Pontifícia Universidade Católica
do Rio de Janeiro

Guylerme Velasco de Souza Figueiredo

Bench – Um gerador de dados para testar algoritmos de alinhamento de esquemas conceituais

Dissertação de Mestrado

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

Orientador: Prof. Marco Antonio Casanova

Rio de Janeiro
Dezembro de 2012

Pontifícia Universidade Católica
do Rio de Janeiro

Guylerme Velasco de Souza Figueiredo

Bench – Um gerador de dados para testar algoritmos de alinhamento de esquemas conceituais

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Marco Antonio Casanova
Orientador
Departamento de Informática - PUC-Rio

Prof. Antonio Luz Furtado
Departamento de Informática - PUC-Rio

Profª Karin Koogan Breitman
Departamento de Informática – PUC-Rio

Prof. Luiz André Portes Paes Leme
UFF

Prof. José Eugenio Leal
Coordenador Setorial do
Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 17 de dezembro de 2012

Guylerme Velasco de Souza Figueiredo

Pós-graduado *lato-sensu* em Análise, Projeto e Gerência de Sistemas de Informação pelo Instituto Federal Fluminense (IFF) em Dezembro de 2011. Obteve o Bacharelado em Ciência da Computação pela Universidade Candido Mendes - Campos (UCAM-Campos) em março de 2006. Tem experiência na área de ciência da computação, com ênfase em banco de dados e engenharia de software. Tem trabalhado como analista de sistemas da Petróleo Brasileiro S/A desde 2006.

Ficha Catalográfica

Figueiredo, Guylerme Velasco de Souza

Bench: um gerador de dados para testar algoritmos de alinhamento de esquemas conceituais / Guylerme Velasco de Souza Figueiredo ; orientador: Marco Antonio Casanova. – 2012.
62 f. : il. (color.) ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2012.
Inclui bibliografia

1. Informática – Teses. 2. Alinhamento de esquemas. 3. Benchmark. 4. OWL. I. Casanova, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Para Keila.
Pela paciência e apoio em todos os momentos.

Agradecimentos

Primeiramente, agradeço a Deus, autor da vida. Tudo que sou e tudo que tenho vem de Deus.

Agradeço em especial à minha amada esposa Keila, que esteve ao meu lado em todos os momentos me apoiando, incentivando e torcendo. Seu amor e carinho tem me dado força e sua paciência foi fundamental nos momentos difíceis.

Agradeço aos meus pais pela formação que me proporcionou.

Agradeço ao meu orientador, Professor Marco Antonio Casanova, pelo seu conhecimento técnico, paciência e pela ajuda em todos os momentos que foram necessários.

Ao professor Mark Douglas Jacyntho pela ajuda e apoio para realização deste projeto.

Ao Raphael do Vale pela ajuda no desenvolvimento da pesquisa.

Aos amigos pela força e pelas orações.

À Petrobras, pelo financiamento desta pesquisa.

Resumo

Figueiredo, Guylerme Velasco de Souza: Casanova, Marco Antonio (Orientador). **Bench - Um gerador de dados para testar algoritmos de alinhamento de esquemas conceituais.** Rio de Janeiro, 2012. 62p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação descreve uma ferramenta para teste e avaliação de algoritmos de alinhamento de esquemas a partir da criação de um conjunto de esquemas conceituais, populados com dados. A ferramenta, simplesmente chamada de *Bench*, possibilita a importação de qualquer conjunto de dados com um esquema conceitual bem definido e oferece facilidades para gerar variações deste esquema através de transformações que refletem alternativas de projeto comumente encontradas. Estas transformações de fato definem alinhamentos de referência entre o esquema original e suas variantes. O *Bench* permite ainda calcular o desempenho de um algoritmo de alinhamento de esquemas submetido para testes, comparando os alinhamentos de referência com os alinhamentos encontrados pelo algoritmo em teste.

Palavras-chave

Alinhamento de esquemas; *benchmark*; OWL.

Abstract

Figueiredo, Guylerme Velasco de Souza: Casanova, Marco Antonio (Advisor). **Bench - A tool to generate benchmark data to test conceptual schema alignment algorithms.** Rio de Janeiro, 2012. 62p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This dissertation describes a tool for generate benchmark data to test schema matching algorithms based on the creation of a set of conceptual schemas, populated with data. The tool, simply called *Bench*, allows importing any data set with a well-defined conceptual schema and offers facilities to generate variations of the schema through transformations that reflect structural alternatives found in typical conceptual modeling. Such transformations in fact define reference alignments between the original schema and its variations. *Bench* also permits evaluating the performance of the schema matching algorithm submitted for testing by comparing the reference alignments with those the algorithm under testing is able to find.

Keywords

Schema matching; benchmark; OWL

Sumário

1. INTRODUÇÃO 12

1.1 Motivação 12

1.2 Objetivo 13

1.3 Trabalhos relacionados 14

1.3.1 Alinhamento de esquemas 14

1.3.2 *Benchmarks* 15

1.3.3 OAEI 17

1.3.4 MatchMaking 18

2. FUNDAMENTOS 21

2.1 Alinhamento de esquemas conceituais 21

2.2 Técnicas e Métodos de Alinhamento de Esquemas 22

2.3 RDF / OWL 23

2.4 *Linked Data* 26

3. GERAÇÃO AUTOMÁTICA DE ESQUEMAS SINTÉTICOS 29

3.1 *Benchmarks* 29

3.2 Transformações 29

3.2.1 Cópia 30

3.2.2 Geração de valor constante 30

3.2.3 Particionamento horizontal 31

3.2.4 Associação de Chave Fictícia 31

3.2.5 Particionamento Vertical 32

3.2.6 Desaninhamento (ou Achatamento) 33

3.2.7 Aninhamento 33

3.2.8 Auto-relacionamento 34

3.2.9 Desnormalização 35

3.2.10 Fusão de Chaves e Objetos 35

3.2.11 Manipulando Valores Atômicos 36

3.3 Geração de esquemas sintéticos 37
3.4 Avaliação de Algoritmos de Alinhamento 38

4. *A Ferramenta BENCH* 41
4.1 Requisitos do *Bench* 41
4.2 Arquitetura do *Bench* 42
4.3 Implementação do *Bench* 44
4.3.1 Importação de *datasets* 45
4.3.2 Geração de Esquemas Sintéticos 47
4.3.3 Avaliação de Algoritmos de Alinhamento de Esquemas 50
4.3.4 Camada de Persistência 50
4.4 Exemplo de uso do *Bench* 52

5. CONCLUSÃO 58

6. BIBLIOGRAFIA 60

Lista de imagens

Figura 1 - Conjunto de propriedades de Classe1.propriedade1, Classe2.propriedade2 e Livro.autor. 15
Figura 2 - Classificação das abordagens de alinhamento de esquemas 23
Figura 3 - A "nuvem" *Linking Open Data* 28
Figura 4 - Transformação Cópia 30
Figura 5 - Transformação Geração de Valor Constante 30
Figura 6 - Transformação Particionamento Horizontal e Associação de Chave Fictícia 31
Figura 7 - Transformação Particionamento Vertical 32
Figura 8 - Transformação Desaninhamento 33
Figura 9 - Transformação Aninhamento 34
Figura 10 - Transformação Auto-relacionamento 34
Figura 11 - Transformação Desnormalização 35
Figura 12 - Transformação Fusão de Chaves e Objetos 36
Figura 13 - Transformação Manipulando Valores Atômicos 37
Figura 14 - Comparando alinhamento de esquemas automático 39
Figura 15 - Modelo de Caso de Uso do *Bench* 42
Figura 16 - Divisão da arquitetura do MatchMaking 42
Figura 17 - *Bench* estendendo o *MatchMaking* 43
Figura 18 - Extensão do *MatchMaking* para implementação de comandos 46
Figura 19 - Arquitetura do *Bench* integrada ao *MatchMaking* - Geração de esquemas sintéticos 47
Figura 20 - Representação dos módulos do *Bench* 48
Figura 21 - Camada de persistência do *Bench* 50
Figura 22 - Modelo de Dados do *Bench* 52
Figura 23 - Modelo do eBay 53

Lista de tabelas

Tabela 1 - Características dos casos de teste 18

1. INTRODUÇÃO

1.1 Motivação

Muito se fala nos dias atuais de integração, convergência e centralização. Porém, quando o assunto é dados, observa-se uma grande dificuldade em integrá-los de forma transparente e não traumática. Vemos ocorrências como esta quando instituições financeiras se fundem, pois é necessário integrar suas carteiras de clientes, onde as estruturas de banco de dados podem ser extremamente diferentes, tornando a identificação de conceitos semelhantes uma tarefa muito árdua. O mesmo ocorre em empresas de grande porte, onde vários sistemas foram criados no passado para atender a diversas necessidades e, atualmente, identifica-se muita redundância.

Um problema central no contexto de integração de bancos de dados consiste em alinhar os seus esquemas conceituais. *Alinhamento de esquemas* refere-se a problema de encontrar um alinhamento μ entre conceitos em um *esquema de origem* S e conceitos do *esquema de destino* T tal que, se $t = \mu(s)$, então s e t possuem o mesmo significado (CASANOVA, *et al.*, 2007).

Para este problema de alinhamento de esquemas existem muitas soluções, principalmente quando o foco são soluções de integração na *Web*. Neste ínterim grandes empresas como a Google, Yahoo e Microsoft, têm investido milhões de dólares a fim de criar uma ferramenta de busca capaz de "entender" as diversas variações de um determinado conceito, tendo em vista encontrar resultados cada vez melhores quando uma pessoa realiza uma consulta na *Web*.

No entanto, um dos desafios é justamente ter uma base de dados diversificada que possa ser utilizada para testar e aferir a qualidade dos algoritmos de alinhamento de esquemas que são desenvolvidos. De fato, a dificuldade de se encontrar um banco de dados que possibilite testar as diversas técnicas de alinhamento de esquemas é um obstáculo no caminho da evolução e criação de soluções que resolvam de forma ótima e correta os problemas relacionados a

alinhamento de esquemas. É justamente esta lacuna que o presente trabalho visa preencher.

1.2 Objetivo

A motivação primária do presente trabalho surgiu da dificuldade em criar um conjunto de esquemas conceituais, populados com dados, que possibilite testar as diversas técnicas de alinhamento de esquemas. Os esquemas representam modelos conceituais alternativos para a mesma realidade e refletem variações estruturais típicas encontradas em modelagem conceitual.

É muito comum na modelagem conceitual existir representações diversas sobre o mesmo conceito, ou seja, para um caso de modelagem de profissional em uma aplicação empresarial, tal conceito pode ser modelado como uma só entidade PROFISSIONAL ou até mesmo ser modelado como várias entidades especializadas deste profissional, por exemplo, PROFISSIONAL_PROPRIO e PROFISSIONAL_TERCEIRIZADO. Assim, os algoritmos de alinhamento de esquemas conceituais devem ser capazes de identificar esses conceitos como similares.

Baseando-se nesta motivação, o objetivo principal deste trabalho consiste em desenvolver uma ferramenta para criar bases de dados que representem esquemas conceituais com suas respectivas variações. Estas bases formam então *benchmarks* para algoritmos de alinhamento de esquemas. A ferramenta registra ainda as transformações sofridas pelo esquema conceitual de origem para que se identifiquem os alinhamentos que deveriam ser encontradas pelos algoritmos de alinhamento, traçando assim um resultado esperado. Com isso, vislumbra-se a possibilidade de disponibilizar uma ampla e variada gama de benchmarks que possibilite testar algoritmos que implementam técnicas de alinhamento de esquemas conceituais.

O segundo objetivo consiste em criar um método de avaliação para algoritmos de alinhamento de esquemas submetidos a um *benchmark*. De fato, a partir do esquema conceitual de origem, dos esquemas conceituais gerados através das transformações e dos alinhamentos induzidos por estas transformações, é possível estabelecer a precisão de um algoritmo de alinhamento de esquemas.

1.3 Trabalhos relacionados

1.3.1 Alinhamento de esquemas

Pode-se definir um *alinhamento* entre um *esquema de origem* S_1 e um *esquema de destino* S_2 como um conjunto de mapeamentos, também chamados de *alinhamentos*, entre elementos do esquema S_1 e elementos do esquema S_2. Além disso, cada alinhamento pode estar associado a uma *expressão de mapeamento* que especifica como os elementos de S_1 e S_2 estão relacionados (RAHM;BERNSTEIN, 2001).

Em Rahm & Bernstein (2001), as técnicas de alinhamento de esquemas podem ser classificadas em basicamente três grupos: *baseadas em esquema*, *baseadas em instâncias* (ou *extensivas*) e *híbridas*.

Nas técnicas baseadas no esquema, as evidências para gerar os alinhamentos são extraídas das definições do esquema, como nomes, descrições, tipos de dados, relacionamentos e afirmações. Por exemplo, a propriedade *Cliente.nome* do esquema de origem pode ser correspondente à propriedade *Comprador.nomeCompleto* do esquema de destino, já que seus nomes (‘*nome’* e ‘*nomeCompleto*’) são sintaticamente similares.

Nas técnicas baseadas em instâncias, as evidências são extraídas a partir dos dados associados aos elementos dos esquemas. Por exemplo, se as propriedades *Classe1.propriedade1* e *Classe2.propriedade2* do banco de dados de origem e de destino, respectivamente, apresentam os valores representados na Figura 1, então, apesar de os nomes das propriedades não estarem sintaticamente relacionados, as duas propriedades parecem ser correspondentes porque elas possuem o mesmo conjunto de valores. Além disso, se existe um conhecimento prévio de que a propriedade *Livro.autor* pode assumir valores do conjunto apresentado na Figura 1, então pode-se inferir que as duas propriedades anteriores devem ser equivalentes a *Livro.autor* pois seus conjuntos de valores são semelhantes.

Classe1.propriedade, *Classe2.propriedade* ∈ {“Guimarães Rosa”,”Machado de Assis”,”William Shakespeare”}

Livro.autor ∈ {“Edgar Allan Poe”,”Guimarães Rosa”,”Machado de Assis”,”William Shakespeare”}

Figura 1 - Conjunto de propriedades de Classe1.propriedade1, Classe2.propriedade2 e Livro.autor.

Em (GOMES, 2010) propõe-se a criação de um infraestrutura para alinhamento de esquemas, formulada como um framework que auxilia a implementação de técnicas de alinhamento de esquemas OWL baseadas em instâncias, que dependem da definição de funções de similaridade para avaliar a proximidade semântica dos elementos de dois esquemas diferentes. Em (SAYYADIAN, *et al.*, 2005) é apresentada uma abordagem para automaticamente otimizar um sistema de alinhamento de esquemas. Dado um esquema *S*, este esquema é alinhado com esquemas sintéticos, para que o verdadeiro mapeamento seja conhecido. Em (MORGADO, 2010) foi desenvolvido uma forma para criar várias bases de dados fundamentadas em cenários sintéticos, visando possibilitar testes de algoritmos de alinhamento de esquemas.

Em (MELNIK, *et al.*, 2001) foi proposto um algoritmo de alinhamento de esquemas baseado em grafos. O algoritmo em posse de dois grafos (esquema, catálogo, ou outras estruturas de dados) como entrada, produz como saída um mapeamento entre os nós correspondentes dos grafos. Após isso, a precisão do algoritmo é medida pelo número de ajustes necessários.

1.3.2 *Benchmarks*

Alguns trabalhos têm sido elaborados com o objetivo de criar esquemas de dados sintéticos para que se possam ser utilizados para avaliar algoritmos de alinhamento.

eTuner (SAYYADIAN, *et al.*, 2005) é uma abordagem sistemática de configuração de parâmetros de controle nos sistemas de alinhamento de esquemas

suportados por uma coleção de casos de testes gerados automaticamente. Na essência, o eTuner considera este processo de configuração como um problema de busca que localiza configurações eficazes para valores de parâmetros sob certas circunstâncias. O espaço de configuração de um sistema de alinhamento é definido com diferentes combinações do "alinhadores" de componentes que potencialmente reque valores candidatos para vários parâmetros de controle. Em ordem para modificar um sistema de alinhamento, eTuner primeiro gera uma coleção de casos de testes sintéticos onde a verdade absoluta é conhecida. Então aplica-se o sistema nos casos de teste, compara-se os resultados com a verdade absoluta e depois os resultados são reportados. A configuração com a qual os sistemas de alinhamento tem alcançado os melhores valores é identificada como um parâmetro de configuração eficaz. Os esquemas de teste ao qual eTuner avalia os sistemas são gerados pela aplicação de um número de regras para introduzir perturbações nos esquemas existentes.

STBenchcmark (ALEXE, *et al.*, 2008) é um *benchmark* para comparação de sistemas de construção de alinhamento visual interativo que visam apoiar um especialista na geração precisa de especificações de alinhamento entre dois esquemas com o menor esforço. STBenchmark cria um conjunto básico de cenários de alinhamento que são resultado de uma cuidadosa análise de várias aplicações de integração de dados. Ao invés de exaustivamente enumerar todas as possibilidades de cenários de alinhamento, o autor apenas pretende apresentar casos comuns que possuem relevância na indústria.

XBenchMatch (DUCHATEAU, *et al.*, 2007) é um *benchmark* voltado para alinhamento de esquemas XML. Ele provê uma ferramenta para computar propriedades dos cenários dados, aplicando sistemas de alinhamento e resultados de alinhamento. XBenchMarch escolhe quatro conjuntos de esquemas XML do mundo real como tarefas de alinhamento, cada um representando um cenário de alinhamento.

MatchBench (GUO, *et al.*) é um *benchmark* que tem por objetivo inferir correspondências de esquemas entre dois esquemas através de um bom entendimento das propostas de alinhamento de esquemas existentes, e como essas correspondências serão inferidas a partir dos alinhamentos. O MatchBench foi projetado e implementado para obter conhecimento sobre os sistemas de alinhamento existentes. Desta forma, os objetivos do MatchBench são:

- 1- identificar a medida para quais sistemas de alinhamento podem diagnosticar heterogeneidades;
- 2- estabelecer sob que circunstâncias sistemas específicos lutam para diagnosticar os tipos de heterogeneidades;
- 3- suportar a identificação de construções de *matchers* complementares sobre os resultados do 1 e 2.

1.3.3 OAEI

A metodologia utilizada pela OAEI disponibiliza, além de alguns casos de teste para que os participantes possam testar seus sistemas, um ambiente de execução para que testem as ferramentas.

Com essa iniciativa, é possível submeter diversos algoritmos a uma avaliação centralizada de forma a obter a eficácia de cada solução proposta.

Na última edição do OAEI (OAEI 2011), foram consideradas 4 categorias, compreendendo 6 data sets e modalidades de avaliações diferentes. Foram elas:

- *The benchmark track*: Apresenta um conjunto sistemático de *benchmark*. O objetivo deste conjunto de *benchmarks* é identificar as áreas que cada algoritmo de alinhamento é forte ou fraco. O teste é baseado em uma ontologia particular dedicada a um domínio muito específico da bibliografia e um número de ontologias alternativas do mesmo domínio para que as referências de alinhamento sejam providas.
- *The expressive ontologies track*: Oferece ontologias do mundo real utilizando capacidades da modelagem OWL:
    - Anatomia: O caso real de anatomia é sobre alinhar a anatomia de um rato adulto e parte da enciclopédia NCI descrevendo a anatomia humana.
    - Conferência: O objetivo da tarefa da conferência é descobrir todas as correspondências corretas sem uma coleção de ontologias descrevendo o domínio das conferências organizadas.
- *Oriented alignments*: Focaliza a avaliação de alinhamentos que contém outras relações de equivalência. Fornece dois *data sets* de

ontologias reais, gerados a partir de dados acadêmicos e de catálogos de cursos.

- *Model matching*: Compara ferramentas de alinhamento de modelos da comunidade de ontologias *Model-Driven Engineering* (MDE). Os modelos a serem alinhados foram automaticamente gerados de um repositório baseado em modelos.
- *Instance matching*: O objetivo da tarefa de alinhamento de instâncias é avaliar a performance de ferramentas diferentes em tarefas de alinhamento de indivíduos RDF que foram obtidos de diferentes origens mas descrevem a mesma entidade do mundo real. O alinhamento de instâncias é organizado em duas subaéreas:
    - *Data interlinking* (DI): Neste ano, o DI dedicou-se a obter ligações do New York Times (NYT) com DBPedia, Freebase e Geonames. O data set do NYT inclui 4 sub-conjuntos: pessoas, localizações, organizações e descrições que devem ser alinhados entre eles para detectar duplicações.
    - *OWL data track* (IIMB): A categoria de dados sintéticos OWL tem como objetivo prover um conjunto de avaliações para vários tipos de transformações, incluindo transformações de valores, de estrutura e transformações lógicas.

Test	formalism	relations	confidence	modalities	language	SEALS
Benchmarks	OWL	=	[0 1]	open	EN	✓
Anatomy	OWL	=	[0 1]	open	EN	✓
Conference	OWL-DL	=, <=	[0 1]	blind + open	EN	✓
Di	RDF	=	[0 1]	open	EN	
Iimb	RDF	=	[0 1]	Open	EN	

Tabela 1 - Características dos casos de teste

1.3.4 MatchMaking

O *MatchMaking* (GOMES, 2010) é uma ferramenta de software desenvolvida com o intuito de auxiliar a implementação de técnicas de alinhamento de esquemas OWL baseadas em instâncias, que dependem da definição de funções

de similaridade para avaliar a proximidade semântica dos elementos de dois esquemas diferentes. A ferramenta foi projetada para permitir a utilização de diferentes funções de similaridade e a troca dos algoritmos de alinhamento, facilitando assim a experimentação com diferentes configurações de alinhamento.

O *Matchmaking* tem por objetivo prover uma infraestrutura para facilitar a criação dos algoritmos de alinhamento de esquemas, tendo em vista o ponto de vista do criador do algoritmo, e diminuir a complexidade de uma operação de alinhamento levando-se em consideração o ponto de vista do usuário da ferramenta.

A ferramenta deve armazenar a proveniência de cada processo de alinhamento ocorrido, ou seja, deve acomodar um histórico de todos os alinhamentos contendo os parâmetros de execução, etapas do algoritmo que foram processadas, funções de similaridade utilizadas e representações de dados aplicadas para que o processo fosse concluído. Esses dados sobre proveniência precisam estar disponíveis para o usuário poder comparar os diferentes resultados obtidos.

Por fim, com o intuito de fomentar a comparação de resultados, o *MatchMaking* deve ser capaz de realizar operações de alinhamento em série, automaticamente, ou seja, o usuário poderá definir esquemas de origem e de destino (e seus *datasets*) e especificar uma série de operações de alinhamento que devem ser executadas. Após isso, a infraestrutura executará cada operação de alinhamento especificada a série de forma automática, sem necessidade de interação adicional com o usuário.

Com esta proposta, o *MatchMaking* é capaz de fornecer facilidades para o desenvolvimento de algoritmos de alinhamento de esquemas, com uma estrutura reutilizável e adaptável que pode ser usada por um desenvolvedor para testar seus próprios algoritmos, funções de similaridade e representações de dados. Outro detalhe importante é que apesar de a infraestrutura utilizar esquemas descritos em ontologias OWL, são necessárias pequenas modificações em seu código para que o *Matchmaking* consiga processar outras tecnologias.

Diferente do *MatchMaking*, o *Bench* irá gerar *benchmarks* para que possam ser utilizados inclusive pelo próprio *MatchMaking* para realização de testes de algoritmos de alinhamento de esquemas.

2. FUNDAMENTOS

Este capítulo aborda em detalhe alinhamento de esquemas, introduz as principais técnicas de alinhamento de esquemas conhecidas e apresenta um resumo de RDF e OWL. Este resumo preliminar permite discutir como construir uma base de dados que possa ser utilizada em testes de algoritmos de alinhamento de esquemas.

2.1 Alinhamento de esquemas conceituais

Um *esquema conceitual de banco de dados* ou, simplesmente, um *esquema* é uma descrição em alto nível de como os conceitos de um banco de dados estão organizados. O alinhamento de um esquema de origem S com um esquema de destino T define conceitos em T em termos dos conceitos de S.

O problema de encontrar o alinhamento entre esquemas se torna um desafio quando diferentes vocabulários são usados para se referir ao mesmo conceito do mundo real (CASANOVA, *et al.*, 2007). Neste caso, uma abordagem conveniente, às vezes chamada de *extensiva*, *baseada em instâncias* ou *semântica*, é detectar como um mesmo objeto do mundo real é representado em diferentes bancos de dados e usar a informação obtida desta maneira para alinhar os esquemas. Essa abordagem é fundamentada na interpretação tradicionalmente aceita de que "termos têm a mesma extensão quando denotam a mesma coisa" (tradução livre) (QUINE, 1968).

Alinhamento de esquemas é um requisito fundamental em muitas aplicações de banco de dados, tais como mediação de consultas, integração de banco de dados, alinhamento de catálogos e construção de *data warehouses* (CASANOVA, *et al.*, 2007).

Como exemplo de integração de banco de dados, podemos citar o caso de uma empresa de grande porte que possua várias filiais, cada uma com seus próprios sistemas, criados independentemente. Com este cenário típico, podemos

ter uma infinidade de conceitos e dados semelhantes, mas implementados de várias formas e em vários locais diferentes. Para isso, é necessária uma visão mais ampla que permita identificar os pontos de convergência para se promover a integração/unificação destes dados redundantes.

Um caso semelhante ocorre quando duas empresas se fundem e suas carteiras de clientes precisam ser unificadas. Neste caso, os dados de clientes, estruturados de diferentes formas por cada empresa, precisam ser alinhados, a fim de que a empresa fruto da fusão possa ter uma base de clientes integrada, consistente e sem redundâncias.

Estes diversos casos expostos são muito comuns nas empresas, por essa razão é fundamental buscarmos alinhar e unificar as bases de dados legadas da empresa. Assim sendo, poderemos garantir consistência, integridade, confiabilidade e disponibilidade da informação para apoiar os diversos processos de negócio da empresa.

2.2 Técnicas e Métodos de Alinhamento de Esquemas

O trabalho de (RAHM;BERNSTEIN, 2001) explana as diversas abordagens de alinhamento de esquemas, inclusive citando a importância de se combinar essas abordagens de forma a obter um alinhamento com maior precisão.

Primeiramente, é sugerido realizar a execução de uma única abordagem utilizando somente um critério. Posteriormente deve ser realizada a combinação das abordagens individuais utilizando múltiplos critérios, juntamente com um *matcher* híbrido ou pela combinação de múltiplos resultados de alinhamento produzidos por diferentes algoritmos de alinhamento com um *matcher* composto.

Dentre as abordagens citadas por (RAHM;BERNSTEIN, 2001) (Figura 2), são consideradas os seguintes critérios de classificação:

- Instância x Esquema: abordagens de alinhamento podem considerar dados de instância (ou seja, conteúdo dos dados) ou somente informação em nível de esquema.
- Elemento x Estrutura de alinhamento: alinhamento pode ser obtido através de elementos individuais do esquema, tais como atributos ou pela combinação de elementos, tais como estruturas complexas de esquemas.
- Linguagem x Restrição: um alinhamento pode utilizar uma abordagem baseada em linguística (baseados nos nomes ou descrições textuais de

elementos do esquema) ou uma abordagem baseada nas restrições (baseada nas chaves e relacionamentos).

- Cardinalidade do alinhamento: no geral, um resultado de alinhamento pode relacionar um ou mais elementos de um esquema à um ou mais elementos de um outro esquema.
- Informações auxiliares: a maioria dos *matchers* não trabalham somente com os esquemas S1 e S2, mas também com informações auxiliares, tais como, dicionários, esquemas globais, etc.

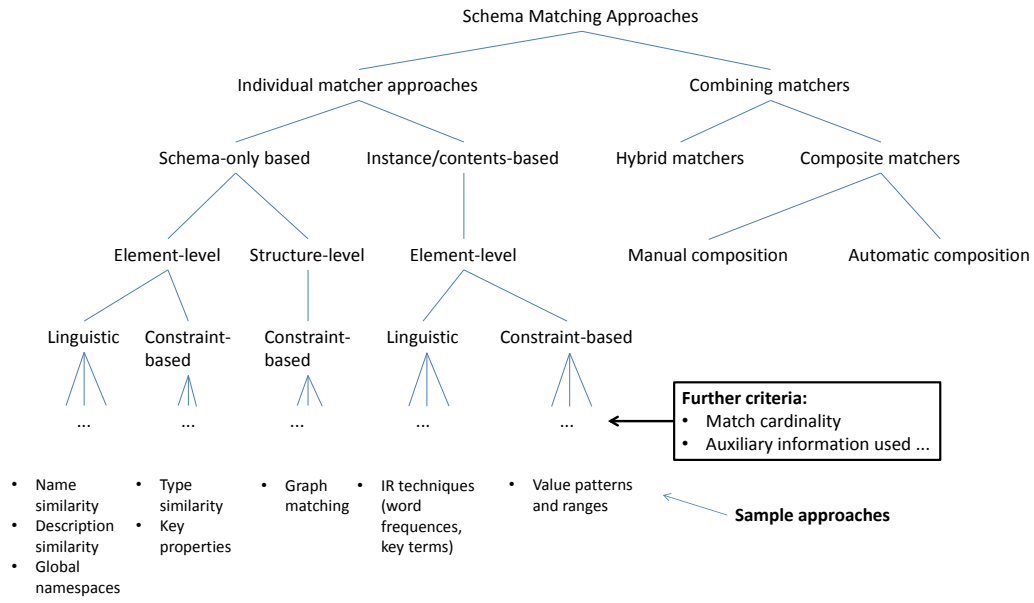

Figura 2 - Classificação das abordagens de alinhamento de esquemas

2.3 RDF / OWL

(KLYNE, *et al.*, 2004) definem como recurso tudo aquilo que possui alguma identidade, podendo esta ser digital (como um documento eletrônico, uma imagem ou um serviço), física (como um livro) ou uma coleção de outros recursos. Um *Uniform Resource Identifier* (URI) é uma cadeia de caracteres utilizada para identificar um recurso na *Web*. Uma *URIref* apresenta um dos usos mais comuns para URI, acrescentando um fragmento de identificação opcional precedido pelo caractere "#".

RDF apresenta uma linguagem de descrição que pode ser utilizada de forma padronizada e assim aplicações compartilham e trocam de informações, sem que haja perda de significado. No caso deste trabalho, este recurso será utilizado

apenas com o intuito de transportar os dados. Essas informações podem ser sobre recursos da *Web* ou objetos que são identificados na *Web* mas não podem ser recuperados por ela (como pessoas, veículos, etc.). Uma afirmação em RDF (ou simplesmente afirmação, ou statement) é uma tripla (S,P,O), onde

- *S* é um recurso chamado de *sujeito* da afirmação
- *P* é um recurso chamado de *propriedade* (as vezes chamado de *predicado*) da afirmação, ele denota uma relação binária
- *O* pode ser tanto um recurso ou um literal, chamado de objeto de uma assertiva; se *O* é um literal então O também é chamado de *valor* da propriedade *P*

Apesar de RDF possuir uma boa flexibilidade e permitir que consigamos descrever recursos com relativa precisão, apenas a propriedade *rdf:type* possui uma semântica pré-estabelecida. Essa restrição impede que estruturas complexas como classes, hierarquia de classes e propriedades possam ser descritas. Para isso existem as extensões de RDF, como RDF *Schema* [8], que formam um vocabulário reservado utilizado para descrever esses itens de uma aplicação.

Uma classe, no RDF *Schema*, é um recurso que possui a propriedade *rdf:type* com o valor *rdfs:Class*. Por exemplo:

```
Escola rdf:type rdfs:Class
```

Para definir uma classe como subclasse de outra, deve-se utilizar a propriedade *rdfs:subClassOf* para relacionar as duas classes. Subclasses são transitivas. Por exemplo:

```
Carro rdfs:subClassOf Veículo

CarroTransporte rdfs:subClassOf Carro
```

e, por transitividade:

```
CarroTransporte rdfs:subClassOf Veículo
```

Uma propriedade é qualquer instância da classe *rdfs:Property*. A propriedade *rdfs:domain* é usada para indicar que uma propriedade em particular se aplica a uma determinada classe, e a propriedade *rdfs:range* é usada para indicar que os valores de uma propriedade em particular são instâncias de uma

determinada classe ou, alternativamente, são instâncias (literais) de um tipo de dados do XML *Schema* (XML *Schema datatype*). Por exemplo:

```
peso rdf:type rdf:Property

peso rdfs:range xsd:double

peso rdfs:domain Veículo
```

A especialização de uma propriedade é descrita através de *rdfs:subPropertyOf*. Uma propriedade RDF pode ter zero ou mais sub propriedades; todas as propriedades *rdfs:range* e *rdfs:domain* do esquema RDF que se aplicam a uma propriedade RDF também são aplicadas em cada uma das suas subpropriedades (subpropriedades tem o mesmo conceito de subclasses. Por exemplo, se a propriedade `motorFlex` for subpropriedade de `motorCombustão`, isso significa que a instância que possui `motorFlex` como propriedade também possui `motorCombustão`).

Uma instância de uma classe é um recurso que possui o valor da propriedade *rdf:type* com o valor do *URIref* da classe correspondente. Um recurso pode ser uma instância de mais de uma classe. Para definir o valor de uma propriedade de uma classe na instância, basta criar uma afirmação como no exemplo:

```
I.    KombiXSR0001 rdf:type CarroTransporte
II.   KombiXSR0001 peso 1205,00
```

No caso, em (I) a instância `KombiXSR0001` é criada como sendo da classe CarroTransporte. Já em (II) o peso `1205,00` é atribuído.

OWL (BECHHOFER, *et al.*, 2004) é uma linguagem de ontologia *Web* criada pelo W3C *Web Ontology Working Group* que estende o RDF *Schema* definindo um vocabulário mais abrangente, fornecendo meios de se tratar relacionamentos, cardinalidades e outras definições complexas. A OWL tem três dialetos: OWL Lite, OWL DL e OWL Full, cada uma com mais expressividade que a anterior. A OWL Lite, por exemplo, permite somente cardinalidades de valor 0 ou 1, enquanto a OWL Full não faz esse tipo de restrição.

Na OWL, uma classe é qualquer recurso que possua o valor *owl:Class* na propriedade *rdf:type*, sendo que *owl:Class* é uma subclasse de *rdf:Class*.

A OWL separa as propriedades em duas categorias diferentes: propriedades de objeto (*object properties*), que relacionam indivíduos a indivíduos, e propriedades de tipos de dados (*datatype properties*), que relacionam indivíduos a dados literais. A primeira categoria define os relacionamentos entre as classes. Ambas são subpropriedades de *rdfs:Property*.

Uma propriedade especial, a *owl:sameAs*, define que dois recursos representam o mesmo individuo. Por exemplo, a tripla RDF *(uri1,owl:sameAs,uri2)* define que *uri1* e *uri2* representam o mesmo individuo (ou instância) no banco de dados. A propriedade *owl:equivalentClass* e *owl:equivalentProperty* são análogos à propriedade *owl:sameAs*, mas esses se referem a duas classes e a duas propriedades, respectivamente.

Um banco de dados OWL é um conjunto de triplas no vocabulário da OWL. Note que o conjunto de triplas pode incluir indistintamente definições relativas ao esquema conceitual do banco de dados e instâncias do banco.

2.4 *Linked Data*

De acordo com (BIZER, *et al.*, 2008), *Linked Data* refere-se ao uso de RDF e HTTP (*Hypertext Transfer Protocol*) para publicar dados estruturados na *Web* e para conectar dados de diferentes fontes de dados, permitindo de forma eficiente dados de uma fonte serem conectado com dados de outra fonte.

Os princípios do *Linked Data* foram primeiramente descrito por (BERNERS-LEE, 2006), e fornece amplas orientações sobre quais editores de dados deveriam começar a realizar a *Web of Data* (Rede de dados). O guia foi ampliado por documentos técnicos tais como (BIZER, *et al.*, 2007) (SAUERMANN, *et al.*, 2007) que capturam melhores práticas que estão surgindo da comunidade *Linked Data* e fornece receitas sobre quais sistemas de publicações podem ser baseados.

A *Web of Data* pode ser acessada utilizando navegadores do *Linked Data*, assim como os tradicionais documentos *Web* são acessados utilizando navegadores HTML. No entanto, ao invés de *links* entre páginas HTML, navegadores do *Linked Data* permitem usuários a navegar entre diferentes fontes

de dados através de *links* RDF. Isso permite ao usuário começar com uma fonte de dados e então navegar através de uma potencial rede interminável de fontes de dados conectados por *links* RDF, assim como um documento tradicional da *Web* pode ser rastreado por *links* RDF. Trabalhando com dados rastreados, sistemas de busca podem prover habilidades para consultas sofisticadas, parecidas com as disponíveis por banco de dados relacionais convencionais. Devido ao fato dos resultados das consultas serem dados estruturados e não somente *links* para páginas HTML, eles podem ser processados imediatamente, possibilitando assim uma nova classe de aplicações baseadas na *Web of Data*.

O que une documentos tradicionais na *Web* são os *links* hipertextos entre páginas HTML. Por sua vez o que une os dados na *web* são os *links* RDF. Um *link* RDF simplesmente indica que uma parte de dados tem algum tipo de relacionamento com outra parte de dados Estes relacionamentos podem ser de diferentes tipos. Por exemplo, um *link* RDF que conecta dados sobre pessoas pode afirmar que duas pessoas se conhecem; um *link* RDF que conecta informações sobre uma pessoa com informações sobre publicações em uma base de dados bibliográficos poderia afirmar que a pessoa é o autor de um artigo específico.

Uma evidência da ocorrência da *Web of Data* é o projeto *Linking Open Data*, uma base comunitária fundada em Fevereiro de 2007 e apoiada pelo W3C *Semantic Web Education and Outreach Working Group*. O objetivo do projeto é identificar conjunto de dados que estão disponíveis com licenças abertas, republicar este RDF na *Web* e interligá-lo com outros.

Em 2007, o tamanho da *Web of Data* estava em torno de dois bilhões de triplas RDF, originárias de *data sets* em diversos domínios tais como informações geográficas, informações de senso, pessoas, companhias, comunidades online, linguagens humanas, publicações científicas, filmes, musicas, livros e revisões. Estes *data sets* são interligados por em torno de três milhões de *links* RDF.

Uma noção da abrangência e do tamanho da "nuvem" *Linking Open Data* é vista na Figura 3. Como este diagrama mostra, os principais eixos de ligação são sites como DBpedia e Geonames.

Além da publicação e interligação dos *data sets*, existe também o trabalho contínuo com a comunidade do projeto em navegadores *Linked Data*, rastreadores *Linked Data*, sistema de busca *Web of Data* e outras aplicações que consomem *Linked Data* da *web*.

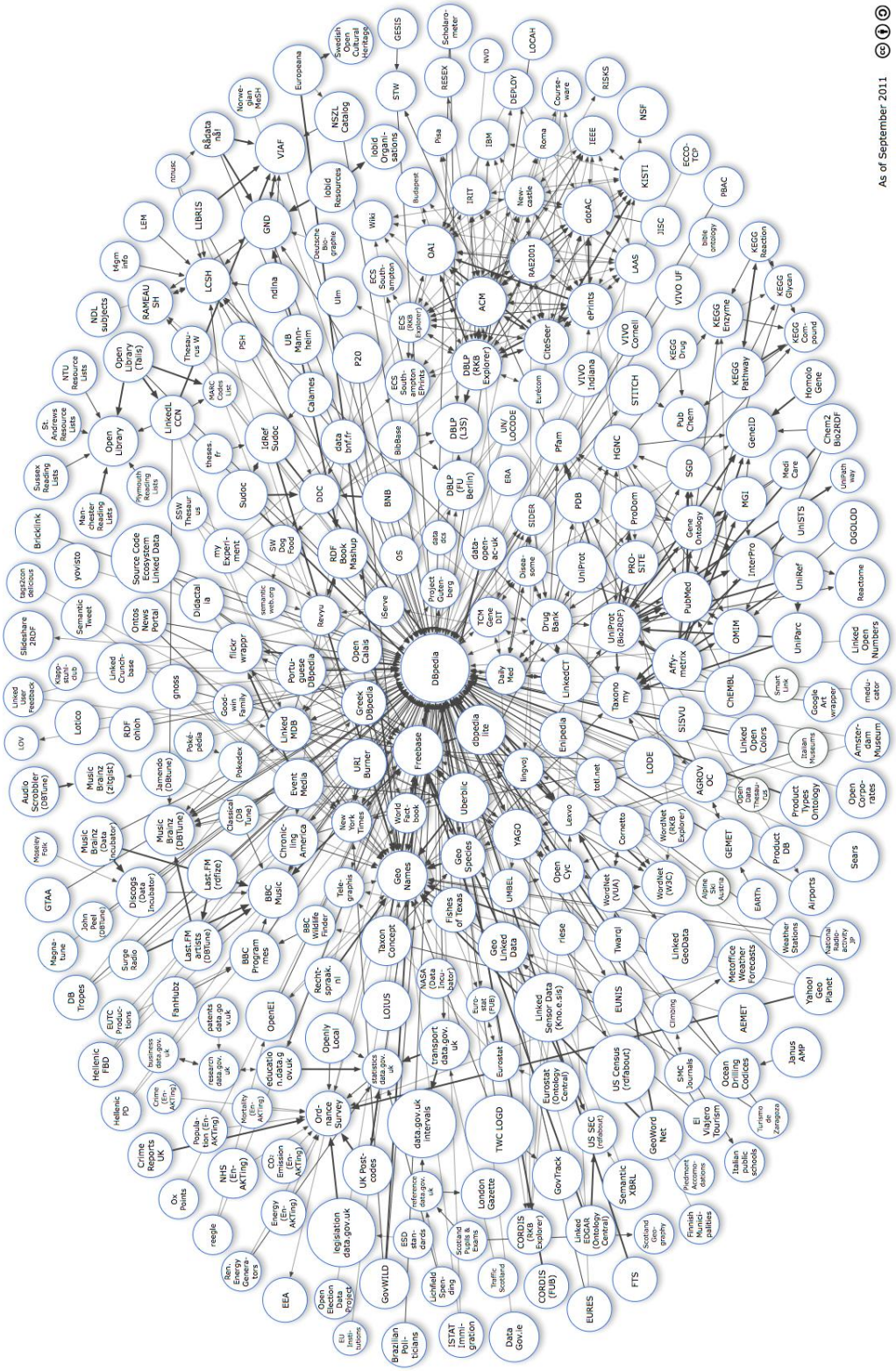

Figura 1 - A "nuvem" *Linking Open Data*

3. GERAÇÃO AUTOMÁTICA DE ESQUEMAS SINTÉTICOS

Este capítulo aborda a geração automática de esquemas sintéticos. A seção 3.1 descreve os conceitos principais sobre um *benchmark* e seu papel na área de pesquisa de algoritmos de alinhamento. Por sua vez, a seção 3.2 trata das possíveis transformações ou variações que um esquema pode sofrer. A seção 3.3 aborda a geração de esquemas sintéticos a partir de *datasets* existentes; estes por sua vez sofrendo transformações que criarão novos esquemas sintéticos. Por fim, a seção 3.5 comenta como esta proposta foi implementada no benchmark *Bench*.

3.1 *Benchmarks*

"Em computação, *benchmark* é o ato de executar um programa de computador, um conjunto de programas ou outras operações, a fim de avaliar a performance relativa de um objeto, normalmente executando uma série de testes padrões e ensaios nele." (WIKIPEDIA, 2012).

A partir desta definição, a ferramenta proposta neste trabalho, chamada de *Bench*, visa facilitar a criação de benchmarks e avaliar algoritmos de alinhamento de esquemas. Para demonstrar como tal proposta pode ser implementada, discorreremos a seguir quais são as características necessárias à geração de *benchmarks*.

3.2 Transformações

Chamamos de *transformações* as variações que um esquema pode sofrer, gerando assim novos esquemas. A ferramenta *Bench* utilizará estas transformações para avaliar o desempenho dos algoritmos de alinhamento de esquemas.

A seguir serão descritos as transformações referenciadas por (ALEXE, *et al.*, 2008) e adotadas na construção do *Bench*.

3.2.1 Cópia

Em muitas aplicações de transformações de dados é frequente o caso em que a instância ou a sub-instância do esquema de origem simplesmente é copiada para o esquema de destino.

Por exemplo, considere os esquemas de origem e de destino mostrados na Figura 4. O esquema de origem descreve um conjunto de registros *Source/Protein*, com três sub-elementos. Este conjunto de registros é então copiado para o conjunto de registros *Souce/Protein*.

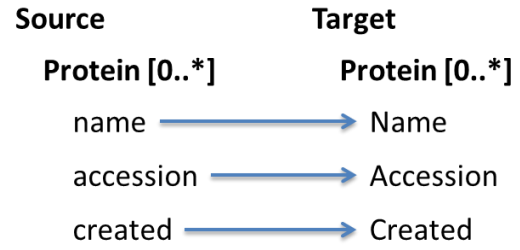

Figura 4 - Transformação Cópia

3.2.2 Geração de valor constante

Esta transformação representa a situação em que algumas constantes precisam ser criadas no esquema de destino.

Por exemplo, considere o esquema de destino mostrado na Figura 5. O esquema de origem pode ser qualquer esquema e ser até mesmo omitido. Aqui, o elemento *Target/DataSet* é criado para gravar o nome do banco de dados e a data que este foi criado.

Neste caso, a constante "*SwissProt*" e "*July 4th*" são os *Name* e *LoadingDate*, respectivamente.

Figura 5 - Transformação Geração de Valor Constante

3.2.3 Particionamento horizontal

Esta transformação representa a situação onde os dados da origem são particionados em dois ou mais fragmentos no destino. Isso pode ser necessário, por exemplo, quando um conjunto de elementos cresce através do tempo a ponto de afetar o desempenho do sistema de gerenciamento de banco de dados. Como descrito, particionamento horizontal tipicamente requer suporte a filtros de dados (condições de seleção).

Por exemplo, considere os esquemas de origem e de destino mostrados na Figura 6. O esquema de origem descreve um conjunto de registros *Source/Gene*, com três sub-elementos. O esquema de destino descreve dois conjuntos de registros, *Target/Gene* e *Target/Synonym*, respectivamente. Os registros da origem são particionados horizontalmente em dois conjuntos no destino, baseados no valor do tipo do sub-elemento: *Target/Gene* contém os registros do *Source/Gene* cujo tipo é "*primary*" e *Target/Synonym* contém os registros *Source/Gene* cujo tipo não é "*primary*". Isso é o típico caso de particionamento horizontal, onde o esquema de cada partição de destino é idêntico ao esquema de origem. No entanto, foi omitido o sub-elemento tipo nas partições de destino.

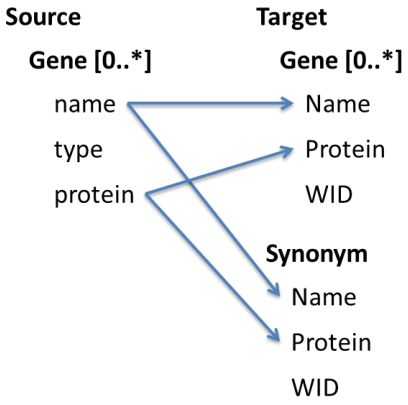

Figura 6 - Transformação Particionamento Horizontal e Associação de Chave Fictícia

3.2.4 Associação de Chave Fictícia

Em *data warehouses*, objetos frequentemente recebem novos identificadores únicos (ou chaves), que pode ser diferente dos identificadores originais.

Considere novamente o exemplo da Figura 6, exceto que agora o esquema de destino considera os sub-elementos WID. Como no cenário de particionamento horizontal, *Target/Gene* consiste dos registros *Source/Gene* em que o tipo é

"*primary*" e *Target/Synonym* consiste dos registros *Source/Gene* em que o tipo não é "*primary*". Além disso, um novo valor é gerado para cada registro no destino. *Target/Gene*/WID é a chave para os registros *Target/Gene* e *Target/Gene*/WID é a chave para os registros *Target/Synonym*.

3.2.5 Particionamento Vertical

Diferentemente do particionamento horizontal, o particionamento vertical divide o conjunto de registros verticalmente em dois ou mais conjunto de registros no destino. Em outras palavras, o destino é tipicamente uma projeção de conjuntos de registros da origem.

Como exemplo, considere os esquemas de origem e de destino mostrados na Figura 7. Os quatro primeiro sub-elementos e os dois últimos de *Source/Reaction* são sub-elementos do *Target/Reaction* e *Target/ChemicalInfo*, respectivamente. Além disso, *Target/Reaction* e *Target/ChemicalInfo* são relacionados através da restrição envolvendo *CoFactor*. A transformação divide cada registro em *Source/Reaction* em dois registros menores no destino. Ao mesmo tempo, um valor único de *CoFactor* é utilizado para relacionar estes dois registros menores no destino.

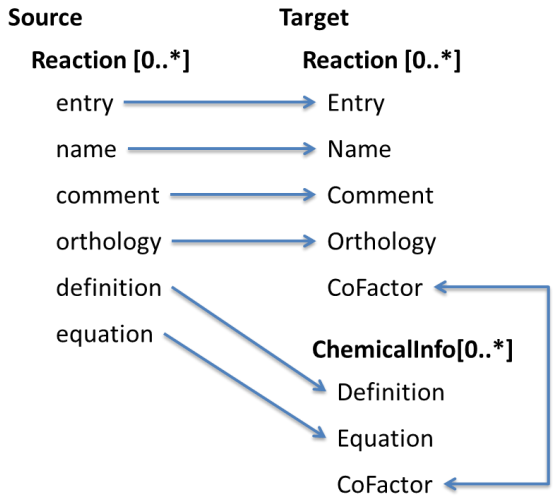

Figura 7 - Transformação Particionamento Vertical

3.2.6 Desaninhamento (ou Achatamento)

Outra transformação comum em integração de dados e, particularmente, em armazenamento *XML-to-relational* é o desaninhamento de estruturas.

Por exemplo, considere os esquemas de origem e de destino mostrados na Figura 8. O esquema de origem descreve um conjunto de registros *Source/Reference*, que possui um conjunto aninhado de registros de autores. O esquema de destino descreve num conjunto de registros *Target/Publication* sem conjuntos aninhados. O conjunto de registros *Reference* na origem é "achatado" no conjunto de registros *Publication* no destino. Em outras palavras, *Target/Publication* consiste em um conjunto de registros que é o resultado do produto cartesiano de cada registro *Source/Reference* com seus respectivos conjuntos de tuplas *Author* aninhados.

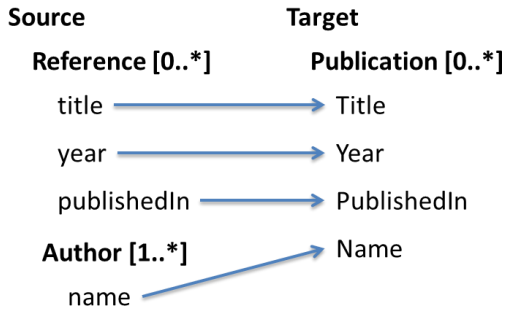

Figura 8 - Transformação Desaninhamento

3.2.7 Aninhamento

A transformação aninhamento, que é o oposto de desaninhamento, ocorre com frequência em integração de dados, publicação *relational-to-XML* e evolução de esquema.

Por exemplo, considere os esquemas de origem e de destino mostrados na Figura 9. *Source/Reference* é um conjunto de dados "achatados". No destino, dados sobre referências são organizado de tal forma que, para cada ano, existe associado um conjunto de autores que publicaram naquele ano e, para cada autor, existe um conjunto associado de publicações daquele autor.

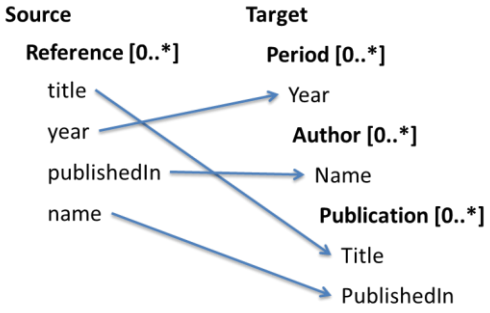

Figura 9 - Transformação Aninhamento

3.2.8 Auto-relacionamento

Relacionamento é a principal operação utilizada para associar dados localizados em conjuntos diferentes ou mesmo entre dados pertencentes ao mesmo conjunto.

O exemplo a seguir é derivado de uma aplicação real de integração de dados *Biowarehouse*. Considere os esquemas de origem e de destino mostrados na Figura 10. O esquema de origem descreve um único conjunto de registros *Source/Gene*, com três subelementos descrevendo o nome do gene, o tipo (se é "*primary*" ou não) e a proteína que gene pertence. O esquema de destino descreve dois conjuntos de registros, *Target/Gene* e *Target/Synonym* respectivamente. Como uma proteína pode ter múltiplos genes, mas somente um gene é primário, o gene primário é armazenado no *Target/Gene*, enquanto todos os outros genes da mesma proteína são armazenados no *Target/Synonym*. Além disso, uma chave estrangeira *Target/Synonym*/WID referenciando *Target/Gene/Name* indica o gene primário de cada sinônimo e a proteína à qual eles pertencem.

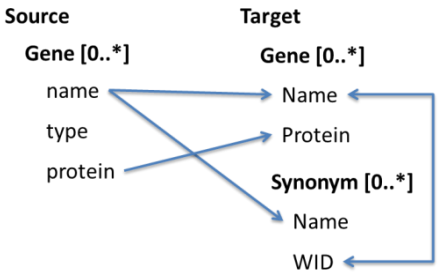

Figura 10 - Transformação Auto-relacionamento

3.2.9 Desnormalização

Dados relevantes ou sobrepostos são geralmente encontrados em diferentes bases de dados e precisam ser combinados. A desnormalização retrata essa situação e pode ser vista como o cenário inverso do particionamento vertical.

Por exempo, considere os esquemas de origem e de destino mostrados na Figura 11. Se um registro *Source/Name* tem um valor *id* igual ao valor *taxID* de um registro no *Source/Node*, então os dois registros são combinados para formar o registro *Target/Taxon*. Note que registros *Source/Name* e *Source/Note* podem ser combinados através dos sub-elementos *id-parentID*. No entanto, neste cenário, é necessário que os registros *Source/Name* e *Source/Node* sejam unidos através dos sub-elementos *id-taxID*.

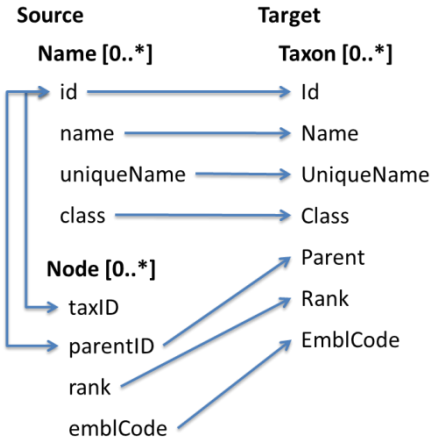

Figura 11 - Transformação Desnormalização

3.2.10 Fusão de Chaves e Objetos

As transformações citadas até então não levaram em consideração que restrições de chave podem existir na origem ou no destino. Chaves são frequentemente utilizadas para unificar dados de diferentes origens que se referem ao mesmo objeto no mundo real e eliminar duplicação. Tais cenários ocorrem frequentemente na integração de dados.

Por exemplo, considere os esquemas origem e destino mostrados na Figura 12. Toda vez que um experimento é executado, detalhes sobre o experimento são registrados no *Source/Experiment*. Cada experimento é unicamente identificado pelos sub-elementos *contact* e *date* e pode ter zero ou mais datas. Um experimento pode também ser associado a zero ou mais

FlowCytometrySample/Probes. Cada *FlowCytometrySample* é também identificado unicamente pelo *contact* e *date*. *Target/Experiment* consolida dados de experimentos e sondas (*probes*), baseados no *contact* e *date*, sem fazer distinção entre dados de experimentos e sondas. Nesta transformação, são migrados todos os *ExperimentalData* do *Source/Experiments* e todos os *Probes* do *Source/FlowCytometrySample* para *Target/Experiment/ExperimentalData*. No destino, todos os *ExperimentalData* do *Source/Experiment* e todos os *Probes* do *Source/FlowCytometrySample* são agrupados pelo *contact* e *date*. Por isso, *Target/Experiment* é essencialmente um *full outer join* de *Source/Experimet* e *Source/FlowCytometrySample* no *contact* e *date*.

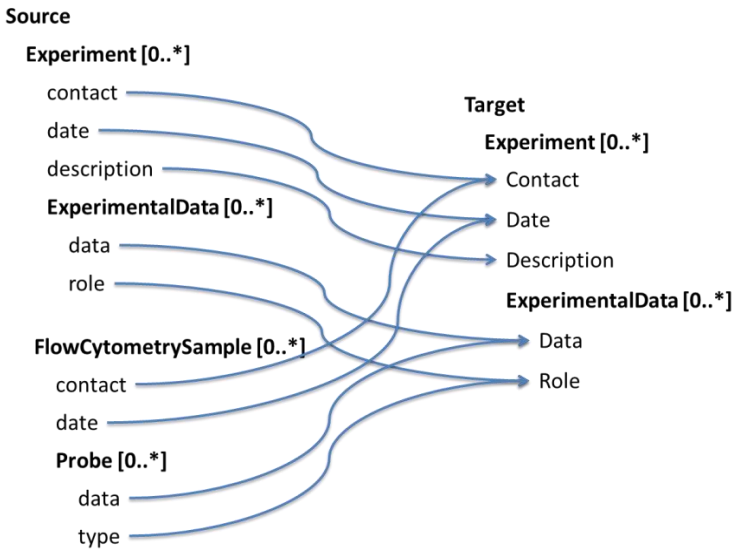

Figura 12 - Transformação Fusão de Chaves e Objetos

3.2.11 Manipulando Valores Atômicos

Na integração de dados e evolução de sistemas, dados armazenados como um elemento atômico em um banco de dados podem ser modelados como mais de um elemento atômico em outro banco de dados. A situação inversa também ocorre com frequência na prática. Devido à heterogeneidade semântica, algumas vezes, existe também a necessidade de aplicar uma função a um valor atômico para mapeá-lo corretamente para um outro banco de dados. Um exemplo simples seria a tradução de dólares em euros. Em todos estes casos, o alinhamento de um elemento para um ou mais elementos e vice versa tipicamente exigirá à aplicação uma ou mais funções.

Por exemplo, considere os esquemas origem e destino mostrados na Figura 13. A transformação assume a existência de três funções *getFirstName*(), *getLastName*() e *concat*(). Para cada registro *Source/Contact*, um registro *Target/Contact* é criado, onde seu valores de *FirstName* e *LastName* são obtidos aplicando-se *getFirstName*() e *getLastName*(), respectivamente, ao valor de *name* do registro de *Source/Contact*. A situação inversa ocorre com *Target/Contact/Address*, onde o valor de *Address* é o resultado da aplicação da função *concat*() sobre *street*, *city* e *zip* de *Address* no registro *Source/Contact*.

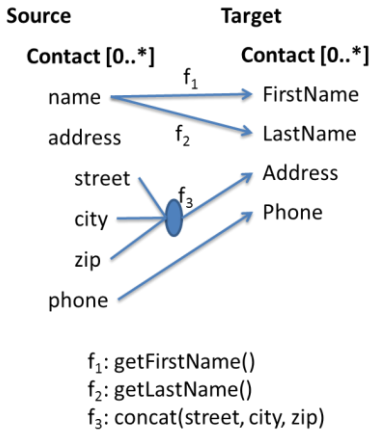

Figura 13 - Transformação Manipulando Valores Atômicos

3.3 Geração de esquemas sintéticos

Para se obter uma variedade de esquemas que possibilitem os testes de algoritmos de alinhamento de esquemas, pode-se fazer uso de basicamente duas estratégias.

A primeira estratégia consiste em selecionar esquemas do mundo real que tratem de conceitos próximos. É essencial que estes esquemas tratem dos mesmos conceitos, pois, caso contrário, não seria viável identificar similaridades entre os esquemas. Desta forma, mesmo os esquemas tendo sido criados sob circunstâncias e formas de pensar diferentes, como tratam de conceitos similares, é possível utilizá-los para testar algoritmos de alinhamento. A *Linked Data Cloud* apresenta vários esquemas do mundo real e como eles se relacionam. Portanto, presta-se bem a esta estratégia.

A segunda estratégia, e mais relevante para este trabalho, consiste em utilizar esquemas reais para, a partir deles, criar outros esquemas sintéticos utilizando as transformações citadas na Seção 3.2. Esta segunda estratégia permite

assim registrar os alinhamentos entre os esquemas do benchmark e utilizá-los para avaliar algoritmos de alinhamento. Outro benefício seria o fato de que esta tarefa de gerar esquemas sintéticos pode ser automatizada, possibilitando criar uma enorme variedade de esquemas com um esforço relativamente baixo, diferentemente da primeira estratégia, onde a identificação de esquemas afins necessita intervenção manual dispendiosa.

Desta forma, optou-se pela segunda estratégia na construção da ferramenta *Bench*.

3.4 Avaliação de Algoritmos de Alinhamento

Esta seção resume o método utilizado para avaliar os algoritmos de alinhamento proposto por (Do, Melnik, & Rahm, 2003).

3.4.1 Comparação de alinhamentos

De acordo com (DO, *et al.*, 2003), apesar de alinhamento de esquemas ser considerado um problema importante e o desenvolvimento de sistemas que permitam a automação deste processo seja foco de intensa pesquisa, a forma de avaliar a eficácia dos sistemas para alinhamento automático de esquemas ainda não é clara.

Para mostrar a eficácia de seus sistemas, os autores normalmente demonstram sua aplicação em um cenário do mundo real ou conduzem um estudo utilizando uma gama de tarefas de alinhamento de esquemas. Infelizmente, a avaliação de sistemas é feita utilizando-se diversas metodologias, métricas e conjuntos de dados, tornando assim difícil a comparação entre dois sistemas.

O *Ontology Alignment Evaluation Initiative* (OAEI - Iniciativa de Avaliação de Alinhamento de Ontologias), detalhado no item 1.3.3, é uma iniciativa coordenada que organiza a avaliação de um crescente número de sistemas de alinhamento de ontologias. O principal objetivo do OAEI é comparar sistemas e algoritmos na mesma base e permitir a qualquer um tirar conclusões sobre a melhor estratégia de alinhamento (EUZENAT, *et al.*, 2011).

Conforme observado em (MELNIK, *et al.*, 2001), uma questão crucial na avaliação de algoritmos de *matching* é que a definição precisa do conjunto de alinhamentos desejado é muitas vezes impossível.

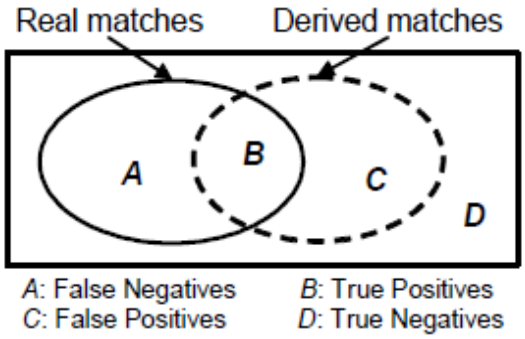

Figura 14 - Comparando alinhamento de esquemas automático

Por exemplo, considere os conjuntos mostrados na Figura 14. Em particular, o conjunto de alinhamentos derivados é composto de:

- Verdadeiros positivos (B), ou seja, alinhamentos corretos
- Falsos positivos (C), ou seja, alinhamentos falsamente propostos pela operação de alinhamento automático
- Falsos negativos (A), ou seja, alinhamentos corretos que não são identificados automaticamente
- Verdadeiros negativos (D), ou seja, alinhamentos falsos, que foram corretamente desconsiderados pela operação de alinhamento automático.

Intuitivamente, falsos negativos e falsos positivos reduzem a qualidade do alinhamento.

3.4.2 Cálculo de Exatidão

Baseado em (MELNIK, *et al.*, 2001), o objetivo é estimar o custo do esforço para o usuário modificar o resultado dos alinhamentos propostos $P=\{(x_1, y_1), ..., (x_n, y_n)\}$ para o resultado esperado $I=\{(a_1, b_1) ,...,(a_m, b_m)\}$.

Em (MELNIK, *et al.*, 2001), afirma-se que o esforço pode ser medido em termos de adições e deleções realizadas em P. Seja $c=||P \cap I||$ o número de sugestões corretas, n o número de sugestões encontradas e m o número de sugestões esperadas. A diferença $(n - c)$ denota o número de falsos positivos que devem ser removidos de P, e $(m – c)$ o número de falsos negativos, ou seja, alinhamentos que precisam ser adicionados. Para simplificar, vamos assumir que as deleções e adições possuem o mesmo esforço, e que a verificação de um alinhamento correto não tenha custo. Se o usuário realizar todo alinhamento manualmente (sem cometer erros), seriam necessárias m operações. Então, o custo

para realizar o ajuste manual após aplicar o algoritmo de alinhamento é $\frac{n-c+(m-c)}{m}$. Desta forma, a *exatidão* é medida pela fórmula:

$$Exatidão = 1 - \frac{n-c+(m-c)}{m}$$

Assim, se $n = m = c$, o resultado da exatidão será 1. Na definição acima, a noção de exatidão somente faz sentido se a precisão não for menor do que 0.5, ou seja, se ao menos a metade dos alinhamentos retornados são corretos. Se esta condição for falsa, a exatidão é negativa. De fato, se mais da metade dos alinhamentos retornados estiverem errados, o usuário teria um esforço maior para remover os falsos positivos e adicionar os alinhamentos que faltaram do que para realizar o alinhamento manualmente. Como esperado, a melhor exatidão (igual a 1.0) é atingida quando ambos, precisão e *recall* são iguais a 1.0. Note que a exatidão é inclinada à precisão. Por exemplo, se o *recall* e precisão medirem respectivamente 0.7 e 0.9, a exatidão será 0.62. Este valor de exatidão é maior do que se precisão e *recall* forem 0.9 e 0.7, que resultaria em uma exatidão de 0.51.

3.4.3 F1-Mesure

Outra métrica implementada no *Bench* foi a *F1-measure,* definida como a média harmônica da *precision* e *recall* (FORMAN, 2002).

A média harmônica (*harmonic mean*) é definida pela relação entre a média geométrica definida por $G = \sqrt{x_1 * x_2}$, e a média aritmética definida por $A = \frac{x_1 + x_2}{2}$. Logo, a média harmônica é definida por $H = \frac{G^2}{A}$.

A *precision P* é definida como $P = \frac{TP}{TP + FP}$, onde *TP* são os *true positives* (verdadeiros positivos – resultados corretos) e *FP* os *false positives* (falsos positivos – resultados inesperados). O *recall* R é definido como $R = \frac{TP}{TP + FN}$, onde *FN* são os *false negatives* (falsos negativos – resultados faltantes).

Logo, a *F1-measure* é obtida substituindo x_1 por *P* e x_2 por *R*, resultado em $F1 = \frac{\left(\sqrt{P * R}\right)^2}{\left(\frac{P+R}{2}\right)} = 2\frac{P * R}{P + R}$.

4. A Ferramenta *BENCH*

Este capítulo apresenta a ferramenta *Bench* e aponta a sua integração com o *MatchMaking* (GOMES, 2010). A seção 4.1 aborda os requisitos que devem ser atendidos pelo *Bench*. A seção 4.2 detalha a arquitetura do *Bench.* Por fim, a seção 4.3 delineia a implementação do *Bench*.

4.1 Requisitos do *Bench*

O principal problema a ser resolvido pela ferramenta *Bench* consiste na geração de esquemas conceituais, populados com dados, e alinhamentos entre eles de tal forma que a ferramenta possa ser utilizada para testes de algoritmos de alinhamento. Diferente de ser um simples *benchmark* para algoritmos de alinhamento, o *Bench* permite de fato gerar benchmarks diversificados e avaliar, de forma unificada e sob o mesmo critério, algoritmos de alinhamento.

O *Bench* está integração com o *MatchMaking* (GOMES, 2010), possibilitando assim uma estrutura para desenvolvimento e avaliação de algoritmos de alinhamento de forma mais completa.

Seguindo o modelo de caso de uso apresentado na Figura 16, os principais requisitos atendidos pelo *Bench* são:

- Importar *datasets*: Importar de *datasets* em formato RDF/OWL estrutura e dados.
- Gerar esquemas sintéticos: A partir de um esquema conceitual inicial, criar esquemas através da aplicação de transformações, armazenando os alinhamentos entre os esquemas indizidos pelas transformações aplicadas.
- Avaliar algoritmos: Implementar o cálculo da exatidão de algoritmo de alinhamento.
- Gerar relatórios de análise: Apresentar o resultado da avaliação de um dado algoritmo de alinhamento.
- Integração com o *MatchMaking*.

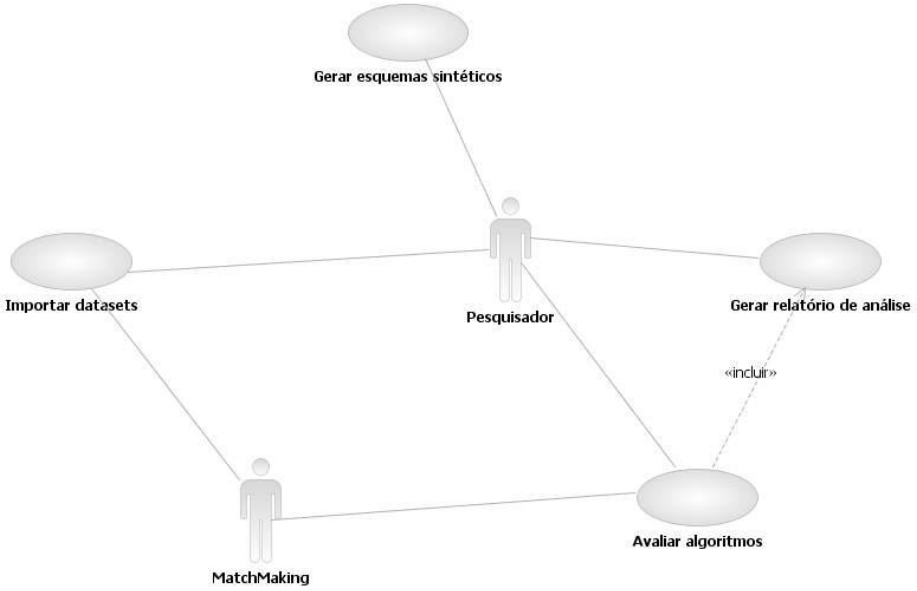

Figura 15 - Modelo de Caso de Uso do *Bench*

4.2 Arquitetura do *Bench*

A arquitetura do *Bench* foi projetada em harmonia com a arquitetura do *MatchMaking* (GOMES, 2010), com o intuito de criar uma solução integrada para desenvolvimento e avaliação de algoritmos de alinhamento.

A Figura 16 apresenta o modelo arquitetural proposta em (GOMES, 2010).

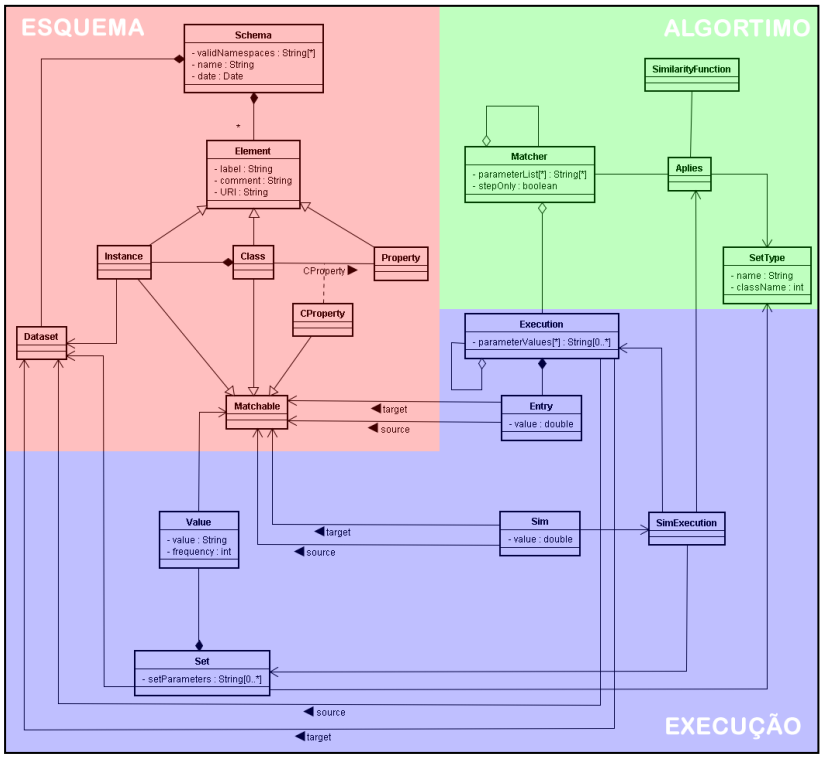

Figura 16 - Divisão da arquitetura do MatchMaking

Um esquema (*Schema)* é definido como uma agregação de elementos: classes, propriedades e instâncias (*class, property* e *instance*, respectivamente). Uma propriedade contextualizada (*CProperty*) define um relacionamento entre classes e propriedades, capturando o fato de que uma mesma propriedade pode estar definida para mais de uma classe do esquema.

Em cada esquema, podemos ter vários conjuntos de dados (*Datasets*) que representam instâncias de classes obtidas em um determinado momento. O *Matchmaking* consegue controlar vários desses conjuntos de forma independente. Nas técnicas de alinhamento baseadas em instâncias, ou nas técnicas híbridas, os valores das propriedades de cada uma das instâncias de classe podem alterar o resultado do alinhamento. Portanto, é importante diferenciar os conjuntos de dados que foram utilizados em um alinhamento.

O *Bench* utiliza as classes que representam esquemas (representadas em cor-de-rosa na Figura 17). O *Bench* implementa, além da importação de esquemas já oferecida pelo *MatchMaking*, a importação de dados, ou seja, dos indivíduos contidos no *dataset* RDF/OWL. A Figura 17 representa as novas classes do *Bench*.

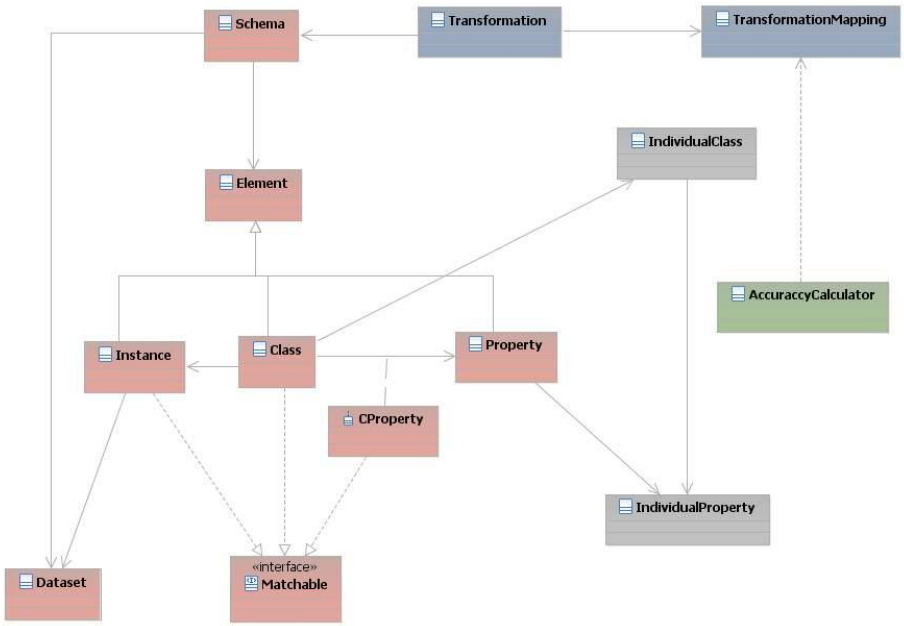

Figura 17 - *Bench* estendendo o *MatchMaking*

As classes *IndividualClass* e *IndividualProperty* são responsáveis por tratar cada instância de um determinado *dataset*, ou seja, elas irão manipular as instâncias das classes e das propriedades, respectivamente.

A classe *Transformation* é responsável pela transformação nos esquemas, ou seja, é a classe que, ao ser instanciada, registra qual transformação será executada e quais as classes (do esquema conceitual) estão envolvidas. Assim, ela permite executar as transformações do esquema de origem para o esquema de destino, aplicando-se uma das transformações citadas na Seção 3.2.

A classe *TransformationMapping* armazena, no banco de dados interno do *Bench*, as transformações do esquema de origem para o esquema de destino, ou seja, os alinhamentos induzidos pelas transformações.

Por fim, a classe *AccuracyCalculator* calcula a exatidão do algoritmo em avaliação pelo *Bench*. Os métodos desta classe permitem identificar os alinhamentos corretos, os alinhamentos errados e os alinhamentos esperados, a partir dos alinhamentos salvos pela *TransformationMapping* e dos alinhamentos obtidos pelo algoritmo de alinhamento de esquemas submetido ao *Bench*. De posse destes alinhamentos, o valor de exatidão para o algoritmo de alinhamento de esquemas submetido pode então ser calculado.

4.3 Implementação do *Bench*

A implementação do *Bench* segue a proposta de arquitetura descrita na seção anterior, enfatizando a integração com a infraestrutura do *MatchMaking*. Desta forma, esta seção tratará cada pacote e a funcionalidade implementada.

Os pacotes principais são:

- **bench.beans**: apresentam as classes de *beans*. Ou seja, são as classes utilizadas para o transporte de dados entre os pacotes e aplicativos terceiros.
- **bench.commands**: é a superfície do sistema. Aqui se encontram todos os comandos disponíveis para os aplicativos terceiros.
- **bench.schema**: representa um esquema alinhável. As classes desse pacote são utilizadas pelos algoritmos de alinhamento para obter os dados de um determinado esquema.

- **bench**.**engine**: é neste pacote que esse encontra a engrenagem que irá realizar os cálculos de avaliação de algoritmos.
- **bench.dao**: é a camada de acesso aos dados da ferramenta. Todas as consultas ao banco de dados são realizadas a partir desse pacote.
- **bench**.**utils**: pacote onde ficam funcionalidades básicas que são utilizadas pelos demais pacotes.
- **bench**.**configuration**: retrata as configurações gerais para utilização do benchmark.

Este pacotes seguem a proposta de organização do MatchMaking, incluindo apenas o pacote *engine*. A partir desta estrutura, as classes implementadas foram devidamente localizadas de acordo com a responsabilidade de cada pacote.

As seções seguintes detalham a implementação e estão organizadas de acordo com a funcionalidade. A última seção aborda como os pacotes/classes são utilizadas por toda estrutura.

4.3.1 Importação de *datasets*

O primeiro passo para que seja possível gerar esquemas sintéticos é justamente a importação de *datasets*. Através desta funcionalidade implementada no *Bench*, torna-se exequível a importação de *datasets* no formato RDF/OWL de qualquer esquema. Assim, é aberta a possibilidade de se trabalhar com quaisquer conjuntos de dados, independente de que conceitos estes dados tratam.

Esta importação disponibiliza no banco de dados relacional não somente o esquema RDF, como também as instâncias. Assim, o esquema pode ser trabalhado de diversas formas.

Uma vez importado um ou mais *datasets*, é possível gerar os esquemas sintéticos conforme descrito a seguir.

Com objetivo de implementar a importação dos dados de *datasets*, foi necessário realizar uma extensão na funcionalidade do *MatchMaking* de importar esquema. Sendo assim, foi implementada uma sobrescrita da classe *NewSchema*.

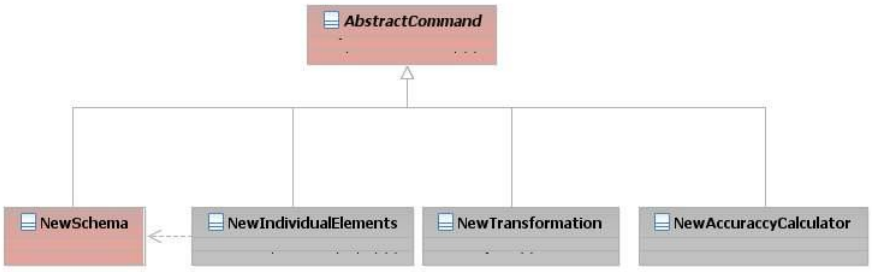

Figura 18 - Extensão do *MatchMaking* para implementação de comandos

Em geral, toda execução do *Bench* é implementada a partir da classe abstrata *AbstractCommand* fornecida pelo *MatchMaking* conforme mostrado na Figura 18.

Cada classe herdada de *AbstractCommand*, é responsável pelo controle de fluxo da execução de cada caso de uso, ou seja, as classes *NewAccuraccyCalculator*, *NewSchemaIndividuals* e *NewTransformation* irão garantir a execução das operações de cálculo de exatidão do algoritmo, da importação de *datasets* (estrutura e dados) e da geração de esquemas sintéticos, respectivamente.

A importação de *datasets* acontece em 2 etapas, executadas por duas classes distintas. A primeira etapa é a importação do esquema, ou seja, a importação das classes e propriedades. Com isso, é obtida a estrutura utilizada pelo *dataset* importado para armazenar as informações. Estas estruturas são representadas pelas classes Clss e Property, com a classe CProperty representando a ligação que cada classe tem com sua respectiva propriedade. De posse destas estruturas importadas é possível a realização de testes com algoritmos de alinhamentos baseados em estrutura, ou seja, aqueles que não se utilizam das instâncias para realizar o processo de alinhamento de esquemas.

No entanto, a proposta deste trabalho não é possibilitar aferição de um escopo reduzido de algoritmos, e sim de um contexto bem maior. Para isso se faz necessário a execução da segunda etapa do processo de importação de *dataset*, que é justamente o processo de importação dos dados que populam as estruturas disponibilizadas.

Neste segundo passo, a responsabilidade da importação dos indivíduos é justamente da classe *NewSchemaIndividuals*, pois a partir da herança obtida pela extensão da classe *NewSchema*, foi criado um método para realização de

importação das instâncias, ou seja, dos indivíduos. Na operação de importação de indivíduos, os dados referentes a cada instância são armazenados nas classes *IndividualClass* e *IndividualProperty*.

Com essas duas etapas, é possível recriar o *dataset* importado de forma completa, ou seja, contemplando sua estrutura e seus dados. Assim é viável trabalhar com diferentes tipos de algoritmos de alinhamento de esquemas, independente de sua estratégia de atuação.

4.3.2 Geração de Esquemas Sintéticos

Através do *Bench*, é possível gerar esquemas sintéticos, utilizando as transformações. Uma vez obtido um conjunto suficiente de esquemas, populados com dados, é possível então avaliar algoritmos de alinhamento.

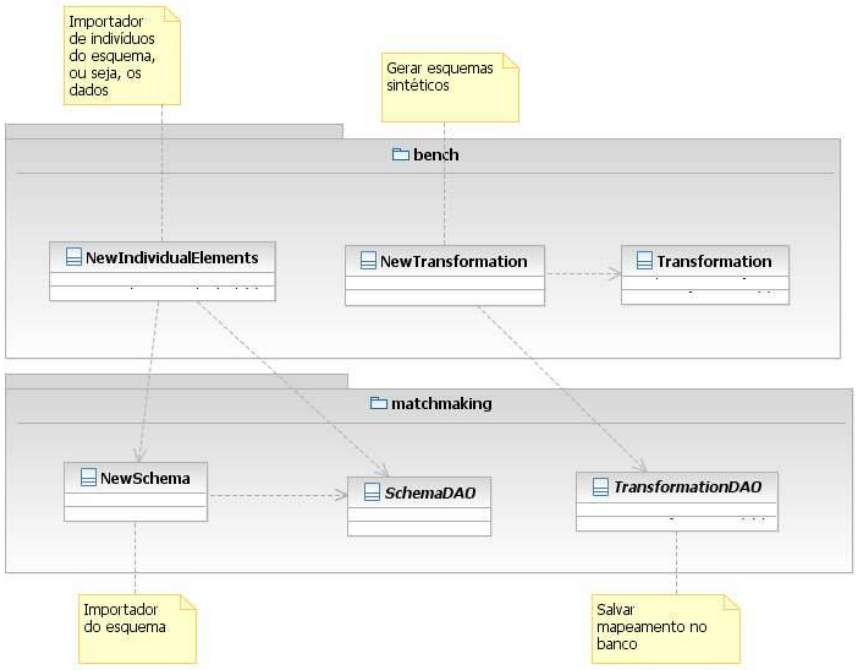

Figura 19 - Arquitetura do *Bench* integrada ao *MatchMaking* - Geração de esquemas sintéticos

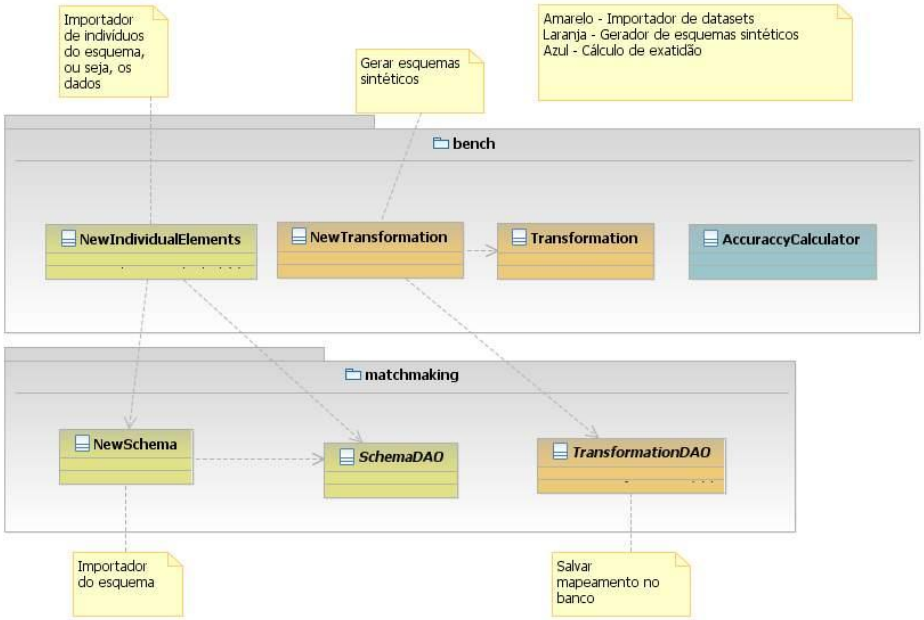

Figura 20 - Representação dos módulos do *Bench*

A Figura 19 e a Figura 20 mostram a arquitetura do *Bench* integrada à do *MatchMaking* (GOMES, 2010).

A classe *NewTransformation* é responsável pelo processo de geração de esquemas sintéticos a partir do esquema de origem é de responsabilidade. Este processo tem como pré-requisito a importação de *datasets* descrita na seção anterior, pois é justamente a partir destes *datasets* importados que serão gerados esquemas sintéticos. O processo começa com a seleção das transformações a serem aplicadas ao esquema de origem. Esta escolha pode ser feita pelo usuário do *Bench* ou pode ser feita de forma randômica pela própria ferramenta. Inicialmente o *Bench* foi disponibilizado com um determinado conjunto de transformações. Porém, nada impede de que novas transformações sejam acrescentadas ao *benchmark* estendendo, desta forma, as possibilidades.

Uma vez que são elencadas quais classes e quais transformações serão aplicadas, a classe *NewTransformation* recebe estes dados e dá início ao processo de gerações de esquemas sintéticos. No *Bench* a geração de esquemas sintéticos foi implementado a partir da criação de *views* sobre o banco de dados relacional que armazena o *dataset* importado. Desta forma, além de atingir o objetivo de se ter variações de determinados esquemas, ganha-se também no armazenamento, uma vez que os dados não precisam ser replicados, o que aumentaria consideravelmente o consumo de espaço de armazenamento.

A criação das *views* é solicitada pela classe *NewTransformation*. Porém, como se trata de uma operação a ser realizada diretamente no banco de dados, essa responsabilidade é delegada ao DAO. Os detalhes de implementação do DAO e suas diferentes implementações para cada banco de dados serão descritos na Seção 4.3.4. Os esquemas gerados ficam disponíveis em um banco de dados conforme modelo ilustrado na Figura 22 da Seção 4.3.4.

A classe *TransformationMapping* é responsável por persistir cada transformação no banco de dados de forma que seja possível saber exatamente a quais transformações o esquema de origem foi submetido, ou seja, quais alinhamentos devem ser encontradas pelo algoritmo de alinhamento submetido a avaliação.

Em resumo, a geração dos esquemas sintéticos consiste em criar *views* sobre os *datasets* importados utilizando as transformações.

4.3.3 Avaliação de Algoritmos de Alinhamento de Esquemas

A terceira das principais funcionalidades oferecidas pelo *Bench* é a avaliação dos algoritmos de alinhamento. Esta funcionalidade é basicamente implementada pela classe *NewEvaluation*, que é responsável por receber os alinhamentos obtidos pelo algoritmo de alinhamento de esquemas e identificar os parâmetros que serão necessários para a classe *AccuraccyCalculator* realizar o cálculo de exatidão do algoritmo (ver Seção 3.4).

A classe *NewEvaluation* irá identificar os parâmetros a partir da comparação entre os alinhamentos obtidos pelo algoritmo e os alinhamentos salvos no momento da geração do esquema sintético. Finalmente ela irá repassar estes dados para o *AccuraccyCalculator* realizar o cálculo de exatidão (novamente ver Seção 3.4).

4.3.4 Camada de Persistência

Inicialmente, a camada de persistência foi estendida a partir da infraestrutura MatchMaking, oferecendo a possibilidade de trabalhar não somente com o Postgres, implementado no MatchMaking, mas também com banco de dados Oracle e MySQL. Desta forma, conforme se observa na Figura 21, toda implementação foi feita tanto para o banco de dados Oracle, quanto para o MySQL.

A Figura 21 apresenta a implementação da camada de persistência.

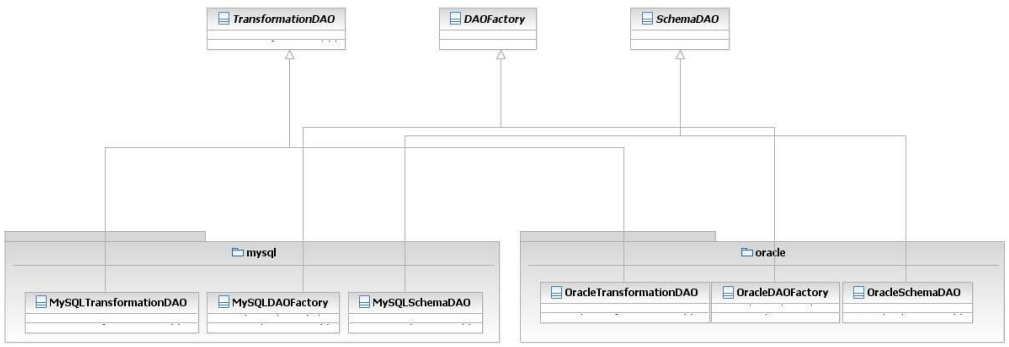

Figura 21 - Camada de persistência do *Bench*

A camada de persistência pode ser resumida nas três interfaces representadas, *TransformationDao*, *DaoFactory* e *SchemaDao.*

A *TransformationMapping* é responsável por garantir a persistências dos alinhamentos induzidos pelas transformações indicadas na Seção 4.3.2. Conforme já citado, ela irá armazenar a origem, destino e a regra de transformação aplicada na tabela *TRANSFORMATIONS*. Com isso, será possível reconhecer as transformações aplicadas para gerar cada esquema sintético.

A classe *DaoFactory* é a implementação do *Design Pattern Factory*, pois é a partir dela que todas os demais DAOs serão criados.

A *SchemaDao* é uma complementação da implementação feita pelo MatchMaking com a possibilidade de importação dos indivíduos dos *datasets*. Esta classe irá popular a maior parte das tabelas do banco de dados, uma vez que será ela a responsável por persistir toda informação de esquemas.

Por fim o banco de dados foi implementado com as mesmas extensões comentadas anteriormente, ou seja, incluindo tabelas para armazenar dados dos indivíduos dos arquivos RDF/OWL além de uma tabela para armazenar dados do mapeamento. A Figura 22 ilustra o modelo de dados relacional do *Bench*.

A implementação do *Bench* foi planejada de forma extensível permitindo que novas funcionalidades sejam agregadas, contanto que respeitada a arquitetura proposta pelo *MatchMaking*. Desta forma, tanto o próprio *MatchMaking* quanto o *Bench* são extensíveis a novas funcionalidades.

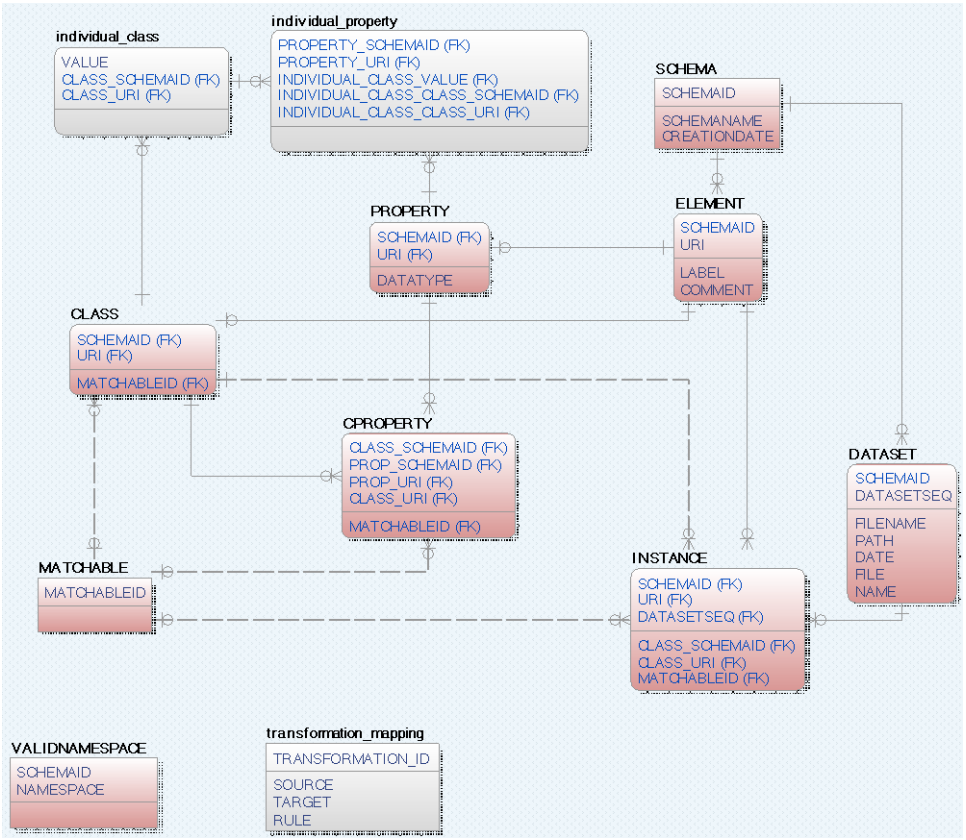

Figura 22 - Modelo de Dados do *Bench*

4.4 Exemplo de uso do *Bench*

Com o objetivo de gerar um *benchmark* é necessário fazer uso de bases de dados/*datasets* para que possam ser aplicadas as transformações, e assim gerar os esquemas diversos para que possam ser testados os algoritmos de alinhamento. Em particular, a *Linked Data Cloud* oferece um grande conjunto de *datasets* livre de licença em RDF/OWL.

Para exemplificar o uso do *Bench*, utilizamos dados extraídos do eBay, organizados de acordo com o esquema da Figura 23.

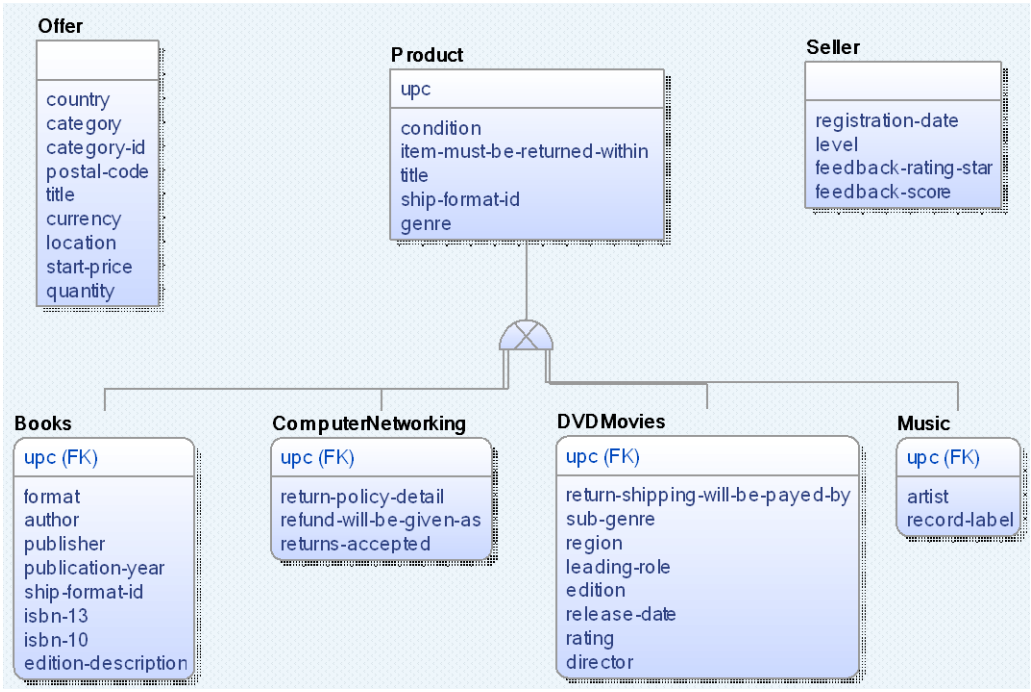

Figura 23 - Modelo do eBay

Um conjunto de triplas do *dataset* é exibido abaixo:

```
<rdf:Description rdf:about="http://www.ebay.com/ontology#_180288562627">
    <product rdf:nodeID="A0"/>
    <rdf:type rdf:resource="http://www.ebay.com/ontology#Offer"/>
    <product rdf:nodeID="A1"/>
    <category-id rdf:datatype="http://www.w3.org/2001/XMLSchema#string">378</category-id>
    <country rdf:datatype="http://www.w3.org/2001/XMLSchema#string">GB</country>
    <currency rdf:datatype="http://www.w3.org/2001/XMLSchema#string">USD</currency>
    <category rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      Books:Nonfiction Books
    </category>
    <location rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      United Kingdom
    </location>
    <title rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      William Shakespeare | King Henry VI Part 2: Pt. 2 | NEW
    </title>
    <start-price rdf:datatype="http://www.w3.org/2001/XMLSchema#double">17.11</start-price>
    <dispatch-time-max rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
      5
    </dispatch-time-max>
    <seller rdf:resource="http://www.ebay.com/ontology#thesaintbookstore"/>
    <quantity rdf:datatype="http://www.w3.org/2001/XMLSchema#int">15</quantity>
  </rdf:Description>
```

Este conjunto ilustra, por exemplo, que o registro é uma oferta (*Offer*) que pertence ao país GB (Grã-Bretanha).

O *dataset* exposto acima foi utilizado para testar a habilidade do *Bench* em importar esquemas, partindo da execução da funcionalidade exposta no item 4.3.1.

Após a importação realizada pelo Bench é possível realizar uma consulta na tabela CLASS e identificar as classes importadas, tais como, Offer, Books entre outras (Figura 24). Esta base dados importada será utilizada não somente como modelo referência para comparação dos esquemas, mas também é a partir deste modelo que os demais esquemas sintéticos serão gerados.

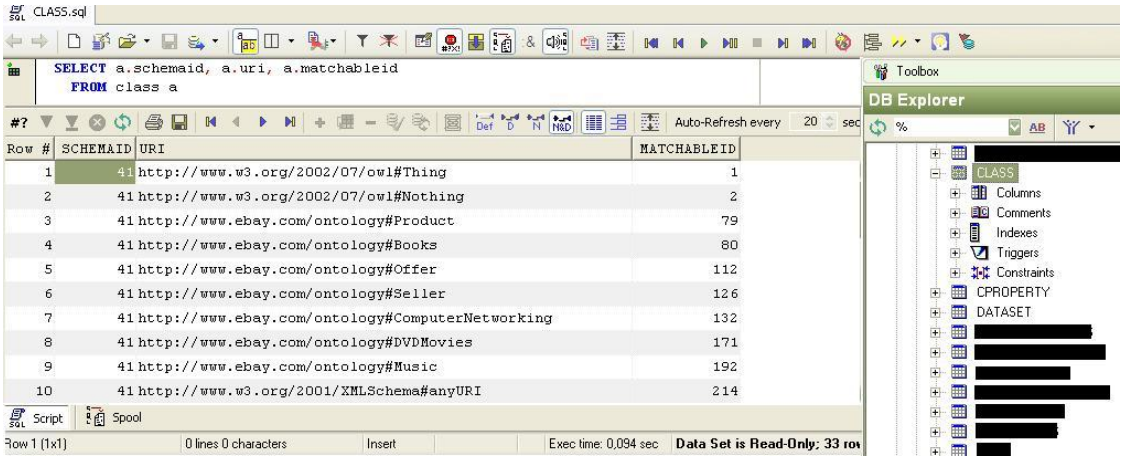

Figura 24 - Demonstração do dataset importado

Com a execução da funcionalidade de geração de esquema sintéticos percebe-se na Figura 25 as views criadas representando os esquemas gerados.

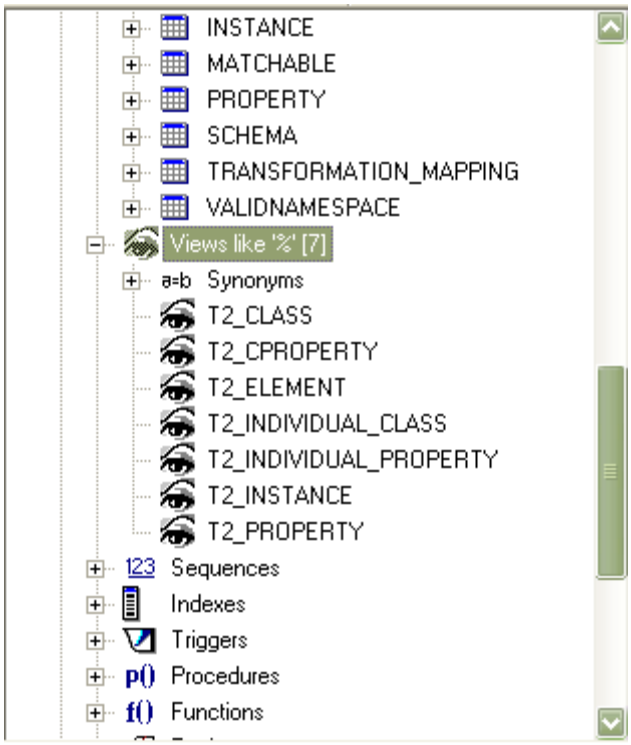

Figura 25 - Views geradas pelo Bench

As views foram criadas utilizando como padrão de nomenclatura a identificação da transformação em seu nome, assim, é possível identificar todo esquema sintético criado.

Conforme já citado, a construção da view por si só promove a transformação do esquema original no esquema sintético. Exemplo disso é o código-fonte de uma das views criadas dinâmicamente demonstradas na figura abaixo:

```
CREATE OR REPLACE VIEW t2_individual_class
(value, class_schemaid, class_uri )
AS
(SELECT "VALUE", CLASS_SCHEMAID,
REPLACE(CLASS_URI, 'Offer', 'Product') "CLASS_URI" FROM
(INDIVIDUAL_CLASS))
```

Neste exemplo, a transformação executada foi a geração de um esquema sintético onde a entidade originalmente Offer foi generalizada para Product no esquema sintético.

Com essa transformação, pode-se perceber que na tabela INDIVIDUAL_CLASS, por exemplo, trupla encontra-se renomeado na *view* criada.

```
SELECT a.value, a.class_schemaid, a.class_uri
  FROM individual_class a
```

#	VALUE	CLASS_SCHEMAID	CLASS_URI
1	_110110127882	70	http://www.ebay.com/ontology#Offer

Figura 26 - Entidade no esquema original

```
SELECT a.value, a.class_schemaid, a.class_uri
  FROM t2_individual_class a
  where value = '_110110127882'
```

#	VALUE	CLASS_SCHEMAID	CLASS_URI
1	_110110127882	70	http://www.ebay.com/ontology#Product

Figura 27 - Entidade no esquema sintético

Desta mesma forma, a transformação ocorre em todo o modelo, permitindo desta forma que seja possível obter um modelo conceitual sintético a partir do modelo originalmente importado.

No processo de transformação, as regras aplicadas são registradas no banco de dados, conforme visto na figura abaixo, onde descreve que a entidade Offer foi transformada na entidade Product utilizando a regra de transformação GENERALIZATION.

```
SELECT a.transformation_id, a.source, a.target, a.rule
  FROM transformation_mapping a
  where transformation_id = 2
```

#	TRANSFORMATION_ID	SOURCE	TARGET	RULE
1	2	Offer	Product	GENERALIZATION

Figura 28 - Mapa identificando a transformação

5. CONCLUSÃO

Esta dissertação abordou a criação de uma ferramenta, chamada de *Bench*, para gerar *benchmarks* para avaliação de algoritmos de alinhamento de esquemas.

Em mais detalhe, a dissertação introduziu uma ferramenta para benchmark, simplesmente chamado de *Bench*, que possibilita a importação de qualquer conjunto de dados com um esquema conceitual bem definido e oferece facilidades para gerar variações deste esquema através de transformações que refletem alternativas de projeto comumente encontradas. Estas transformações de fato definem alinhamentos de referência entre o esquema original e suas variantes. O *Bench* permite ainda calcular o desempenho de um algoritmo de alinhamento de esquemas submetido para testes, comparando-se os alinhamentos de referência com os alinhamentos encontrados pelo algoritmo em teste.

Por fim, uma característica interessante do *Bench* é a sua integração com o *MatchMaking*, através da extensão da infraestrutura oferecida por esta segunda ferramenta. Desta forma, o *MatchMaking* em conjunto com o *Bench* oferecem uma solução completa para desenvolvimento, teste e avaliação de algoritmos para alinhamento de esquemas.

5.1 Trabalhos futuros

Como trabalhos futuros sugere-se algumas opções. A primeira delas seria possibilitar o *Bench* trabalhar com banco de dados puramente triplificado, como por exemplo o Virtuoso. Assim, a disponibilização dos esquemas sintéticos se dariam a partir da criação de visões utilizando consultas SPARQL. Desta forma, não seria necessário despender tanto custo para realização da importação dos *datasets* RDF/OWL para o banco de dados relacional atualmente utilizado. Uma outra proposta de trabalho seria viabilizar no *Bench* uma forma de tornar a geração do esquema sintético uma geração com maior significado semântico, pois a geração, da forma que está implementada, seleciona aleatoriamente as transformações e as aplica no esquema original. Com a proposta de realizar

transformações com semântica, os esquemas sintéticos continuariam sendo gerados automaticamente, porém, as transformações poderiam ser mais coerentes em relação ao modelo gerado.

6. BIBLIOGRAFIA

ALEXE, B. et al. **STBenchmark:** Towards a Benchmark for Mapping Systems. VLDB Endowment. Auckland, New Zealand: [s.n.]. 2008.

BECHHOFER, S. et al. OWL Web Ontology Language Reference. **W3C Recommendation**, 2004. Disponivel em: <http://www.w3.org/tr/owl-ref>. Acesso em: Abril 2012.

BERNERS-LEE, T. Linked Data. **W3C**, 2006. Disponivel em: <http://www.w3.org/DesignIssues/LinkedData.html>. Acesso em: Abril 2012.

BILKE, A.; NAUMANN, F. **Schema matching using duplicates**. Proc. of the 21st Int'l. Conf. on Data Engineering. [S.l.]: [s.n.]. 2005. p. 69-80.

BIZER, C. et al. How to Publish Linked Data on the Web, 2007. Disponivel em: <http://www4.wiwiss.fuberlin.de/bizer/pub/LinkedDataTutorial/>. Acesso em: Junho 2012.

BIZER, C. et al. **Linked Data on the Web**. Workshop at the 17th International World Wide Web Conference. Beijin: [s.n.]. 2008.

BRAUNER, D. F. et al. **An instance-based approach for matching export schemas of geographical database Web services**. Proc. of the IX Brazilian Symp. on GeoInformatics (GEOINFO). [S.l.]: [s.n.]. 2007. p. 109-120.

BREITMAN, K. K. et al. Semantic Web: concepts, technologies and applications. **Springer**, Londres, 2007.

CASANOVA, M. A. et al. Database Conceptual Schema Matching. **Computer**, n. 40, p. 102-104, 2007.

DO, H.-H. et al. Comparison os Schema Matching Evaluations. **Lecture Notes in Computer Science**, 2003. 221-237.

DUCHATEAU, F. et al. Xbenchmatch: a benchmark for xml schema. **The VDBL Journal**, 2007. 1318–1321.

EUZENAT, J. et al. **Final results of the Ontology Alignment Evaluation Initiative 2011**. Ontology Alignment Evaluation Initiative. Prague: [s.n.]. 2011.

EUZENAT, J.; SHVAIKO, P. Ontology matching. **Springer-Verlag**, 2007.

FORMAN, G. **An Extensive Empirical Study of Feature Selection Metrics for Text Classification**. HP Laboratories. Palo Alto, CA. 2002.

GOMES, R. V. A. Matchmaking – Uma infraestrutura para alinhamento de esquemas, Rio de Janeiro, 2010.

GUO, C. et al. **MatchBench: Benchmarking Schema Matching Algorithms for Schematic Correspondences**. University of Manchester. Manchester.

HAY, D. C. **Data Model Patterns**. New York: Dorset House Publishing, 1995.

KLYNE, G. et al. Resource Description Framework (RDF): Concepts and abstract syntax. **W3C Recommendation**, 2004. Disponivel em: <http://www.w3.org/tr/rdf-concepts>. Acesso em: Março 2012.

LAVIE, A. et al. **The Significance of Recall in Automatic Metrics for MT Evaluation**. 6th Conference of the Association for Machine Translation in the Americas. Washington, DC: [s.n.]. 2004. p. 134-143.

LEME, L. A. P. et al. **Instance-based OWL Schema Matching**. 11TH INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS. Milão: [s.n.]. 2009.

LEME, L. A. P. P. **Conceptual schema matching based on similarity heuristics**. Rio de Janeiro. 2009.

MADHAVAN, J. et al. **Web-scale data integration:** You can afford to Pay as You Go. Proc. of the 3rd Biennial Conf. on Innovative Data Systems Research. [S.l.]: [s.n.]. 2007. p. 342-350.

MELNIK, S. et al. **Similarity Flooding: A Versatile Graph Matching Algorithm**. Stanford University. California. 2001.

MORGADO, C. S. A. **Ferramenta para teste de desempenho de algoritmos de alinhamento de esquemas**. Rio de Janeiro: [s.n.]. 2010.

QUINE, W. V. O. Ontological Relativity. **The Journal of Philosophy**, LXV, n. 7, 4 Abril 1968. 185-212.

RAHM, E.; BERNSTEIN, P. A. A survey of approaches to automatic schema matching. **The VLDB Journal 10**, n. 4, 6 Setembro 2001. 334-350.

SAUERMANN, L. et al. Cool URIs for the Semantic Web. **W3C**, 2007. Disponivel em: <http://www.w3.org/TR/cooluris/>. Acesso em: Junho 2012.

SAYYADIAN, M. et al. **Tuning Schema Matching Software using Synthetic Scenarios**. 31st VLDB Conference. Trondheim: [s.n.]. 2005.

WIKIPEDIA. Wikipedia (Benchmark_(computacao). **Wikipedia**, 2012. Disponivel em: <http://pt.wikipedia.org/wiki/Benchmark_(computação)>. Acesso em: Junho 2012.