Geração Automática de Mapeamentos R2RML

Vânia Maria P. Vidal[1], José Maria Monteiro[1], Luís Eufrasio Teixeira Neto[1]

[1] Universidade Federal do Ceará
{vvidal,zemaria,luiseufrasio}@lia.ufc.br

Abstract. This paper proposes an approach to generate RDF views of relational data, using SQL views. The paper shows a set of correspondence assertions to specify the RDF view in terms of the relational schema and shows how the mappings defined by the view correspondence assertions could be transformed in R2RML mapping.

Resumo. Este artigo propõe uma abordagem para gerar visões RDF a partir de dados relacionais, usando visões SQL. O artigo mostra um conjunto de assertivas de correspondência que especifica a visão RDF em termos do esquema relacional e mostra como os mapeamentos definidos pelas assertivas podem ser transformados em um mapeamento R2RML.

Categories and Subject Descriptors: H. Information Systems [**H.m. Miscellaneous**]: Databases

Keywords: SBBD, template

1. INTRODUÇÃO

R2RML [Das et al. 2012] é uma linguagem para expressar mapeamentos customizados de bancos de dados relacionais para datasets RDF [Lassila et al. 1999]. Tais mapeamentos permitem a visualização de dados relacionais existentes na forma de um modelo de dados RDF, cuja estrutura e vocabulário alvo são escolhidos pelo autor do mapeamento.

Todo mapeamento R2RML é criado para um esquema relacional e um vocabulário alvo específicos. A entrada para um mapeamento R2RML é um banco de dados relacional que está de acordo com o esquema. A saída é um dataset RDF que utiliza termos definidos no vocabulário alvo.

No entanto, para criarmos mapeamentos R2RML precisamos ter conhecimentos avançados da linguagem R2RML e isso consome tempo. Além disso, os usuários terão que redefinir os mapeamentos R2RML sempre que ocorrerem mudanças no esquema da base relacional. Portanto, devem ser desenvolvidas ferramentas que facilitem a tarefa de criação e manutenção de mapeamentos.

Propomos neste artigo uma abordagem onde o mapeamento R2RML mapping é derivado das assertivas de correspondência, que especificam relacionamentos entre o esquema da ontologia alvoe o esquema do banco relacional.

As principais contribuições desse artigo são: um algoritimo que gera os mapeamentos R2RML a partir das assertivas de correspondência e a ferramenta "R2RML Mappings-By-Assertions" que facilita a criação das assertivas de correspondência e a geração dos mapamentos R2RML.

Esse artigo está organizado como segue. A Seção 2 discute visões R2RML. A Seção 3 apresenta as assertivas de correspondência. A Seção 4 discute como gerar visões RDF usando as assertivas. A Seção 5 apresenta o algoritmo que automaticamente gera os mapeamentos R2RML referenciando as visões RDF. Finalmente, a Seção 6 apresenta as conclusões.

2. VISÕES R2RML

Conforme definido em [Das et al. 2012], um mapeamento R2RML referencia tabelas lógicas para consultar dados no banco relacional. Existem três tipos de tabelas lógicas: uma tabela relacional, uma visão SQL e uma consulta SQL válida chamada de "Visão R2RML". Esta última é criada dentro do mapeamento para emular uma visão SQL sem precisar alterar o banco de dados.

Na nossa abordagem criaremos visões RDF que tanto podem ser visões SQL como visões R2RML. A criação dessas visões é feita através do algoritmo descrito na seção 4 que recebe como entrada um conjunto de assertivas de correspondência geradas conforme definido na próxima seção.

A vantagem de usarmos visões RDF nos mapeamentos é que resolvemos os problemas de heterogeneidade entre os modelos usando cláusulas SQL, ou seja, transferimos a complexidade do mapeamento para as visões. Assim, a geração automática do arquivo de mapeamento R2RML torna-se viável e de fácil implementação conforme veremos na seção 5.

3. CRIANDO ASSERTIVAS DE CORRESPONDÊNCIA

A criação de assertivas de correspondência é uma forma de mapear o vocabulário de uma ontologia alvo nos termos do vocabulário de uma ontologia fonte.

Nesse trabalho implementamos uma ferramenta gráfica chamada "R2RML Mapping-By-Assertions" que permite ao usuário importar a ontologia OWL alvo, gerar a ontologia OWL fonte a partir do esquema do banco de dados, criar as assertivas por meio de correspondências entre Classes, Propriedades de Dados e Propriedades de Objetos, gerar as Visões RDF e gerar o mapeamento R2RML a partir das assertivas.

Definimos então um processo simples de criação das assertivas, com os seguintes passos:

1. Importa a ontologia OWL alvo;
2. Executa a geração da ontologia OWL fonte por meio de mapeamento direto usando o comando da plataforma D2RQ: `generate-mapping [parâmetros conexão banco] --w3c -v jdbcURL`;
3. Importa a ontologia OWL fonte gerada no passo 2;
4. Configura os parâmetros para conexão com o banco de dados fonte;
5. Cria as assertivas de correspondência entre as ontologias;

A Figura 1 mostra a geração de uma assertiva que mapeia a classe *Publicação* da ontologia *Vendas* nos termos da classe *Livro* da ontologia *Amazon*. A Tabela I mostra as assertivas criadas para o mapeamento dessas classes.

As assertivas podem ser classificadas conforme os casos descritos em [Sacramento et al. 2010b]. Esses casos são essenciais para os algoritmos de geração das visões e de geração do mapeamento a partir das visões. Nesse artigo dizemos que o caso ao qual a assertiva pertence é o tipo da assertiva.

De forma simplificada, existem 6 tipos de assertiva:

1. Mapeamento entre Classes;
2. Mapeamento de Propriedade de Dados para uma Constante;
3. Mapeamento direto de Propriedades (de Dados ou de Objeto);

4. Mapeamento indireto de Propriedades (de Dados ou de Objeto) através de um caminho na ontologia local;
5. Mapeamento indireto através de um caminho na ontologia de domínio, pode ser entre:

   5.1. Propriedades de Dados;

   5.2. Classes;

   5.3. Propriedades de Objetos.
6. Mapeamento indireto de Propriedades (de Dados ou de Objeto) através de caminhos na ontologia local e de domínio.

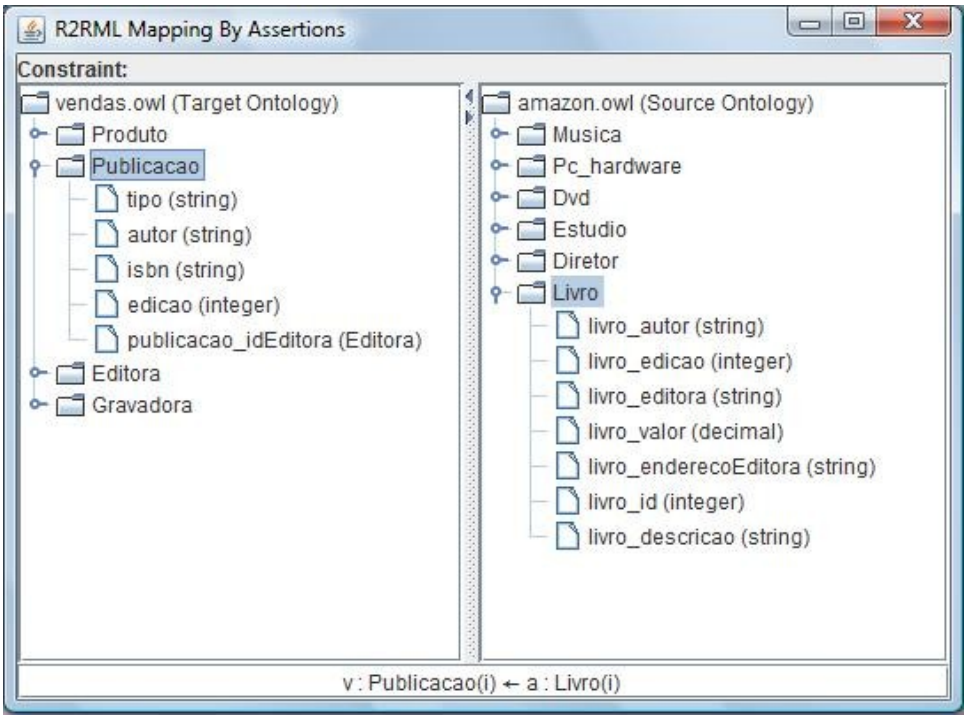

Fig. 1. Exemplo de Criação de uma Assertiva

Tabela I. Assertivas Criadas para Mapear *v:Publicacao* em termos de *a:Livro*

#	**Assertiva**
1	*v:Publicacao(i) ← a:Livro(i)*
2	*v:autor(i, x) ← a:livro_autor(i, x), a:Livro(i)*
3	*v:isbn(i, x) ← a:livro_id(i, x), a:Livro(i)*
4	*v:edicao(i, x) ← a:livro_edicao(i, x), a:Livro(i)*
5	*v:Editora(fEditora(x)) ← a:livro_editora(i, x), a:Livro(i)*
6	*v:Editora(fEditora(x)) ← a:livro_editora(i, x), a:Livro(i)*
7	*v:publicacao_idEditora(i, fEditora(x)) ← a:livro_editora(i, x), a:Livro(i)*

4. GERANDO VISÕES RDF

Tomando como exemplo as assertivas da tabela I criamos as visões das classes *Publicação* e *Editora* seguindo os passos do algoritmo de geração de visões RDF ilustrado na figura 2.

As visões geradas estão listadas na figura 3. Elas podem ser criadas no banco de dados ou incluídas dentro do arquivo de mapeamento sem a necessidade de alterar o banco. No nosso algoritmo de geração do mapeamento mostrado na próxima seção utilizamos a segunda abordagem.

Entrada: Assertivas de Correspondência (Lista *L*)
Saída: Visão SQL

1. Sejam *select*, *from* e *where* strings de carasteres;
2. *select* := "SELECT "; *from* := "FROM "; *where* := "WHERE ";
3. Seja *C* uma Lista de Classes;
4. *C* := *agruparPorClassesDominio*(*L*);
5. **Para** cada classe *c* em *C* f**aça**
   5.1. Seja *A* uma lista de assertivas;
   5.2. *A* = *recuperarAssertivas*(*c*);
   5.3. **Para** cada assertiva *a* de *A* **Faça**
      1. **No caso de**
      2. **Caso** *tipo*(*a*) = '1' **Ou** *tipo*(*a*) = '5.2'
         1. *from* := *from* + *classeLocal*(*a*);
         2. **Se** *a* tem restrição **Então**
            1. *where* := *where* + *restriçãoLocal*(*a*);
         3. **fim-se**
      3. **Caso** *tipo*(*a*) = '3' **Ou** *tipo*(*a*) = '5.1' **Ou** *tipo*(*a*) = '5.3'
         1. *select* := *select* + *propriedadeLocal*(*a*);
      4. **Caso** tipo(a) = '4' **Ou** *tipo*(*a*) = '6'
         1. Sejam *subSelect*, *subFrom* e *subWhere* strings de caracteres;
         2. *subSelect* := "(SELECT " + *propriedadeLocal*(*a*); *subFrom* := " FROM "; *subWhere* := " WHERE ";
         3. Seja *P* uma lista de propriedades de objeto;
         4. *P* := *propriedadesCaminhoLocal*(*a*);
         5. **Para cada** propriedade *p* em *P* **Faça**
            1. *subFrom* := *subFrom* + *imagem*(*p*);
            2. Seja *q* a próxima propriedade de *p* em *P*;
            3. **Se** *q* é nulo **Então**
               1. *q* := *propriedadeLocal*(*a*);
            4. **fim-se**
            5. *subWhere* := *imagem*(*p*) + "." + *p* + " = " + *imagem*(*q*) + "." + *colunaPk*(*imagem*(*q*));
         6. **fim-para**
         7. *select* := *select* + *subSelect* + *subFrom* + *subWhere* + ")";
      5. **fim-caso**
   5.4. **fim-para**
6. **fim-para**
7. **retorne** select + from + where;

Fig. 2. Algoritmo de criação das visões SQL

Classe	Visão
Publicação	SELECT autor, id, edicao, editora FROM Livro
Editora	SELECT editora FROM Livro

Fig. 3. Visões SQL geradas a partir das assetivas da Tabela I

5. GERANDO O MAPEAMENTO R2RML

O último passo para geração dos mapeamentos é executar o algoritmo da Figura 4 para gerar o mapeamento final R2RML. Esse algoritmo se baseia no uso dos templates listados na Tabela II.

Dessa forma, para cada assertiva é aplicado o template de acordo com o seu tipo e o resultado é incluído no mapeamento. Os elementos do template **sublinhados e em negrito** são parâmetros de entrada.

Tabela II. Templates para geração do mapeamento R2RML

#	Template
1	<#**classeDominio**TriplesMap> rr:logicalTable <#**classeDominio**View>; rr:subjectMap [rr:template '**classeDominio**/{*colunaPk*(**classeLocal**)}'; rr:class **prefixo**:**classeDominio**;];
2	rr:predicateObjectMap [rr:predicate **prefixo**:**propriedadeDominio**; rr:objectMap [rr:constant '**valorConstante**'];];
3	rr:predicateObjectMap [rr:predicate **prefixo**:**propriedadeDominio**; rr:objectMap [rr:column '**propriedadeLocal**'];];
4	rr:predicateObjectMap [rr:predicate **prefixo**:**propriedadeDominio**; rr:objectMap [rr:parentTriplesMap <#**classeRef**TriplesMap>; rr:joinCondition [rr:child '**propriedadeLocal**'; rr:parent '**colunaChavePai**';];];];

6. CONCLUSÕES

REFERENCES

DAS SOURIPRIYA, SUNDARA SEEMA, CYGANIAK RICHARD, R2RML: RDB to RDF Mapping Language, W3C Working Draft. http://www.w3.org/TR/r2rml/, 2012.

LASSILA ORA, SWICK RALPH R., Resource Description Framework (RDF) Model and Syntax Specification, W3C Proposed Recommendation. http://www.w3.org/TR/PR-rdf-syntax/, 1999.

SACRAMENTO, E. R., VIDAL, V. M., MACÊDO, J. A., LÓSCIO, B. F., LOPES, F. L. R., CASANOVA, M. A., LEMOS, F. (2010b). Towards automatic generation of application ontologies. In Proceeding of the 12th International Conference on Enterprise Information Systems - ICEIS, Funchal, Madeira - Portugal.