeados para registrar proveniência de dados.

# 4 FRAMEWORK BASEADO EM ONTOLOGIA PARA ESPECIFICAR VISÕES MDI

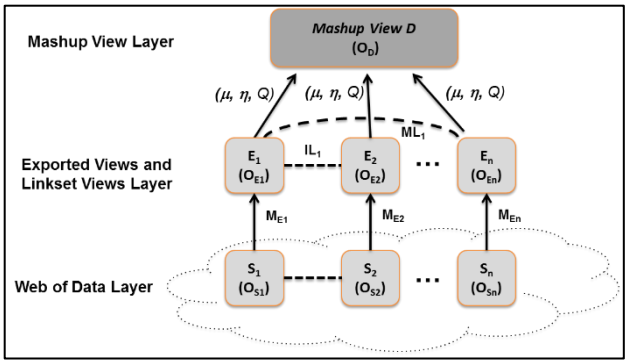
## 4.1 Introdução

Neste Capítulo, discutimos um framework baseado em ontologia para especificar visões de *mashup* de Dados Interligados, ele possui três camadas: Visão de *Mashup* (*Mashup View*), Web de Dados (*Web of Data*) e Visão Exportada (*Exported View*) e Visão de Conjunto de Ligações (*Linkset Views*). No framework mostramos como especificar as visões exportadas, as visões de conjunto de ligações *sameAs*, a função de normalização, as métricas de qualidade e as assertivas de fusão, que são utilizadas para especificar formalmente as visões MDI. Mostramos também, o processo de materialização através dessas especificações. Apresentamos, o estudo de caso que é utilizado em toda dissertação. No estudo de caso, utilizamos uma aplicação sobre dados de música, e mostramos passo a passo a especificação e materialização da visão MDI, conforme proposto no framework.

## 4.2 Visão geral do Framework

Nosso framework para especificar formalmente as visões MDI é resumido na Figura 5.

Figura 5 — Três Camadas do Framework Baseado em Ontologia




Fonte: Elaborada pelo autor.

Na camada *Visão de Mashup*, a ontologia da visão de *mashup* $O_D$ especifica os conceitos da aplicação de *mashup* (ou seja, o modelo conceitual), que é o vocabulário comum para integração de dados exportados pelas fontes de Dados Interligados.

Na camada *Web de Dados*, cada fonte de dados $S_i$ é descrita por uma *ontologia fonte* $O_{S_i}$, publicadas na Web de acordo com os princípios de Dados Interligados. Estas ontologias fontes são representadas na camada *Web de Dados* na Figura 5.

Para cada fonte de dados interligados $S_i$, exportamos uma visão $E_i$. Cada uma dessas visões $E_i$ tem uma ontologia $O_{Ei}$ e um conjunto de regras $M_{Ei}$, que mapeiam conceitos de $O_{Si}$ para conceitos de $O_D$. O vocabulário de $O_{Ei}$ é um subconjunto do vocabulário de $O_D$, cujos termos ocorrem na cabeça das regras em $M_{Ei}$. As ontologias exportadas são representadas nas camadas *Visão Exportada* e *Visão de Conjunto de Ligações* na Figura 5.

Consideramos dois tipos de ligações *sameAs*: *Ligação sameAs exportada* ***(exported sameAs link)***, que são exportadas das fontes de Dados. *Ligação sameAs de Mashup* ***(mashup sameAs link)***, que são criadas automaticamente, baseadas na especificação da visão de conjunto de ligações *sameAs* (CASANOVA, VIDAL, *et al.*, 2014), especificamente definida para aplicação de *mashup*.

Conforme detalhado nas seguintes seções, a especificação de uma visão MDI é uma n-tupla $\lambda = (D, O_D, E_1, ..., E_n, IL_1, ..., IL_p, ML_1, ..., ML_q, \mu, \eta, Q)$, onde:

a) $D$ é o nome da visão de *mashup*;
b) $O_D$ é a ontologia da visão de *mashup*;
c) $E_1, ..., E_n$ são *especificações das visões exportadas* com ontologias $O_{E1},...,O_{En}$, cujo os vocabulário são subconjuntos do vocabulário $O_D$;
d) $IL_1, ..., IL_p$ são *especificações das visões de conjuntos de ligações sameAs exportadas* entre $E_1,...,E_n$;
e) $ML_1, ..., ML_q$ são *especificações de visões de conjuntos de ligações sameAs de mashup* entre $E_1,...,E_n$;
f) $\mu$ é um *conjunto de regras de fusão* de $O_{E1},...,O_{En}$ para $O_D$;
g) $\eta$ é um *símbolo da função de normalização*, cujo a interpretação definida como remapear IRIs de visões exportadas para IRIs das visões MDI;
h) $Q$ é um *conjunto de métricas de avaliação de qualidade*, que são usadas para qualificar a qualidade das fontes de Dados Interligados.

O processo de geração da especificação $\lambda$ de uma visão MDI consiste em cinco etapas:

1) modelagem da ontologia da visão do *mashup*;
2) geração das especificações das visões exportadas;
3) geração das especificações das visões de conjuntos de ligações *sameAs* exportadas;
4) geração das especificações das visões de conjuntos de ligações *sameAs* de *mashup*;
5) definição da função de normalização, métricas de avaliação de qualidade e regras de fusão. Etapas 2 a 5 serão detalhadas nas seções a seguir.

No framework proposto, a materialização de uma especificação da visão MDI é automaticamente processada baseada na sua especificação e consiste de quatro passos:

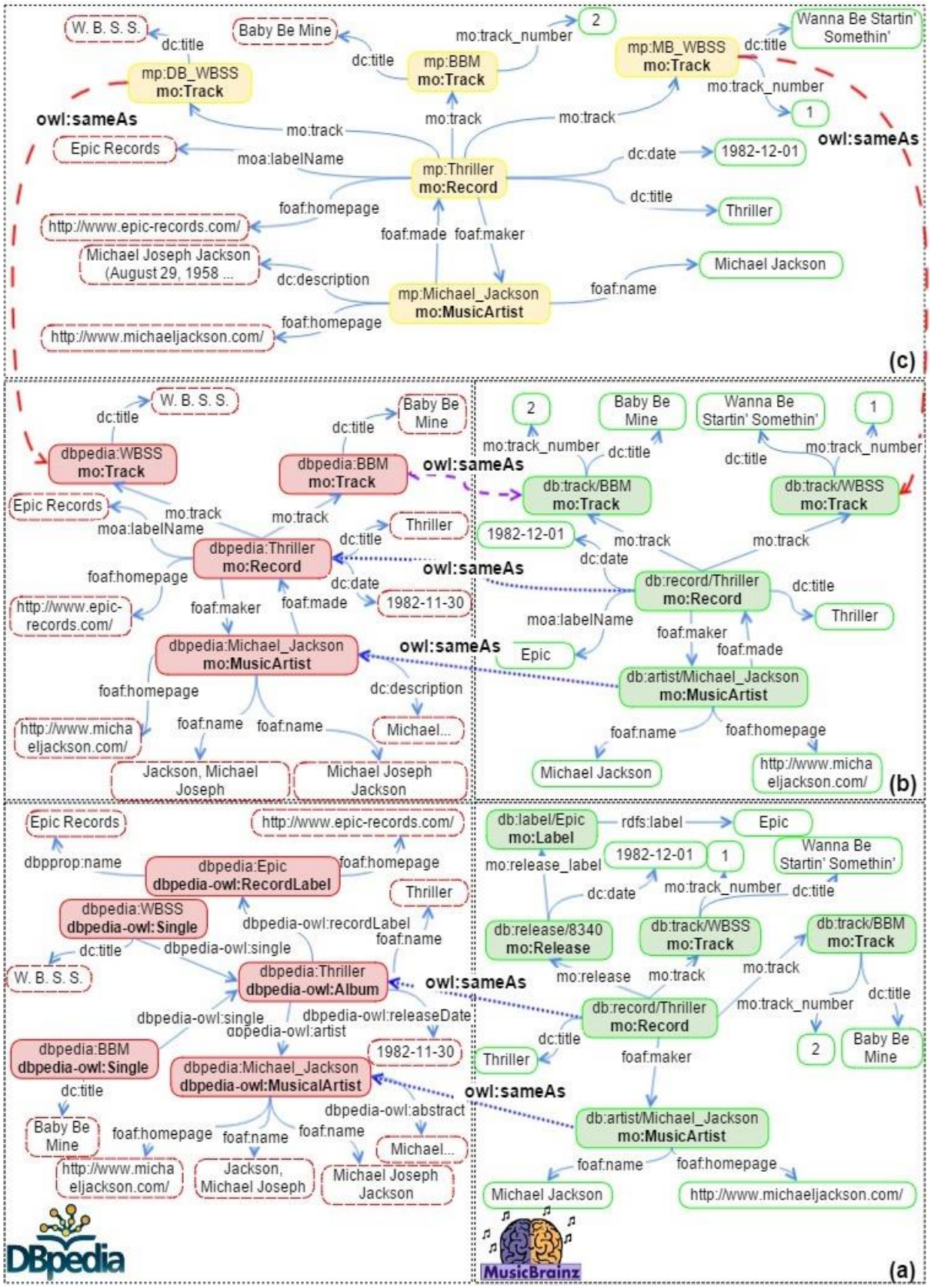
1) materialização das *visões exportadas*. Nessa etapa os dados da fonte são traduzidos para o vocabulário da *visão exportada*, conforme especificado pelas regras de mapeamento na especificação da *visão exportada*.
2) materialização das *visões de conjunto de ligações sameAs exportadas.* Dado uma *visão de conjunto de ligações IL* sobre a fonte de dados ligados *S*, essa etapa exporta *IL* para a materialização da visão de LDM.
3) materialização das *visões de conjunto de ligações sameAs de mashup.* Dado uma especificação da *visão de conjunto de ligações sameAs de mashup ML*, este passo calcula as ligações sameAs baseadas na especificação *ML*.
4) materialização da *visão de mashup*. Esta etapa materializa a *visão de mashup*, aplicando a função de normalização e as regras de mapeamentos para as visões exportadas materializadas e as visões de conjuntos de ligações *sameAs* materializadas. Isto inclui a combinação e fusão de múltiplas representações do mesmo objeto do mundo real para uma única representação e a resolução de inconsistências dos dados.

## 4.3 Estudo de Caso

No estudo de caso, adotamos um exemplo simples de uma aplicação de *mashup* sobre música, chamada *DBB_Music*, que integra dados de duas fontes de dados vinculados: *DBpedia* (BIZER, LEHMANN, *et al.*, 2009) e *MusicBrainz* (SWARTZ, 2002).

Nesta seção, mostramos o passo inicial da geração da especificação da visão MDI, que é a modelagem da ontologia da visão de *mashup*. Nas próximas seções, conforme a definição das especificações (visão exportada, visão de conjunto de ligações, função de normalização, métricas de avaliação de qualidade e fusão de dados), desenvolvemos nosso estudo de caso.

Figura 6 — (a) Estado das Fontes de Dados; (b) Visões Exportadas e Visões dos conjuntos de Ligações Materializados; (c) Visão MDI

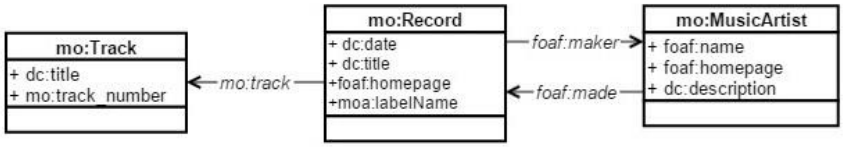



Fonte: Elaborada pelo autor.

### *4.3.1 Ontologia de Aplicação e Fontes de Dados*

A Figura 7 representa, em notação UML, a ontologia de aplicação *MUSIC_OWL* para nosso *mashup*, que reutiliza termos de três vocabulários bem conhecidos: FOAF (*Friend of a Friend*), MO (*Music Ontology*) e DC (*Dublin Core*). Usamos o prefixo "*moa:*" para os novos termos definidos na ontologia *MUSIC_OWL*.

Figura 7 — *MUSIC_OWL*



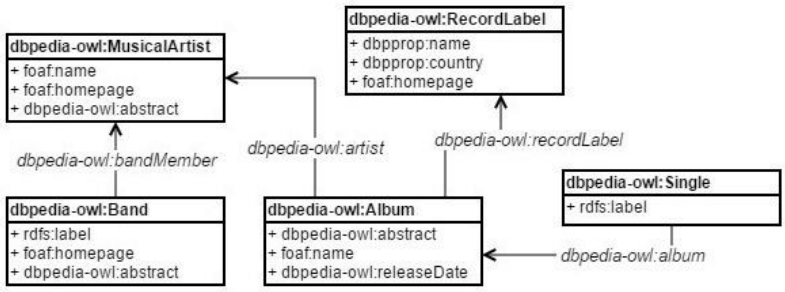

Fonte: Elaborada pelo autor.

Após a modelagem da ontologia de aplicação é importante a seleção das fontes de dados, como dito anteriormente, escolhemos as fontes *MusicBrainz* e *DBpedia*, onde:

a) *DBpedia* é uma fonte de dados cujo conteúdo foi extraído de forma estruturada da Wikipédia, essa informação estruturada é disponibilizada na Web. *DBpedia* usa a *DBpedia Ontology* (The DBpedia Ontology, 2014), que chamaremos de *DBPEDIA_OWL*, e usaremos prefixo "*dbpedia-owl*:" para se referir a ela, Figura 8 podemos ver fragmento da ontologia *DBPEDIA_OWL*;

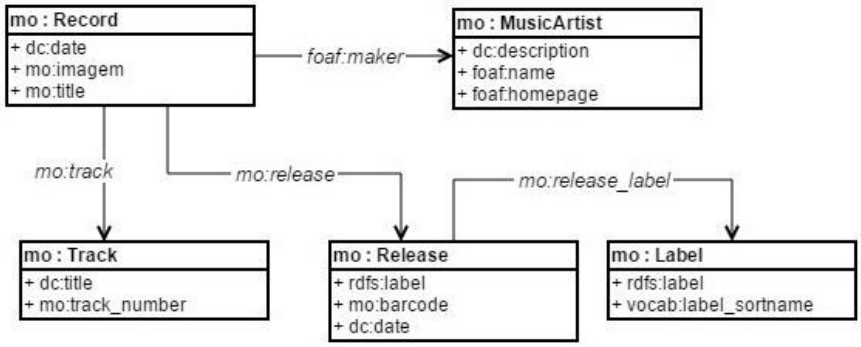
b) *MusicBrainz* é uma enciclopédia aberta de músicas, possui detalhes sobre músicas gravadas, tais como nomes de artistas, títulos de lançamentos, listas de faixas entre outras coisas e torna disponível ao público. *MusicBrainz* utiliza a ontologia, *Music Ontology* (RAIMOND, ABDALLAH, *et al.*, 2007), que chamaremos de *MUSICBRAINZ_OWL* e usamos o prefixo "*mo:*" para se referir a ela, Figura 9 podemos ver fragmento da ontologia MUSICBRAINZ_OWL.

Figura 8 — *DBPEDIA_OWL*




Fonte: Elaborada pelo autor.

Figura 9 — *MUSICBRAINZ_OWL*




Fonte: Elaborada pelo autor.

## 4.4 Especificação e Materialização de visões exportadas

### *4.4.1 Especificação de Visões Exportadas*

Uma especificação de visão exportada é uma quíntupla *(E, S, $O_S$, $O_E$, $M_E$)*, onde:

a) *E* é o nome da visão;

b) *S* é uma fonte de Dados Interligados;

c) $O_S$ é a ontologia de *S*;

d) $O_E$ é a ontologia da visão exportada, tal que os termos de seu vocabulário ocorrem na cabeça das regras de mapeamento em $M_E$;

e) $M_E$ é um conjunto de regra de mapeamento de $O_S$ para $O_E$.

No framework proposto, para cada fonte de dados interligados exportamos uma visão, cuja especificação é automaticamente gerada considerando a ontologia fonte $O_S$ e a ontologia da visão de *mashup* $O_D$ (ontologia alvo). O processo de geração das visões exportadas consiste de dois passos:

a) o primeiro é gerado as regras que mapeiam conceitos de $O_D$ em conceitos de $O_S$;

b) o segundo passo envolve a geração do esquema da visão exportada o qual é induzido das regras de mapeamento conforme descrito em (Vidal, Casanova, & Cardoso, 2013). O vocabulário da visão exportada contém todos os

elementos da ontologia de domínio que têm correspondência com algum elemento da ontologia da fonte. Note, portanto, que o vocabulário da ontologia $O_E$ é um subconjunto do vocabulário da ontologia $O_D$.

#### *4.4.1.1 Regras de Mapeamento*

Nesta seção apresentamos um formalismo baseado em regras para especificar mapeamento entre ontologias.

Um vocabulário *V* é um conjunto de *classes* e *propriedades*. Uma ontologia é um par $O=(V,\Sigma)$, tal que *V* é um vocabulário e $\Sigma$ é um finito conjunto de fórmulas em *V*, as restrições de *O*.

Seja $O_T=(V_T,\Sigma_T)$ uma *ontologia alvo* e $O_S=(V_S,\Sigma_S)$ uma *ontologia fonte*. Seja $\boldsymbol{X}$ um conjunto de variáveis, disjuntas de $V_T$ e $V_S$ e $\boldsymbol{F}$ um conjunto de *símbolos de função*, disjunta de $\boldsymbol{X}$, $V_T$ e $V_S$. Assumimos que cada símbolo de função tem uma determinada aridade. Os símbolos de função 0-ário são chamados de constantes, incluímos IRIs e valores de tipo de dados.

Um termo é uma expressão recursiva construída de símbolos de função, constantes e variáveis, como de costume.

Um literal é uma expressão de uma das formas:

d) um literal Classe da forma $C(t)$, onde *C* é uma classe em $V_T \cup V_S$ e *t* é um termo;

e) um literal Propriedade $P(t, u)$, onde *P* é uma propriedade em $V_T \cup V_S$ e *t* e *u* são termos;

f) $u = f(t_1,\ldots,t_n)$, onde *f* é um símbolo de função n-ário em F e $u, t_1,\ldots,t_n$ são termos.

Um *corpo de regra B* é uma lista de literais. Quando necessário, usamos *"B*$[x_1,\ldots,x_k]$*"* para indicar que as variáveis $x_1,\ldots,x_k$ ocorrem em *B*. Dizemos que *B* está sobre $V_T$ (ou $V_S$), se somente se, todas as classes e propriedades que ocorrem em B são de $V_T$ (ou $V_S$).

Uma regra de mapeamento de $O_S$ para $O_T$, ou simplesmente uma regra de $O_S$ para $O_T$ é uma expressão de uma das formas:

a) $C(x) \leftarrow B[x]$, onde $C$ é uma classe em $V_T$ e $B[x]$ é um corpo de regra sobre $V_S$;

g) $P(x,y) \leftarrow B[x,y]$, onde $P$ é uma propriedade em $V_T$ e $B[x,y]$ é um corpo de regra sobre $V_S$

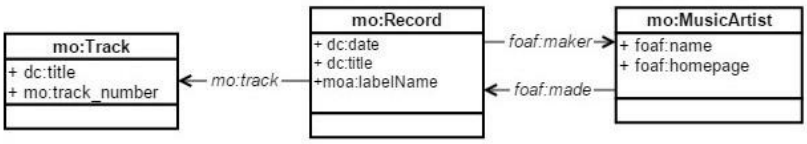
Estendemos a notação de regras de mapeamento para explicar sucessivamente corpos disjuntos e múltiplas fontes de dados.

Um corpo de regra disjunto é uma expressão da forma *"$B_1$ ;…; $B_n$"*, onde $B_i$ é um corpo de regra, para $i=1,...,n$. Note que podemos combinar uma lista de regras com mesma cabeça, "$H \leftarrow B_1$" ,…, "$H \leftarrow B_n$" para uma simples regra com um corpo disjunto, "$H \leftarrow B_1$ ;…; $B_n$".

### *4.4.2 Especificação das Visões Exportadas MusicBrainz_EV e DBpedia_EV*

Para o nosso estudo de caso, geramos duas visões exportadas: *MusicBrainz_EV*, que exporta dados de *MusicBrainz*, e *DBpedia_EV*, que exporta dados da *DBpedia*. Tabela 3 contém as regras de mapeamentos de *MUSICBRAINZ_OWL* para *MUSIC_OWL*, chamaremos essas regras de $M_{MUSICBRAINZ}$. A Figura 10 mostra o esquema da ontologia da visão exportada *MusicBrainz_EV*, induzido pelas regras de mapeamento $M_{MUSICBRAINZ}$.

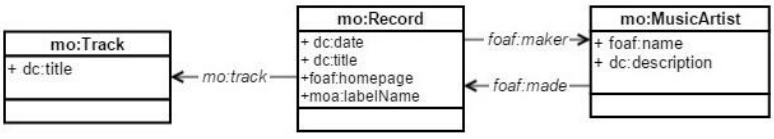
Figura 10 — Ontologia *MusicBrainz_EV*




Fonte: Elaborada pelo autor.

Tabela 4 contém as regras de mapeamentos de *DBPEDIA_OWL* para *MUSIC_OWL*, chamaremos essas regras de $M_{DBPEDIA}$. A Figura 11 mostra o esquema da ontologia da visão exportada *DBpedia_EV* induzido pelas regras de mapeamento $M_{DBPEDIA}$.

Figura 11 — Ontologia *DBPEDIA_OWL*




Fonte: Elaborada pelo autor.

Tabela 3 — Regras de Mapeamento de *MUSICBRAINZ_OWL* para *MUSIC_OWL*

| | |
|---|---|
| mr01 | *mo:Record(r) ← mo:Record(r)* |
| mr02 | *mo:MusicArtist(a) ← mo:MusicArtist(a)* |
| mr03 | *mo:Track(t) ← mo:Track(t)* |
| mr04 | *foaf:homepage(a, h) ← mo:MusicArtist(a), foaf:homepage(a, h)* |
| mr05 | *foaf:name(a, n) ← mo:MusicArtist(a), foaf:name(a, n)* |
| mr06 | *dc:title(s, t) ← mo:Track(s), dc:title(s, t)* |
| mr07 | *mo:track_number(s, t) ← mo:Track(s), mo:track_number(s, t)* |
| mr08 | *mo:track(r, t) ← mo:Record(r), mo:track(r, t)* |
| mr09 | *moa:labelName(r, n) ← mo:Record(r), mo:release(r, s), mo:Release(s), mo:release_label(s, l), mo:Label(l), dbpprop:name(l, n)* |
| mr10 | *dc:date(r, d) ← mo:Record(r), mo:release(r, s), mo:Release(s), dc:date(s, d)* |
| mr11 | *foaf:maker(r, m) ← mo:Record(r), foaf:maker(r, m)* |
| mr12 | *foaf:made(m, r) ← mo:Record(r), foaf:maker(m, r)* |
| mr13 | *dc:title(r, t) ← mo:Record(r), dc:title(r, t)* |

Fonte: Elaborada pelo autor.

Tabela 4 — Regras de mapeamento de *DBPEDIA_OWL* para *MUSIC_OWL*

| | |
|---|---|
| mr01 | *mo:Record(r) ← dbpedia-owl:Album(r)* |
| mr02 | *mo:MusicArtist(a) ← dbpedia-owl:MusicalArtist(a)* |
| mr03 | *mo:Track(t) ← dbpedia-owl:Single(t)* |
| mr04 | *foaf:homepage(a, h) ← dbpedia-owl:MusicalArtist(a), foaf:homepage(a, h)* |
| mr05 | *foaf:name(a, n) ← dbpedia-owl:MusicalArtist(a), foaf:name(a, n)* |
| mr06 | *dc:title(s, t) ← dbpedia-owl: Single(s), rdfs:label(s, t)* |
| mr07 | *mo:track(r, t) ← dbpedia-owl:Single(t), dbpedia-owl:album(r, t)* |
| mr08 | *moa:labelName(r, n) ← dbpedia-owl:Album(r), dbpedia-owl:recordLabel(r, l), dbpedia-owl:RecordLabel(l), dbpprop:name(l, n)* |
| mr09 | *foaf:homepage(r, h) ← dbpedia-owl:Album(r), dbpedia-owl:recordLabel(r, l), dbpedia-owl:RecordLabel(l), foaf:homepage(l, h)* |
| mr10 | *dc:date(r, d) ← dbpedia-owl:Album(r), dbpedia-owl:releaseDate(r, d)* |
| mr11 | *dc:description(a, d) ← dbpedia-owl:MusicalArtist(a), dbpedia-owl:abstract(a, d)* |
| mr12 | *foaf:maker(r, m) ← dbpedia-owl:Album(r), dbpedia-owl:artist(r, m)* |
| mr13 | *foaf:made(m, r) ← dbpedia-owl:Album(r), dbpedia-owl:artist(m, r)* |
| mr14 | *dc:title(r, t) ← dbpedia-owl:Album(r), foaf:name(r, t)* |

Fonte: Elaborada pelo autor.

### *4.4.3 Materialização de Visões Exportadas*

A materialização de uma visão exportada requer traduzir os dados da fonte para o vocabulário da visão exportada, conforme especificado pelas regras de mapeamentos. No nosso framework as regras de mapeamentos são implementadas utilizando a linguagem R2R.

O Framework R2R (BIZER e SCHULTZ, 2010) consiste em uma linguagem de mapeamento para expressar a correspondência entre termos, melhores práticas sobre como publicar mapeamentos na Web e API Java para transformar para transformar dados de acordo com esses mapeamentos (BIZER e SCHULTZ, 2010). A sintaxe da linguagem de mapeamento é muito semelhante à linguagem de consulta SPARQL, o que facilita a curva de aprendizagem. A linguagem de mapeamento abrange a transformação de valores em valores

de diferentes unidades de medida e pode lidar com correspondências do tipo um-para-muitos e muitos-para-um entre os elementos do vocabulário. A API Java transforma os dados da Web para um determinado vocabulário alvo.

Referindo-se ao nosso estudo de caso, Tabela 5 mostra os mapeamentos R2R induzidos pelas regras de mapeamento $M_{DBPEDIA}$. De maneira similar podemos criar os mapeamentos R2R induzidos pelas regras de mapeamento $M_{MUSICBRAINZ.}$

Figura 6 (b) mostra a materialização das visões exportadas *MusicBrainz_EV* e *DBpedia_EV*, obtidas aplicando as regras de mapeamento $M_{MUSICBRAINZ}$ e $M_{DBPEDIA}$, nas fontes de dados MusicBrainz e DBpedia, respectivamente. Na Figura 6 (a), é possível ver o estado das fontes de dados.

Tabela 5 — R2R para materialização de *DBpedia_EV*

```
# Class Mappings

mp:MR01
a r2r:ClassMapping ;
r2r:partOfMappingCollection mp:mapCol_1 ;
r2r:sourcePattern "?SUBJ a dbpedia-owl:Album" ;
r2r:targetPattern "?SUBJ a mo:Record" .

mp:MR02
a r2r:ClassMapping ;
r2r:partOfMappingCollection mp:mapCol_1 ;
r2r:sourcePattern "?SUBJ a dbpedia-owl:MusicalArtist" ;
r2r:targetPattern "?SUBJ a mo:MusicArtist" .

mp:MR03
a r2r:ClassMapping ;
r2r:partOfMappingCollection mp:mapCol_1 ;
r2r:sourcePattern "?SUBJ a dbpedia-owl:Single" ;
r2r:targetPattern "?SUBJ a mo:Track" .

# Property Mappings

mp:MR04
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_2 ;
r2r:mappingRef              mp:MR02 ;
r2r:sourcePattern           "?SUBJ foaf:homepage ?h" ;
r2r:targetPattern           "?SUBJ foaf:homepage ?h" .

mp:MR05
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_2 ;
r2r:mappingRef              mp:MR02 ;
r2r:sourcePattern     "?SUBJ foaf:name ?n" ;
r2r:targetPattern     "?SUBJ foaf:name ?n" .

mp:MR06
a r2r:PropertyMapping ;
```

```
r2r:partOfMappingCollection mp:mapCol_3 ;
r2r:mappingRef                mp:MR03 ;
r2r:sourcePattern      "?SUBJ rdfs:label ?l" ;
r2r:targetPattern      "?SUBJ dc:title ?l" .

mp:MR07
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_1 ;
r2r:mappingRef                mp:MR01 ;
r2r:sourcePattern      "?SUBJ a dbpedia-owl:Single ; dbpedia-owl:album ?a" ;
r2r:targetPattern      "?<a> mo:track ?SUBJ" .

mp:MR08
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_4 ;
r2r:mappingRef                mp:MR01 ;
r2r:sourcePattern      "?SUBJ dbpedia-owl:recordLabel ?r . ?r <http://dbpedia.org/property/name> ?n" ;
r2r:targetPattern      "?SUBJ moa:labelName ?n" .

mp:MR09
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_2 ;
r2r:mappingRef                mp:MR01 ;
r2r:sourcePattern      "?SUBJ dbpedia-owl:recordLabel ?r . ?r foaf:homepage ?h" ;
r2r:targetPattern      "?SUBJ foaf:homepage ?h" .

mp:MR10
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_3 ;
r2r:mappingRef                mp:MR01 ;
r2r:sourcePattern      "?SUBJ dbpedia-owl:releaseDate ?d" ;
r2r:targetPattern      "?SUBJ dc:date ?d" .

mp:MR11
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_3 ;
r2r:mappingRef                mp:MR02 ;
r2r:sourcePattern      "?SUBJ dbpedia-owl:abstract ?d" ;
r2r:targetPattern      "?SUBJ dc:description ?d" .

mp:MR12
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_2 ;
r2r:mappingRef                mp:MR01 ;
r2r:sourcePattern      "?SUBJ dbpedia-owl:artist ?a" ;
r2r:targetPattern      "?SUBJ foaf:maker ?<a>" .

mp:MR13
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_2 ;
r2r:mappingRef                mp:MR01 ;
r2r:sourcePattern      "?SUBJ a dbpedia-owl:Album ; dbpedia-owl:artist ?a" ;
r2r:targetPattern      "?<a> foaf:made ?SUBJ" .

mp:MR14
a r2r:PropertyMapping ;
r2r:partOfMappingCollection mp:mapCol_3 ;
```

```
r2r:mappingRef              mp:MR01 ;
r2r:sourcePattern     "?SUBJ <http://xmlns.com/foaf/0.1/name> ?n" ;
r2r:targetPattern     "?SUBJ dc:title ?n" .
```

Fonte: Elaborada pelo autor.

## 4.5 Especificação e Materialização de visões de Conjunto de Ligações *sameAs* Exportadas

As fontes de dados publicadas nos princípios de Dados Interligados, tendem a possui ligações *owl:sameAs* de recursos da sua fonte de dados para recursos de outras fontes de dados. Essas ligações são resultadas de análises realizadas nas fontes de dados, para identificar recursos iguais em outras fontes. O processo de identificação de similaridade de instância é um processo muito caro, pois é necessário fazer a compara entre instância, ou seja, se temos duas bases com mil instâncias, em um processo simples de comparações, teríamos que fazer um milhão de comparações. Isso torna essas ligações importantes, pois além de serem "confiáveis" (normalmente é feito com uma análise criteriosa) e caro obter essa informação.

Em nosso framework, consideramos as ligações *sameAs* existentes entre recursos de diferentes visões exportadas, logo elas também devem ser exportadas para a visão de *mashup*. Para criar as ligações *sameAs* exportadas, o usuário deve primeiro especificar uma visão de ligação *sameAs* exportada e então materializar as ligações.

Seja $(T, S_T, O_{ST}, O_T, \mu_T)$ e $(U, S_U, O_{SU}, O_U, \mu_U)$ duas visões exportadas. Uma especificação de visão de conjunto de ligações *sameAs* exportadas é uma tupla $(IL, T, U, C, C_{ST}, C_{SU})$, onde:

a) *IL* é o nome da visão;
b) $(T, S_T, O_{ST}, O_T, M_T)$ e $(U, S_U, O_{SU}, O_U, M_U)$ são especificações de visões exportadas;
c) *C* é uma classe em ambos os vocabulários das ontologias das visões exportadas $O_T$ e $O_U$;
d) $C_{ST}$ é uma classe no vocabulário da ontologia fonte $O_{ST}$ de $S_T$, tal que existe uma regra em $M_T$, indicando que instâncias de $C_{ST}$ são mapeadas por instâncias de *C*;

e) $C_{SU}$ é uma classe no vocabulário da ontologia fonte $O_{SU}$ de $S_U$, tal que existe uma regra em $M_U$, indicando que instâncias de $C_{ST}$ são mapeadas por instâncias de $C$.

Seja $\boldsymbol{s}_T$ e $\boldsymbol{s}_U$ estados respectivamente de $S_T$ e $S_U$. A materialização de *IL* em $\boldsymbol{s}_T$ e $\boldsymbol{s}_U$ é o conjunto $IL[\boldsymbol{s}_T,\boldsymbol{s}_U]$, definido como:

$(t, owl{:}sameAs, u) \in IL[\boldsymbol{s}_T,\boldsymbol{s}_U]$, se somente se,

$$(t, owl{:}sameAs, u) \in \boldsymbol{s}_T \wedge (t, rdf{:}type, C_{ST}) \in I[s_T](C_{ST}) \wedge (u, rdf{:}type, C_{SU}) \in I[s_U](C_{SU})$$

Isto é, $IL[\boldsymbol{s}_T,\boldsymbol{s}_U]$ importa ligações *sameAs* de $\boldsymbol{s}_T$ cujo o sujeito esteja na interpretação em $\boldsymbol{s}_T$ da classe $C_{ST}$ e cujo objeto esteja na interpretação em $\boldsymbol{s}_U$ da classe $C_{SU}$.

Voltando ao nosso estudo de caso, agora vamos especificar nossas visões dos conjuntos de ligações *sameAs* Exportadas. Considere, a visões exportadas *MusicBrainz_EV* e *DBpedia_EV*. Como mostra na Figura 6 (a), a fonte de dados *MusicBrainz* contém ligações de instâncias correspondentes da classe *mo:Record* com instâncias da classe *dbpedia-owl:Album* da fonte de dados *DBpedia*. A fonte de dados *MusicBrainz*, possui também, ligações *sameAs* de instâncias correspondentes de *mo:MusicArtist* com *dbpedia-owl:MusicalArtist* da fonte de dados *DBpedia*. De modo, que para a materialização das ligações *sameAs*, para as instâncias das visões exportadas *MusicBrainz_EV* e *DBpedia_EV*, duas visões de conjunto de ligações exportadas devem ser especificadas:

a) ($IL_1$, *MusicBrainz_EV*, *DBpedia_EV*, *mo:Record*, *mo:Record*, *dbpedia-owl:Album*);

b) ($IL_2$, *MusicBrainz_EV*, *DBpedia_EV*, *mo:MusicArtist*, *mo:MusicArtist*, *dbpedia-owl:MusicalArtist*).

Referindo a Figura 6 (a) do nosso estudo de caso, a ligação *sameAs* de *db:artist/Michael_Jackson* para *dbpedia:Michael_Jackson* é materializada por $IL_1$, e a ligação *sameAs* de *db:record/Thriller* de *dbpedia:Thriller* é materializada por $IL_2$.

Podemos gerar de forma automática, consultas *SPARQL* para realizar a materialização da visão de conjunto de ligações exportadas. Intuitivamente, temos as seguintes consultas geradas:

Tabela 6 — Modelo de consulta *SPARQL* para materialização da visão de conjunto de ligações *sameAs* exportadas

```
1. CONSTRUCT { ?s owl:sameAs    ?o.
2.             ?s rdf:type       C.
3.             ?o rdf:type       C.}
4. FROM <http://uri_D2>
5. WHERE
```

```
6.  { ?s owl:sameAs    ?o
7.    ?s rdf:type       C_ST.
8.    FILTER regex(str(?o), "http://uri_ D1").}
```

Fonte: Elaborada pelo autor.

Tabela 7 — Consulta SPARQL para materializar $IL_1$*[$s_{MUSICBRAINZ}$[9],$s_{DBPEDIA}$[10]]*

```
1.  CONSTRUCT { ?s owl:sameAs    ?o.
2.              ?s rdf:type       mo:Record.
3.              ?o rdf:type       mo:Record.}
4.  FROM <http://dbpedia.org>
5.  WHERE
6.  { ?s owl:sameAs    ?o
7.    ?s rdf:type       mo:Record.
  FILTER regex(str(?o), "http://dbtune.org/musicbrainz/resource/").}
```

Fonte: Elaborada pelo autor.

Tabela 8 — Consulta SPARQL para materializar $IL_2$*[$s_{MUSICBRAINZ}$,$s_{DBPEDIA}$]*

```
1.  CONSTRUCT { ?s owl:sameAs    ?o.
2.              ?s rdf:type       mo:MusicArtist.
3.              ?o rdf:type       mo:MusicArtist.}
4.  FROM <http://dbpedia.org>
5.  WHERE
6.  { ?s owl:sameAs    ?o
7.    ?s rdf:type       mo:MusicArtist.
8.    FILTER regex(str(?o), "http://dbtune.org/musicbrainz/resource/").}
```

Fonte: Elaborada pelo autor.

## 4.6 Especificação e Materialização de visões de Conjunto de Ligações *sameAs* de *Mashup*

Para os casos em que não for possível obter das fontes de dados as ligações *owl:sameAs*, precisamos calcular a similaridade entre as instâncias, para identificar as possíveis instâncias iguais.

No framework, ligações *sameAs* de *mashup* são inferidas pelos valores das propriedades correspondentes de recursos definidos nas visões exportadas. Para criar ligações *sameAs* de *mashup*, o usuário deve primeiro especificar a visão de conjunto de ligações *sameAs* de *mashup* e então materializar as ligações. Mais precisamente, uma especificação de visão de conjunto de ligações *sameAs* de *mashup* é um tupla $(ML, T, U, C, p_1, \ldots, p_n, \mu)$, onde:

a) *ML* é o nome da visão;

b) $(T, S_T, O_{ST}, O_T, M_T)$ e $(U, S_U, O_{SU}, O_U, M_U)$ são especificações de visões exportadas;

[9] Estado da fonte de dados MusicBrainz no nosso estudo de caso
[10] Estado da fonte de dados DBPedia no nosso estudo de caso

c) $C$ é uma classe em ambos os vocabulários das ontologias das visões exportadas $O_T$ e $O_U$;

d) $p_1$, …, $p_n$, são propriedades da classe $C$ em ambos os vocabulários das ontologias das visões exportadas $O_T$ e $O_U$;

e) $\mu$ é uma relação $2n$, chamada de *match predicate*.

Seja $\boldsymbol{e}_T$ e $\boldsymbol{e}_U$ estados respectivamente de $T$ e $U$. A materialização de $ML$ em $\boldsymbol{e}_T$ e $\boldsymbol{e}_U$ é o conjunto $ML[\boldsymbol{e}_T,\boldsymbol{e}_U]$ definido como:

$(t, owl:sameAs, u) \in ML[\boldsymbol{e}_T,\boldsymbol{e}_U]$, se somente se, existem triplas

$(t, rdf:type, C), (t, P_1, v_1), \ldots, (t, P_n, v_n) \in \boldsymbol{e}_T$ e

$(u, rdf:type, C), (u, P_1, w_1), \ldots, (u, P_n, w_n) \in \boldsymbol{e}_U$, tal que $(v_1, \ldots, v_n, w_1, \ldots, w_n) \in \mu$

Voltando ao nosso estudo de caso, especificamos as visões de conjunto de ligações *sameAs* de *mashup*. Considere as visões exportadas *MusicBrainz_EV* e *DBPedia_EV*, introduzida anteriormente. Então, visões de conjunto de ligações *sameAs* de *mashup*, podem ser especificadas para instâncias correspondentes das classes *mo:Track*, observe que não temos ligações *owl:sameAs* no estado das fontes Figura 6 (a). Como exemplo, considere a especificação de visão *sameAs* de *mashup* para *mo:Track*:

$(ML_1, MusicBrainz\_EV, DBPedia\_EV, mo:Track, dc:title, p)$, onde o predicado de correspondência $p$ é definido como:

$(v_1, w_1) \in p$ iff $\sigma(v_k, w_k) \geq \alpha$, para $k=1$, onde $\sigma$ é o 3-gram distance (KONDRAK, 2005) e $\alpha = 0.5$.

Referindo-se ao nosso estudo de caso, Figura 6 (b) mostra as ligações *sameAs* automaticamente criadas usando $ML_1$.

## 4.7 Especificação de Função de Normalização

A especificação de visão MDI inclui uma função de normalização, denotada $\eta$, que remapeia todas as IRIs nas visões exportadas que são declaradas para representar o mesmo objeto, via ligação *sameAs*, para uma *IRI* canônica alvo. A função de normalização deve satisfazer o seguinte axioma (usando uma notação infixa a para propriedades *sameAs*):

(N1) $\forall x_1 \forall x_2 ( x_1 \; sameAs \; x_2 \Leftrightarrow \eta(x_1) = \eta(x_2) )$

quer dizer que a função de normalização deve remapear para a mesma IRI, duas IRIs $x_1$ e $x_2$, se somente se, eles sejam declarados equivalentes através de uma declaração *sameAs* da forma "$x_1$ *sameAs* $x_2$".

A função de normalização particiona as *IRIs* dos recursos das visões exportadas em conjunto de classes de equivalência. Na materialização da visão MDI, todas as *IRIs* da mesma classe de equivalência são homogeneizadas, agrupando todas as propriedades dessas *IRIs* em uma *IRI* canônica alvo. A *IRI* canônica tem ligação *owl:sameAs* apontando para as *IRIs* originais, que torna possível para aplicações referir de voltar as fontes de dados originais na Web.

Referindo-se ao nosso estudo de caso, as classes de equivalência induzidas pelas ligações *sameAs* na Figura 6 (b) são:
$\varepsilon_1$={*dbpedia:Michael_Jackson*, *db:Artist/Michael_Jackson*}, $\varepsilon_2$=**{***dbpedia:Thriller*, *db:record/Thriller*}, $\varepsilon_3$=**{***dbpedia:BBM*, *db:track/BBM*}, $\varepsilon_4$=**{***dbpedia:WBSS*}, $\varepsilon_5$=**{** *db:track/WBSS*}.

## 4.8 Especificação de Métricas de Avaliação de Qualidade

A definição popular de qualidade é "adequação ao uso" (JURAN, 1974). Portanto, a interpretação de qualidade de alguns itens de dados depende de quem vai usar essa informação, e qual a tarefa para a qual pretendem utilizá-la. Enquanto um usuário pode achar que a qualidade dos dados é suficiente para determinada tarefa, pode não ser suficiente para outra tarefa ou outro usuário. Além disso, a qualidade é geralmente percebida como multifacetada, como a "adequação ao uso" pode depender de diversas dimensões, tais como a exatidão, pontualidade, integridade, pertinência, objetividade, credibilidade, compreensibilidade, consistência, concisão, disponibilidade e verificabilidade (BIZER e CYGANIAK, 2009).

Para a avaliação da qualidade utilizamos as informações de proveniência das triplas. Esses metadados contêm informações sobre a história dos dados, como por exemplo, data da última atualização e fonte de dados, Tabela 9 mostra informações de proveniência das fontes do nosso estudo de caso.

Tabela 9 — Dados de proveniência do estudo de caso

| Proveniência | DBpedia | MusicBrainz |
|---|---|---|
| **Última atualização** | 26/12/2014 | 15/12/2014 |
| **Origem** | DBpedia | MusicBrainz |

Fonte: Elaborada pelo autor.

Um indicador de qualidade é o aspecto de um item de dados ou conjunto de dados que pode dar uma indicação para o usuário de adequação dos dados para algum uso a que se destina (MENDES, MÜHLEISEN e BIZER, 2012). Métrica de avaliação de qualidade e criada para avaliar um indicador de qualidade e para isso usamos uma função de pontuação. Avaliação é feita para um conjunto de triplas e os dados de proveniência são usados para obter informações necessárias para avaliação. O resultado da avaliação são pontuação entre 0 e 1, onde quanto mais próximo do 1 melhor foi avaliado o indicador de qualidade, esse resultado é adicionado a proveniência.

No nosso caso de estudo podemos criar um indicador de qualidade *reputation*, é avaliada a qualidade dos dados com base na origem da fonte de dado, usaremos a informação da origem dos dados (na proveniência) da Tabela 9 e consideraremos a seguinte prioridade: *MusicBrainz* > *DBpedia*. Os dados de origem *MusicBrainz* terão pontuação maior que os de origem *DBpedia*.

No nosso caso de estudo criamos um indicador de qualidade *reputation*, que avaliada a qualidade dos dados com base na origem da fonte de dado, usaremos a informação da origem dos dados (na proveniência) da Tabela 9 e consideraremos a seguinte prioridade: *MusicBrainz* > *DBpedia*. Os dados de origem *MusicBrainz* terão pontuação maior que os de origem *DBpedia*.

## 4.9 Especificação de Assertivas de Fusão

Nesta seção, seja $D$ uma visão LDM, $O_D=(V_D,\Sigma_D)$ a ontologia de $D$, e $\eta$ a função de normalização. Seja $C$ uma classe em $\boldsymbol{V_D}$. Definimos $Props(C, \boldsymbol{V_D})$ = {$P$ / $P$ é uma propriedade em $\boldsymbol{V_D}$ e $C$ é uma subclasse do domínio de $P$}

Nossa abordagem, o usuário é livre para definir como resolver o problema de valores de atributos quando combinamos múltiplas representações do mesmo objeto do mundo real em uma única representação (*IRI* canônica). Essa é especificada com a ajuda das assertivas de propriedades de fusão de dados.

Uma assertiva de propriedade de fusão de dados (APF) para uma propriedade *P* no contexto de uma classe *C* é uma expressão da forma:

$\Psi: P[C] \equiv f/Q$

onde $\Psi$ é o nome da APF, *C* é uma classe em $V_D$, *P* e *Q* são propriedades em *Props*(*C*, $\boldsymbol{V_D}$), e *f* é um símbolo de função de fusão de dados, que denota funções cujo domínio é um conjunto de conjuntos de indivíduos e cujo a imagem é um conjunto de indivíduos..

Assertivas de fusão de dados devem ser consideradas como uma notação abreviada para uma classe de regras de fusão de dados. Um conjunto *A* de assertivas de fusões de dados sobre o vocabulário $\boldsymbol{V_D}$ de *D*, induz um conjunto de regras de fusão para *D* da seguinte forma:

a) Seja *C* uma classe em $\boldsymbol{V_D}$ e *P* seja uma propriedade em *Props*(*C*, $\boldsymbol{V_D}$). Assume que $\Psi: P[C] \equiv f/Q$ está em $\boldsymbol{A}$, e que $S_1, \ldots, S_n$ são todas as visões exportadas de *D* cujo vocabulário contém a classe *C* e a propriedade *Q*. Então, a regra de fusão para *P* no contexto de *C* induzida por *A* é:

$P(x,u) \leftarrow u = f(v/B[x,v])$, onde:

$B[x,v] = ((C(x_1),Q(x_1,v))_{S1}, x = \eta(x_1)); \ldots ;((C(x_n),Q(x_n,v))_{Sn}, x = \eta(x_n))$

O subscrito $S_i$ indica a visão exportada sobre o qual $(C(x_i), Q(x_i,v))$ será avaliado. Intuitivamente, $\{v/B[x,v]\}$ denota o conjunto de indivíduos *v* que satisfazem $B[x,v]$. Então, $f(v/B[x,v])$ denota o conjunto de indivíduos que *f* mapeia para o conjunto $\{v/B[x,v]\}$.

b) Seja *C* uma classe em $\boldsymbol{V_D}$ e *P* uma propriedade em *Props*(*C*, $\boldsymbol{V_D}$). Assume que não há nenhuma assertiva em $\boldsymbol{A}$ para a propriedade *P* no contexto de *C*, e que $S_1, \ldots, S_n$ são todas as visões exportadas de *D* cujo vocabulário contém a classe *C* e a propriedade *P*. Então, a regra de fusão para *P* no contexto de *C* induzida por *A* é:

$P(x,v) \leftarrow ((C(x_1),P(x_1,v))_{S1}, x = \eta(x_1)); \ldots ;((C(x_n),P(x_n,v))_{Sn}, x = \eta(x_n))$

c) Seja *C* uma classe em $\boldsymbol{V_D}$. Assume que $S_1, \ldots, S_n$ são todas as visões exportadas de *D* cujo vocabulário contém a classe *C*. Então, a regra de fusão classe *C* induzida por *A* é:

$C(x) \leftarrow ((C(x_1))_{S1}, x = \eta(x_1)); \ldots ;((C(x_n))_{Sn}, x = \eta(x_n))$.

No nosso caso de estudo, a APF para a propriedade *moa:labelName* no contexto da classe *mo:Record*,

$\Psi$*:moa:labelName*[*mo:Record*]≡*KeepSingleValueByReputation/moa:labelName*

induz a seguinte regra de fusão:

*moa:labelName(x,u)←u=KeepSingleValueByReputation(v/*

*(mo:Record(y),moa:labelName(y,v))$_{DBPedia\_EV}$,x= η(y);*

*(mo:Record(z),(moa:labelName(z,v))$_{MusicBrainz\_EV}$,x= η(z))).*

A regra de fusão cria uma tripla da forma "*(x, moa:labelName, u)*" na materialização da visão LDM tomando *u* como nome com maior reputação para x, maior reputação é obtida pegando maior pontuação da métrica de qualidade *reputation* na proveniência dos dados para materialização das duas visões exportadas, *MusicBrainz_EV* e *DBPedia_EV*.

Quando APF não é especificada para propriedade *moa:labelName* no contexto da classe *mo:Record*, então a regra de fusão padrão:

*moa:labelName(x,v)←(mo:Record(y),moa:labelName(y,v))$_{DBPedia\_EV}$,x= η(y));*

*(mo:Record(z),(moa:labelName(z,v))$_{MusicBrainz\_EV}$,x= η(z)))*

é induzida. A regra de fusão cria uma tripla para todos os valores de *moa:labelName* vindo das visões exportadas.

## 4.10 Materialização de Visão de *Mashup* de Dados Interligados

Nesta seção, descrevemos com materializar uma visão MDI através da aplicação da função de normalização e regras de fusão de dados para a materialização das visões exportadas e materialização das visões dos Conjuntos de Ligações *sameAs*. O processo materialização inclui a combinação e a fusão de múltiplas representações do mesmo objeto do mundo real para uma representação única e a resolução das inconsistências de dados (BLEIHOLDER e NAUMANN, 2008).

Iniciamos introduzindo a notação que é usada nesta seção:

a) $\lambda = (D, O_D, E1,..., En , IL1 ,..., ILp , ML1,..., MLq, \mu, \eta, Q)$ é uma especificação da visão MDI;

b) $V_D$ é o vocabulário de $O_D$;

c) $\boldsymbol{C} = \{ C / C$ é uma classe em $V_D\}$;

d) $\boldsymbol{P} = \{ P / P$ é uma propriedade em $V_D\}$;

e) $\boldsymbol{e}_1 ,..., \boldsymbol{e}_n$ são estados de $E_1 ,..., E_n$ e $\boldsymbol{l}_1 ,..., \boldsymbol{l}_m$ são estados de $IL_1 ,..., IL_p, ML_1,..., ML_q$, com $m=p+q$;

f) $\boldsymbol{s} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, \boldsymbol{l}_1, \ldots, \boldsymbol{l}_m)$;

g) *IRIs(**s**)* = { *x* / *x* é o sujeito de uma tripla em tripleset ***s*** };

h) $\boldsymbol{\varepsilon}_{1,\ldots,}\boldsymbol{\varepsilon}_w$ são classes de equivalência induzidas por $\boldsymbol{l}_1, \ldots, \boldsymbol{l}_m$;

i) $[\boldsymbol{\varepsilon}_i]$ indica a *IRI* que representa $\boldsymbol{\varepsilon}_i$;

j) $\eta^s$ indica a interpretação de $\eta$ para $\boldsymbol{\varepsilon}_{1,\ldots,}\boldsymbol{\varepsilon}_w$;

k) *I[**e**](T)* indica o conjunto de triplas que o estado ***e*** atribui para um classes ou propriedade *T*, ou seja, a interpretação de *T* em ***e***.

O estado ***s*** induz um estado ***d*** de *D* da seguinte forma:

a) Seja *C* uma classe em $V_D$. Suponha que $\mu$ tem uma regra de fusão cuja cabeça usa *C* e que a regra é da forma:

$C(x) \leftarrow (C(x_1)_{S1}, x=\eta(x_1)); \ldots ;(C(x_p)_{Sq}, x=\eta(x_n))$

Então, a interpretação de C induzida por $\mu$ *em* ***s***, também indica *I[**s**](C)*, é definida como:

$(x, rdf{:}type, C) \in I[\boldsymbol{s}](C)$ iff $(x, rdf{:}type, C) \in I[\boldsymbol{e_1}](C), x=\eta^s(x_1)) \vee \ldots \vee (x, rdf{:}type, C) \in I[\boldsymbol{e_n}](C), x=\eta^s(x_n))$

b) Seja *P* uma propriedade em $V_D$. Suponha que $\mu$ tem uma regra de fusão cuja cabeça usa *P* e que a regra é da forma:

$P(x,v) \leftarrow (C(x_1),P(x_1, v))_{S1}, x=\eta(x_1)); \ldots ;((C(x_n), P(x_n, v))_{Sn}, x=\eta(x_n))$

Então, a interpretação de *P* induzida por $\mu$ em ***s***, também indica *I[**s**](P)*, é definida como:

$(x, P, v) \in I[\boldsymbol{s}](P)$ iff $((x, rdf{:}type, C) \in I[\boldsymbol{e_1}](C), (x,P,v) \in I[\boldsymbol{e_1}](P), x=\eta^s(x_1)) \vee \ldots \vee ((x, rdf{:}type, C) \in I[\boldsymbol{e_n}](C), (x,P,v) \in I[\boldsymbol{e_n}](P), x=\eta^s(x_n))$

c) Seja *P* uma propriedade em $V_D$. Suponha que $\mu$ tem uma regra de fusão cuja cabeça usa *P* e que a regra é da forma:

$P(x, u) \leftarrow u = f(v/B[x,v])$, onde:

$B[x,v]=((C(x_1),Q(x_1,v))_{S1}, x=\eta(x_1)); \ldots ;((C(x_n),Q(x_n,v))_{Sn}, x=\eta(x_n))$

Então, a interpretação de *P* induzida por $\mu$ em ***s***, também indica *I[**s**](P)*, é definida como:

$(x, P, u) \in I[\boldsymbol{s}](P)$ sse $u = I[\boldsymbol{s}](f)(\{v / I[\boldsymbol{s}](B[x,v])=true\})$

d) Se houver mais de uma regra de fusão em $\mu$ cuja cabeça usa *P*, então *I[**s**](P)* é a união de todos conjuntos de triplas, como definido no lado direito da implicação dupla acima.

e) Lembre-se que $\boldsymbol{C}$ é o conjunto de classes em $V_D$ e $\boldsymbol{P}$ é o conjunto de todas propriedades em $V_D$ e $\boldsymbol{\varepsilon}_i$ é uma classe de equivalência induzida por $\boldsymbol{l}_{1,\ldots,}\ \boldsymbol{l}_m$. Definimos a função *DataFusion* da seguinte forma:

$$DataFusion(\boldsymbol{\varepsilon}_i, \boldsymbol{s}) = \cup_{C \in C}\{\ (x, rdf{:}type, C) \in I[\boldsymbol{s}](C) / x = [\boldsymbol{\varepsilon}_i]\ \} \cup$$
$$\cup_{P \in P}\{\ (x, P, y) \in I[\boldsymbol{s}](P) / x = [\boldsymbol{\varepsilon}_i]\ \} \cup$$
$$\cup\ \{\ ([\boldsymbol{\varepsilon}_i], owl{:}sameAs, y) / y \in \boldsymbol{\varepsilon}_i \wedge y \neq [\boldsymbol{\varepsilon}_i]\ \}$$

f) Definimos também o estado $\boldsymbol{d}$ ou a materialização de *D* induzida por *s* como (lembre que $\boldsymbol{\varepsilon}_{1,\ldots,}\boldsymbol{\varepsilon}_w$ são classes de equivalência induzidas por $\boldsymbol{l}_{1\ ,\ldots,}\ \boldsymbol{l}_m$):

$$\boldsymbol{d} = \cup_{i=1,..,w}\ DataFusion(\boldsymbol{\varepsilon}_i, \boldsymbol{s})$$

No estudo de caso, Figura 6(c) mostra o estado da visão de *mashup* calculada do estado das visões exportadas e visões de conjuntos de ligações *sameAs* em Figura 6(b). A visão do *mashup* tem 5 recursos que são calculados pela aplicação da função *DataFusion* para classes de equivalência $\boldsymbol{\varepsilon}_{1,\ldots,}\boldsymbol{\varepsilon}_5$.

## 4.11 Conclusão

Neste capítulo definimos as especificações das visões exportadas, visões de conjunto de ligação *sameAs* exportadas, visões de conjunto de ligação *sameAs* de *mashup*, função de normalização, métricas de avaliação de qualidade e as assertivas de fusão de dados, que são fundamentais para gerar o processo de geração da visão MDI.

# 5 MANUTENÇÃO INCREMENTAL DE VISÕES MDI

## 5.1 Introdução

Nesse Capítulo, apresentamos uma estratégia de manutenção incremental para visão MDI, quando operações de atualização são aplicadas as fontes de Dados Interligados. Neste Capítulo, seja:

a) $\lambda = (D, O_D, E_1, ..., E_n, IL_1, ..., IL_p, ML_1, ..., ML_q, \mu, \eta, Q)$, uma especificação de visão MDI $D$;

b) $\boldsymbol{e}_1, ..., \boldsymbol{e}_n$ são estados de $E_1, ..., E_n$ e $\boldsymbol{l}_1, ..., \boldsymbol{l}_m$ são estados de $IL_1, ..., IL_p, ML_1, ..., ML_q$, com $m=p+q$;

c) $\boldsymbol{s} = (\boldsymbol{e}_1, ..., \boldsymbol{e}_n, \boldsymbol{l}_1, ..., \boldsymbol{l}_m)$;

d) $\boldsymbol{d}$ o estado de $D$ induzido por $\boldsymbol{s}$;

e) $u$ uma atualização contra uma fonte de dados $\boldsymbol{S_i}$;

f) $\boldsymbol{e'}_1, ..., \boldsymbol{e'}_n$ o novo estado das visões exportadas $E_1, ..., E_m$ e $\boldsymbol{l'}_1, ..., \boldsymbol{l'}_m$ o novo estado das visões sameAs $IL_1, ..., IL_p, ML_1, ..., ML_q$, com $m=p+q$, imediatamente após $u$;

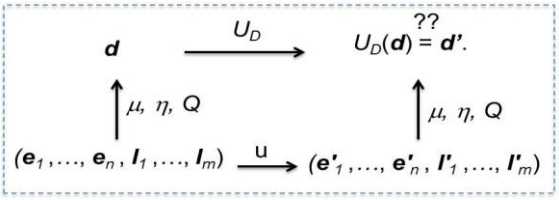
g) $\boldsymbol{t} = (\boldsymbol{e'}_1, ..., \boldsymbol{e'}_n, \boldsymbol{l'}_1, ..., \boldsymbol{l'}_m)$;

h) $\boldsymbol{d'}$ o estado de $D$ induzido por $\boldsymbol{t}$.

## 5.2 Problema da Manutenção Incremental da Visão MDI

O problema da manutenção incremental da visão é esquematicamente descrito pelo diagrama na Figura 12. Seja $\boldsymbol{d'}$ o estado de $D$ induzido por $\boldsymbol{t}$. Dizemos que o conjunto de atualizações $U_D$ sobre o estado $\boldsymbol{d}$ mantém corretamente $D$ se somente se $U_D(\boldsymbol{d}) = \boldsymbol{d'}$.

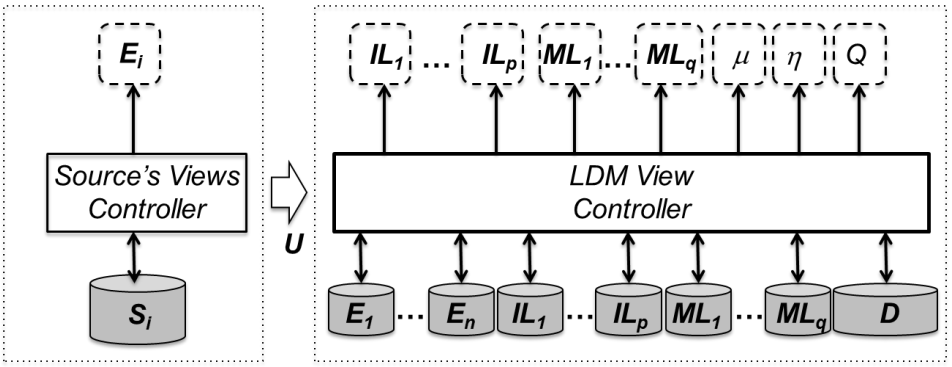
Figura 12 — Problema da Manutenção Incremental da Visão




Fonte: Elaborada pelo autor.

## 5.3 Plataforma para manutenção da Visão MDI

Figura 13 — Sugestão de plataforma para manutenção da visão MDI




Fonte: Elaborada pelo autor.

Na nossa estratégia de manutenção da visão MDI, as visões exportadas, as visões de conjunto de ligações *sameAs* exportados e as visões de conjunto de ligações *sameAs* de *mashup* são materializadas, e serão utilizadas para a manutenção incremental da visão MDI.

Figura 13 mostra os principais componentes da arquitetura que sugerimos para manter incrementalmente a visão MDI *D*. Para cada fonte de dados $\boldsymbol{S_i}$ que exporta uma visão $\boldsymbol{E}_i$ para *D*, existe um *Source View Controller*, com as seguintes funcionalidades:

a) Identificar atualizações em $\boldsymbol{S_i}$ que são relevantes para *D*, isto é, relevantes para uma ou mais visões exportadas, visões de conjunto de ligações *sameAs* exportadas ou visões de conjunto de ligações *sameAs* de *mashup* usado para construir D. Isto é calculado para as especificações das visão exportadas, visões de conjunto de ligações *sameAs* exportadas ou visões de conjunto de ligações *sameAs* de *mashup*. No que segue, assumimos que:
   – Somente uma visão é afetada por *u* e esta é uma visão exportada, que indicamos por $\boldsymbol{E_i}$.

b) Para cada atualização relevante *u*, cria o conjunto: $\boldsymbol{U}$ = {<*r*, $\psi_r$> / *r* é a *IRI* de um recurso afetado por *u* e $\psi_r$, são estados de *r* no estado novo da visão exportada $\boldsymbol{E_i}$};
c) Envia $\boldsymbol{U}$ para a *LDM View Controller*.

Note que $\boldsymbol{U}$ pode ser calculado automaticamente baseado na especificação da visão exportada, o estado novo e antigo de S e a atualização base u. Por isso, não é requerido o acesso à visão de *mashup*. O problema de calcular $\boldsymbol{U}$ está fora do escopo deste trabalho. Um problema similar é abordado em (VIDAL, CASANOVA e CARDOSO, 2013).

A *LDM View Controller* recebe $\boldsymbol{U}$ e então realiza a manutenção incremental da visão exportada $E_s$ e das visões *sameAs* $IL_1, \ldots, IL_p$, $ML_1, \ldots, ML_q$, seguindo pela manutenção incremental da visão de *mashup* D.

O problema da Manutenção Incremental de visões exportadas é muito parecido com o problema abordado em (VIDAL, CASANOVA e CARDOSO, 2013). Portanto, não abordaremos esses problemas neste trabalho.

A manutenção incremental das visões *sameAs* foi abordada no trabalho (CASANOVA, VIDAL, *et al.*, 2014). Portanto, não abordaremos esses problemas neste trabalho.

## 5.4 Manutenção Incremental de Visões de *Mashup*

A fim de realizar a manutenção incremental da visão de *mashup D*, a *LDM View Controller* executa o procedimento *Effect*(r) para cada par $\boldsymbol{p}$=<$\boldsymbol{r}$, $\boldsymbol{\psi_r}$ > em $\boldsymbol{U}$ (veja Tabela 10). *Effect*(r) aplica, para o corrente estado da visão de *mashup*, as atualizações necessárias para manter *D* considerando ($\boldsymbol{e'}_1, \ldots, \boldsymbol{e'}_n$, $\boldsymbol{l'}_1, \ldots, \boldsymbol{l'}_m$) o novo estado das visões exportadas e visões *sameAs*.

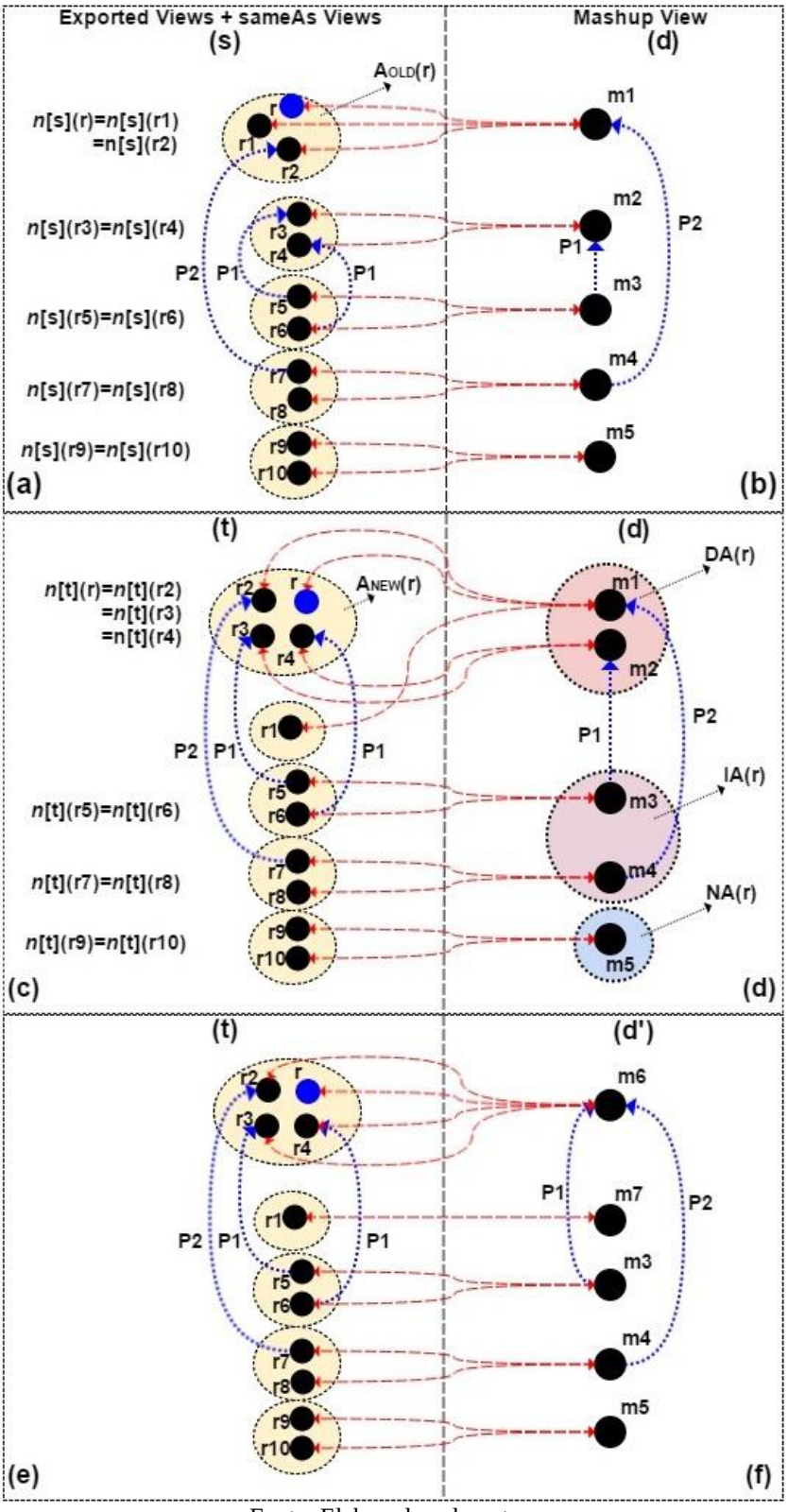
Em nossa estratégia de manutenção incremental da visão de *mashup*, primeiro identificamos os recursos da visão de *mashup* que podem ser afetados pela atualização de $\boldsymbol{r}$, em seguida, calculamos o novo estado dos recursos afetados considerando o novo estado das visões exportadas e visões *sameAs*.

Depois, discutimos como calcular o conjunto de recursos na visão de *mashup* que podem ter sido afetados pelo novo estado de r, em seguida discutiremos como atualizar o

estado dos possíveis recursos afetados, a fim de torná-los consistentes com o novo estado das visões exportadas e visões *sameAs* (***e'***$_{1,\ldots,}$***e'***$_{n}$, ***l'***$_{1}$ ,…, ***l'***$_{m}$).

### *5.4.1 Calculando os Recursos Afetados*

Figura 14 — Atualização da visão MDI




Fonte: Elaborada pelo autor.

Na Figura 14, em (a), (c) e (e) temos os estado das visões exportadas onde os recursos estão agrupados por classes de equivalência, note que em (a) as classes de equivalência são calculadas com base no estado ***s*** (antes da atualização) e em (c) e (e) as classes de equivalência são calculadas com base no estado ***t*** (depois da atualização). Em (b), (d) e (f) temos o estado da visão MDI, onde cada recurso é gerado por uma classe de equivalência de (a), (c) e (e). Em (b) e (d), temos o estado da visão antes da atualização da visão, (f) temos o estado da visão após a atualização da visão.

Lembre-se que:

a) ***s*** o estado das visões exportadas e das visões *sameAs*;
b) ***d*** o estado da visão de *mashup D* induzida por ***s***;
c) ***u*** uma atualização contra uma fonte de dados $\boldsymbol{S_i}$;
d) ***t*** o estado das visões exportadas e das visões *sameAs* imediatamente após uma atualização;
e) ***d'*** o estado de *D* induzido por ***t***;
f) Seja $<\boldsymbol{r}, \boldsymbol{\psi_r}>$ um par em ***U***.

As Definições 5.1, 5.2 e 5.3 indica quais recursos são afetados pelo novo estado do recurso *r*.

**Definição 5.1**: O conjunto de recursos nas visões exportadas que são *diretamente afetados* por ***r***, é:

$\boldsymbol{A(r) = A_{OLD}(r) \cup A_{NEW}(r)}$, onde:

- $\boldsymbol{A_{OLD}(r)} = \{ x / y \in IRIs(s) \wedge \eta[s](\boldsymbol{x}) = \eta[s](\boldsymbol{r}) \}$;
- $\boldsymbol{A_{NEW}(r)} = \{ x / x \in IRIs(t) \wedge \eta[t](\boldsymbol{x}) = \eta[t](\boldsymbol{r}) \}$.

Porém, depois da atualização, *IRIs(s)* e $\eta[s]$ não estão mais disponíveis.

Note que $\eta[\boldsymbol{s}](x) = \eta[\boldsymbol{s}](r)$, se somente se, $(\exists y \in IRIs(\boldsymbol{d})) \wedge ( (y, owl{:}sameAs, r) \in \boldsymbol{d} \wedge (y, owl{:}\ sameAs, x) \in \boldsymbol{d} )$

Portanto, podemos definir por:

$\boldsymbol{A_{OLD}(r)} = \{x / (\exists y \in IRIs(\boldsymbol{d})) \wedge (y, owl{:}sameAs, r) \in d \wedge (y, owl{:}sameAs, x) \in d\}$;
Intuitivamente, temos:

- $x \in \boldsymbol{A_{OLD}(r)}$, se somente se, *x* pertence à classe de equivalência de ***r*** (no estado antigo ***s***). Se ***r*** não está em ***s*** (***r*** é um novo recurso), então $\boldsymbol{A_{OLD}(r)} = \varnothing$.
- $x \in \boldsymbol{A_{NEW}(r)}$, se somente se, $x \in IRIs(t)$ e *x* pertence à classe de equivalência de ***r*** (no estado novo ***t***). Se ***r*** não está em ***t***, então $\boldsymbol{A_{NEW}(r)} = \varnothing$.

Como exemplo, suponha os recursos em ***s*** são particionados em cinco classes de equivalência como mostra Figura 14 (a). Figura 14 (b) mostra os recursos em ***d*** representando cada classe de equivalência de ***s***. *IRIs*(***d***) ={$m_1$, $m_2$, $m_3$, $m_4$, $m_5$}. Figura 14 (c) mostra as novas classes de equivalência para o novo estado ***t***. Na Figura 14 (a)(b), temos que *(m1, owl:sameAs, r)*, *(m1, owl:sameAs, $r_1$)* e *(m1 owl:sameAs, $r_2$)*. Portanto,

$\boldsymbol{A_{OLD}(r)}$={ $r$, $r_1$, $r_2$}

Considerando as novas classes de equivalência para ***t***, mostrado na Figura 14 (c), temos:

$\boldsymbol{A_{NEW}(r)}$={ $r$, $r_2$, $r_3$, $r_4$}

Portanto, pela Definição 5.1, temos ***A***(***r***)={$r$, $r_1$, $r_2$, $r_3$, $r_4$}.

**Definição 5.2**: O conjunto de recursos na visão de *mashup* ***d*** que são *diretamente afetados* por ***r***, é:

$$\boldsymbol{DA(r)} := \{ x / x \in IRIs(d) \wedge y \in \boldsymbol{A(r)} \wedge (x, owl:sameAs, y) \in d\}$$

Intuitivamente, temos:

- $x \in \boldsymbol{DA(r)}$, se somente se, $x \in d$ e $x$ tem uma ligação *sameAs* para um recurso em ***A***(***r***).

Como exemplo, considere os recursos do *mashup* em ***d*** (veja Figura 14 (b) e Figura 14 (d)) e *A*(*r*), temos:

$\boldsymbol{DA(r)}$={$m_1$, $m_2$}

**Definição 5.3**: O conjunto de recursos na visão de *mashup* ***d*** que são *indiretamente afetados* por ***r***, é:

$$\boldsymbol{IA(r)} := \{x / (x, P, y) \in d \wedge x \notin \boldsymbol{DA}(r) \wedge y \in \boldsymbol{DA}(r)\}$$

Intuitivamente, temos:

- $x \in \boldsymbol{IA(r)}$, se somente se, $x \in d$ e $x$ está relacionado com um recurso diretamente afetado por ***r***.

Como exemplo, considere os recursos do *mashup* em d (veja Figura 14 (d)) e DA(r), temos:

$\boldsymbol{IA(r)}$={$m_3$, $m_4$}

**Definição 5.4**: O conjunto de recursos na visão de *mashup* ***d*** que *não são afetados* por r, é:

$$\boldsymbol{NA(r)} := \{x \in \boldsymbol{IRIs(d)} / x \notin \boldsymbol{DA}(r) \wedge x \notin \boldsymbol{IA}(r) \}.$$

**Teorema 5.1**: Se $x \in \boldsymbol{NA}(\boldsymbol{r})$, então $DataFusion(x,\boldsymbol{s}) = DataFusion(x,\boldsymbol{t})$.

Portanto, recursos em $\boldsymbol{d}$ que pertençam ao conjunto $\boldsymbol{NA}(\boldsymbol{r})$, não precisam ser recomputados. Ou seja, precisamos recalcular apenas o estado dos recursos em $\boldsymbol{d}$ que pertençam ao conjunto $\boldsymbol{DA}(r)$ ou ao conjunto $\boldsymbol{IA}(r)$.

Como exemplo, considere o estado $\boldsymbol{d}$ (veja Figura 14 (d)) e $IA(r)$ e $DA(r)$, temos: $\boldsymbol{NA}(\boldsymbol{r})=\{m_5\}$

### *5.4.2 Atualizando os Recursos Afetados*

Nesta seção, mostraremos o processo de atualização dos recursos afetados. No Teorema 5.2, mostramos que a visão MDI é atualizada corretamente.

O procedimento ***Effect*** na Tabela 10, calcula as atualizações da manutenção da visão de *mashup* em duas etapas:

Etapa 1: Remove $\boldsymbol{\Delta_1}$, o estado antigo dos objetos da visão de *mashup* para os recursos em $\boldsymbol{DA}(r)$; e adiciona $\boldsymbol{\Delta_2}$, o estado novo dos objetos da visão *mashup* para os recursos em $\boldsymbol{A}(r)$.

Considerando o estado $\boldsymbol{d}$ e $\boldsymbol{A(r)}$ na Figura 14 (d), temos que:

$\boldsymbol{\Delta_1} = I[\boldsymbol{d}](\boldsymbol{m}_1) \cup I[\boldsymbol{d}](\boldsymbol{m}_2)$ e
$\boldsymbol{\Delta_2} = DataFusion(\varepsilon_1, \boldsymbol{t}) \cup DataFusion(\varepsilon_2, \boldsymbol{t})$, onde:
$\varepsilon_1 = \{r, r_2, r_3, r_4\}$ e $\varepsilon_2 = \{r_1\}$.
$\boldsymbol{d} = (\boldsymbol{d} - \boldsymbol{\Delta_1}) \cup \boldsymbol{\Delta_2}$;

Etapa 2: Atualiza os conjuntos de triplas afetadas pela atualização na Etapa 1.

Considerando o estado $\boldsymbol{d}$ e $\boldsymbol{A(r)}$ na Figura 14 (d), temos que:

$\boldsymbol{\Delta_3} = \{(\boldsymbol{m}_3, P_1, \boldsymbol{m}_2), (\boldsymbol{m}_4, P_2, \boldsymbol{m}_1)\}$
$\boldsymbol{\Delta_4} = \{(\boldsymbol{m}_3, P_1, \boldsymbol{m}_6), (\boldsymbol{m}_4, P_2, \boldsymbol{m}_6)\}$
$\boldsymbol{d} = (\boldsymbol{d} - \boldsymbol{\Delta_3}) \cup \boldsymbol{\Delta_4}$;

Para provar que o algoritmo mantém corretamente a visão, devemos mostrar que o estado novo dos recursos afetados é consistente com o novo estado das visões *sameAs* e visões exportadas (Teorema 5.2 abaixo)

**Teorema 5.2**: Seja $\boldsymbol{d'}$ o estado de $D$ induzido por $(\boldsymbol{e'}_{1,\ldots,}\boldsymbol{e'}_n, \boldsymbol{l'}_1, \ldots, \boldsymbol{l'}_m)$ e $\boldsymbol{d''}$ o estado de $D$ resultado da aplicação $Effect(\boldsymbol{r})$. Então, $\boldsymbol{d' = d''}$.

Tabela 10 — Procedimento *Effect*(r)

*Parâmetros (*conforme definido acima)*:*

$\lambda$, a especificação da visão MDI

$\boldsymbol{t} = (\boldsymbol{e'}_1, \ldots, \boldsymbol{e'}_n, \boldsymbol{l'}_1, \ldots, \boldsymbol{l'}_m)$

***d***, o estado da visão de *mashup D* obtida pela aplicação das regras de fusão a ***s***

*Entrada*: IRI *r*

**Etapa 1**. Computa os novos estados dos recursos diretamente afetados pelo novo estado de r.

1.1 Computa ***A*** conjunto de recursos diretamente afetados:

$\boldsymbol{A_1}$ := { $x$ / $x \in IRIs(t) \wedge y \in IRIs(d) \wedge (y, owl{:}sameAs, r) \wedge (y, owl{:}\ sameAs, x) \in d$ };

$\boldsymbol{A_2}$ := { x / $x \in IRIs(t) \wedge \eta^t(\boldsymbol{x}) = \eta^t(\boldsymbol{r})$ }

$\boldsymbol{A} := \boldsymbol{A_1} \cup \boldsymbol{A_2}$;

1.2. Computa $\varepsilon_1, \ldots, \varepsilon_n$ as novas classes de equivalência induzidas pelos links *sameAs* em $\boldsymbol{l'}_1, \ldots, \boldsymbol{l'}_m$ para os recursos em ***A***;

1.3. Computa $\boldsymbol{\Delta_1}$ os novos estados dos objetos de *mashup* para os recursos em ***A***;

$\boldsymbol{\Delta_1} := \cup_{i=1,..,m}\ DataFusion(\varepsilon_i, \boldsymbol{t})$;

1.4. Computa $\boldsymbol{\Delta_2}$ os estados antigos dos objetos de *mashup* para os recursos em ***A***;

$\boldsymbol{\Delta_2}$ := {$(x, P, z)$ / (x, P, z) $\in$ d $\wedge$ $y \in \boldsymbol{A} \wedge (x, owl{:}sameAs, y) \in d$ };

1.5. $\boldsymbol{d} = (\boldsymbol{d} - \boldsymbol{\Delta_2}) \cup \boldsymbol{\Delta_1}$;

**Etapa 2**. Computa os novos estados das triplas (x, P, y) em ***d*** tal que *y* seja afetado pelas atualizações na etapa 1.5.

2.1. Computa ***B*** o conjunto de triplas afetadas afetados pelas atualizações na etapa 1.

$\boldsymbol{B}$ := {(x, P, y) / (x, P, y) $\in d \wedge (y, owl{:}sameAs, z) \in d \wedge z \in \boldsymbol{A}$(r) $\wedge\ y \neq \eta^t(\boldsymbol{z})$}.

2.2. Computa $\boldsymbol{\Delta_3}$ os novos estados dos triplas em ***B***;

$\boldsymbol{\Delta_3} := \emptyset$;

Para cada $(x, P, y)$ em ***B*** faça {

Seja *Q* uma propriedade das visões exportadas usadas na regra de fusão de P.

**w** : = {$(u, Q, z)$ / $(u, Q, z) \in t \wedge (x, owl{:}sameAs, u) \in d$ };

$\boldsymbol{\Delta_3} := \boldsymbol{\Delta_3} \cup \{(x, P, q) / (x, P, q) \in I[\boldsymbol{w}](P)\};\}$

2.3. $\boldsymbol{d} := \boldsymbol{d} - \boldsymbol{B} \cup \boldsymbol{\Delta_3}$;

Fonte: Elaborada pelo autor.

Para ilustrar essa estratégia, seja *u* a seguinte atualização em DBpedia:

```
1. WITH <http://dbpedia.org>
2. DELETE { ?x dc:title “W. B. S. S.” }
3. INSERT { ?x dc:title “Wanna Be Startin' Somethin'” }
4. WHERE{?x rdfs:type dbpedia-owl:Single.?x dc:title “W.B.S.S”}
```

**Fase 1:** (Executado pela View Controller da *DBpedia*)

**Step 1:** Atualização *u* é relevante para classe *mo:Track* da visão exportada *DBpedia_EV*.

**Step 2:** Considerando o estado da DBpedia na Figura 6(a), a atualização *u* altera a propriedade dc:title da instance ***dbpedia:WBSS*** para “Wanna Be Startin' Somethin'”.

**Step 3:** Envia ***R***={dbpedia:WBSS, $\psi_{\text{dbpedia:WBSS}}$ } para a *LDM View Controller*.

**Fase 2:** (Executado pela LDM View Controller)

**Step 1:** Atualiza a visão exportada *DBpedia_EV* e a visão de conjunto de ligações *sameAs ML1*. Baseado na especificação ML1, os recursos *dbpedia:WBSS* e *db:track/WBSS* são calculados como equivalente. Portanto, uma nova ligação (*dbpedia:WBSS, owl:sameAs, db:track/WBSS*)}é adicionada a ML1.

**Step 2** O procedimento *Effect*(*dbpedia:WBSS*) é executado como segue:

**Step 2.1:** Atualiza os recursos diretamente afetados pela atualização de *dbpedia:WBSS*

***A*={*dbpedia:WBSS, db:track/WBSS*}** (recursos diretamente afetados)

***ε*$_1$={*dbpedia:WBSS, db:track/WBSS*}** (novas classes de equivalência)

**Δ$_1$** é obtida da fusão dos recursos em ***ε*$_1$.**

**Δ$_2$** contém todas as triplas em ***d***, relativo ao antigo estado dos recursos em ***A***.

***d*** = (***d*** - **Δ$_2$**) ∪ **Δ$_1$**.

**Step 2.2**.Atualiza as triplas afetadas pelas atualizações em step 2.1.

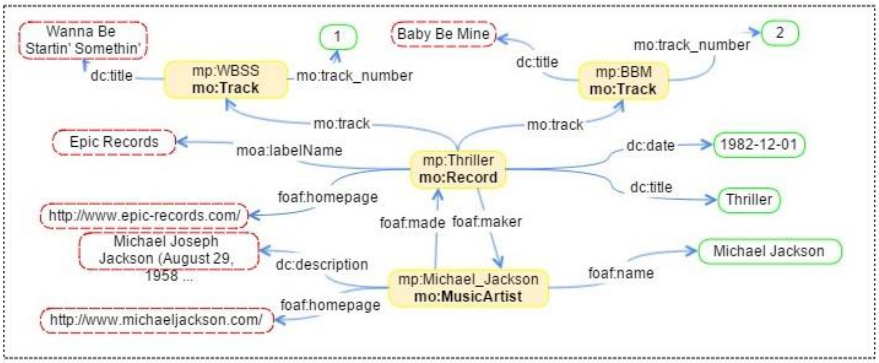
***B***={(*mp:Thriller,mo:track,mp:DB_WBSS*), (*mp:Thriller,mo:track,mp:MB_WBSS*)}

**Δ$_3$**={(*mp:Thriller, mo:track, mp:WBSS*)}

***d*** := ***d*** – ***B*** ∪ **Δ$_3$.**

O estado novo da visão de *mashup* é mostrado na Figura 15. No step 2.1, os antigos estados do recuses afetados em **Δ$_2$** são removidos, e o novo estado dos recursos em **Δ$_1$** são inseridos. No Step 2.2, as triplas afetadas em ***B*** são removidas, e as novas triplas em **Δ$_4$** são inseridas.

Figura 15 — Estado novo da visão do *mashup*




Fonte: Elaborada pelo autor.

## 5.5 Conclusão

Neste Capítulo mostramos como aplicar a estratégia de manutenção incremental para manter atualizada a visão MDI. Mostramos como o processo pode ser feito de forma automática usando as especificações das visões exportadas, visões *sameAs*, função de normalização, métricas de avaliação de qualidade e as assertivas de fusão. No fim, mostramos os Teorema 5.1 e Teorema 5.2, para provar que nossa estratégia mantém corretamente a visão MDI.

# 6 CONCLUSÃO

Neste trabalho, primeiro propomos um framework baseado em ontologia para especificar visões de *Mashup* de Dados Interligados. No framework, uma visão de *mashup* de Dados Interligados é especificada formalmente como a ajuda das visões exportadas, visões dos conjuntos de ligações *sameAs*, funções de normalização, métricas de avaliação de qualidade e funções de fusão de dados. A especificação da visão MDI é usada para automaticamente materializar a visão de *mashup*. Em seguida, traçamos uma estratégia que usa a especificação da visão de *mashup* para manter incrementalmente a visão de *mashup*.

Nossa estratégia abordou o problema de lidar com as mudanças no conjunto de ligações *sameAs* entre IRIs de diferentes fontes de dados e com a questão de recalcular os valores das propriedades de *mashup*, na presença de atualizações sobre as fontes de dados.

## 6.1 Trabalhos Futuros

Implementação da ferramenta para realizar a fusão dos dados, no processo de manutenção incremental de visões MDI.

Implementação da ferramenta responsável pela atualização da visão MDI, recebendo as atualizações das visões exportadas e visões *sameAs* realizar a manutenção na visão de *Mashup*.

# REFERÊNCIAS

ABITEBOUL, S. et al. Incremental Maintenance for Materialized Views over Semistructured Data. **Proceedings of the 24rd International Conference on Very Large Data Bases**, 24-27 August 1998. 38-49.

ALOE - Agile Knowledge Management and Semantic Web (AKSW). **AKSW**. Disponível em: http://aksw.org/Projects/ALOE.html. Acesso em: 10 jan. 2015.

BERNERS-LEE, T. Linked Data. **W3C**, 1998. Disponível em: http://www.w3.org/DesignIssues/LinkedData.html. Acesso em: 10 jan. 2015.

BERNERS-LEE, T. Linked Data. **W3C**, 2006. Disponível em: http://www.w3.org/DesignIssues/LinkedData.html. Acesso em: 10 jan. 2015.

BERNERS-LEE, T.; FIELDING, R.; MASINTER, L. RFC 3986 – Uniform Resource Identifier(URI): Generic Syntax., 2005. Disponível em: http://tools.ietf.org/html/rfc3986. Acesso em: 15 dez. 2014.

BIZER, C. et al. DBpedia - A crystallization point for the Web of Data, 2009. 154–165.

BIZER, C.; CYGANIAK, R. Quality-driven information filtering using the WIQA policy framework. **Web Semant.**, January 2009.

BIZER, C.; HEATH, T.; BERNERS, T. L. Linked data - the story so far. **International Journal on Semantic Web and Information Systems**, 2009. 1-22.

BIZER, C.; JENTZSCH, A.; CYGANIAK, R. State of the LOD Cloud, 2011. Disponível em: http://www4.wiwiss.fu-berlin.de/lodcloud/state/. Acesso em: 15 dez. 2014.

BIZER, C.; SCHULTZ, A. The R2R Framework: Publishing and Discovering Mappings on the Web. **1st International Workshop on Consuming Linked Data (COLD 2010)**, Shanghai, November 2010.

BLEIHOLDER, J.; NAUMANN, F. Data fusion. **ACM Computing Surveys**, 2008. 41(1):1-41.

CASANOVA, M. A. et al. On Materialized sameAs Linksets. **Proc. 25th International Conference on Database and Expert Systems Applications**, Munich, Germany, 8644, 1-5 September 2014. 377-384.

CERI, S.; WIDOM, J. Deriving productions rules for incremental view maintenance. **Proceedings of the 17th International Conference on Very Large Data Bases**, 03-06 September 1991. 577–589.

CYGANIAK, R.; WOOD, D.; LANTHALER, M. RDF 1.1 Concepts and Abstract Syntax. **W3C**, 2014. Disponível em: http://www.w3.org/TR/rdf11-concepts/. Acesso em: 26 jan. 2015.

DIMITROVA, K.; EL-SAYED, M.; RUNDENSTEINER, E. A. **Order-sensitive View Maintenance of Materialized XQuery Views**. [*s.l.*]: [*s.n.*], 2003.

ENDRES, B. N. **Semantic Mashups**. Heidelberg: Springer, 2013.

GIANNOPOULUOS, G. et al. FAGI: A Framework for Fusing Geospatial RDF Data. **The 13th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2014)**, Amantea, Italy, 27-31 October 2014.

GRIFFIN, T.; LIBKIN, L. Algebraic change propagation for semijoin and outerjoin queries. **In SIGMOD Record**, 1998. 27(3):22-27.

GUPTA, A.; MUMICK, I. S. Materialized Views. **MIT Press**, 2000.

HANH, H. H. et al. Semantic Information Integration with Linked Data Mashups Approaches. **International Journal of Distributed Sensor Networks**, 2014.

HEATH, T.; BIZER, C. **Linked Data:** Evolving the Web into a Global Data Space. 1st. ed. ed. [*s.l.*]: Morgan & Claypool, 2011.

JURAN, J. **The Quality Control Handbook**. 3rd edition. ed. New York: McGraw-Hill, 1974.

KNAP, T. . M. J. . D. et al. ODCleanStore: A Framework for Managing and Providing Integrated Linked Data on the Web. **In International Semantic Web Conference (Posters & Demos)**, 2012.

KOBILAROV, G. et al. Media meets semantic web – How the BBC uses DBpedia and Linked Data to make connections. **Proceedings of the 6th European Semantic Web Conference (ESWC 2009)**, Berlin, Heidelberg, 2009. 723–737.

KONDRAK, G. N-gram similarity and distance. **In Proceedings of the Twelfth International Conference on SPIRE**, Buenos Aires, Argentina, 2005. 115–126.

KUNO, H. A.; RUNDENSTEINER, E. A. Incremental Maintenance of Materialized Object-Oriented Views in MultiView: Strategies and Performance Evaluation. **In IEEE TDKE**, 1998. 768–792.

MAGALHÃES, R. P. **UM AMBIENTE PARA PROCESSAMENTO DE CONSULTAS**. UFC. [*s.l.*]. 2012.

MENDES, M.; MÜHLEISEN, H.; BIZER, C. Sieve: Linked Data Quality Assessment and Fusion. **Invited paper at the LWDM 2012**, 2012.

MICHELFEIT, J.; KNAP, T. Linked Data Fusion in ODCleanStore. **In Proceedings of the International Semantic Web Conference (Posters & Demos)**, 2012.

NGONGA, A. C. N. Learning conformation rules for linked data integration. **In Proceedings of Seventh International Workshop on Ontology Matching**, 2012.

NIKOLOV, A. et al. Integration of Semantically Annotated Data by the KnoFuss Architecture. **In Proceedings of EKAW'8**, 2008. 265-274.

RAIMOND, Y. et al. The Music Ontology. **In: International Conference on Music Information Retrieval**, September 2007. 417–422.

SACRAMENTO, E. R. et al. Towards Automatic Generation of Application Ontologies. **JIDM**, 2010. 535-550.

SCHULTZ, A. et al. LDIF - A Framework for Large-Scale Linked Data Integration. **In WWW2012, Developers Track**, 2012.

SCHULTZ, A. et al. LDIF – Linked Data Integration Framework, 2012. Disponível em: http://ldif.wbsg.de/. Acesso em: 26 jan. 2015.

SWARTZ, A. MusicBrainz: A Semantic Web Service. **IEEE Intelligent Systems**, 2002. 76-77.

THE DBpedia Ontology. **DBpedia**, 2014. Disponível em: http://wiki.dbpedia.org/Ontology2014. Acesso em: 15 dez. 2014.

VIDAL, V. M. P. et al. A Mapping-Driven Approach for SQL/XML View Maintenance. **Proceedings of the Tenth International Conference on Enterprise Information Systems**, Barcelona, Spain, 12-16 June 2008. 65-73.

VIDAL, V. M. P. et al. A Framework for Incremental Maintenance of RDF Views of Relational Data. **ISWC - International Semantic Web Conference (Posters & Demos)**, Riva del Garda, Trentino, 21 October 2014. 321-324.

VIDAL, V. M. P.; CASANOVA, M. A.; CARDOSO, D. S. Incremental Maintenance of RDF Views of Relational Data. **Proc. The 12th International Conference on Ontologies, DataBases, and Applications of Semantics**, Graz, Austria, 9-13 September 2013. 572-587.