UNIVERSIDADE FEDERAL DO CEARÁ

Uma Ferramenta para suportar a publicação de visões RDF de dados relacionais

Vânia Maria P. Vidal, José Maria Monteiro, Luís Eufrasio T. Neto

Departamento de Computação

Introdução

- Linked Data: padrão estabelecido como a melhor prática para expor, compartilhar e conectar dados na Web Semântica.
- Grande maioria dos dados corporativos estão armazenados em SGBDs relacionais.
- É preciso publicar os dados relacionais usando o modelo RDF.
- Como fazer isso?

Introdução

- A solução mais utilizada é criar mapeamentos de bases relacionais para ontologias.
- Ferramentas que utilizam essa abordagem: D2R Server, OpenLink Virtuoso, Spyder etc.
- Existe uma linguagem recomendada pela w3c para criação dos mapeamentos: R2RML.
- As ferramentas deverão suportar R2RML.

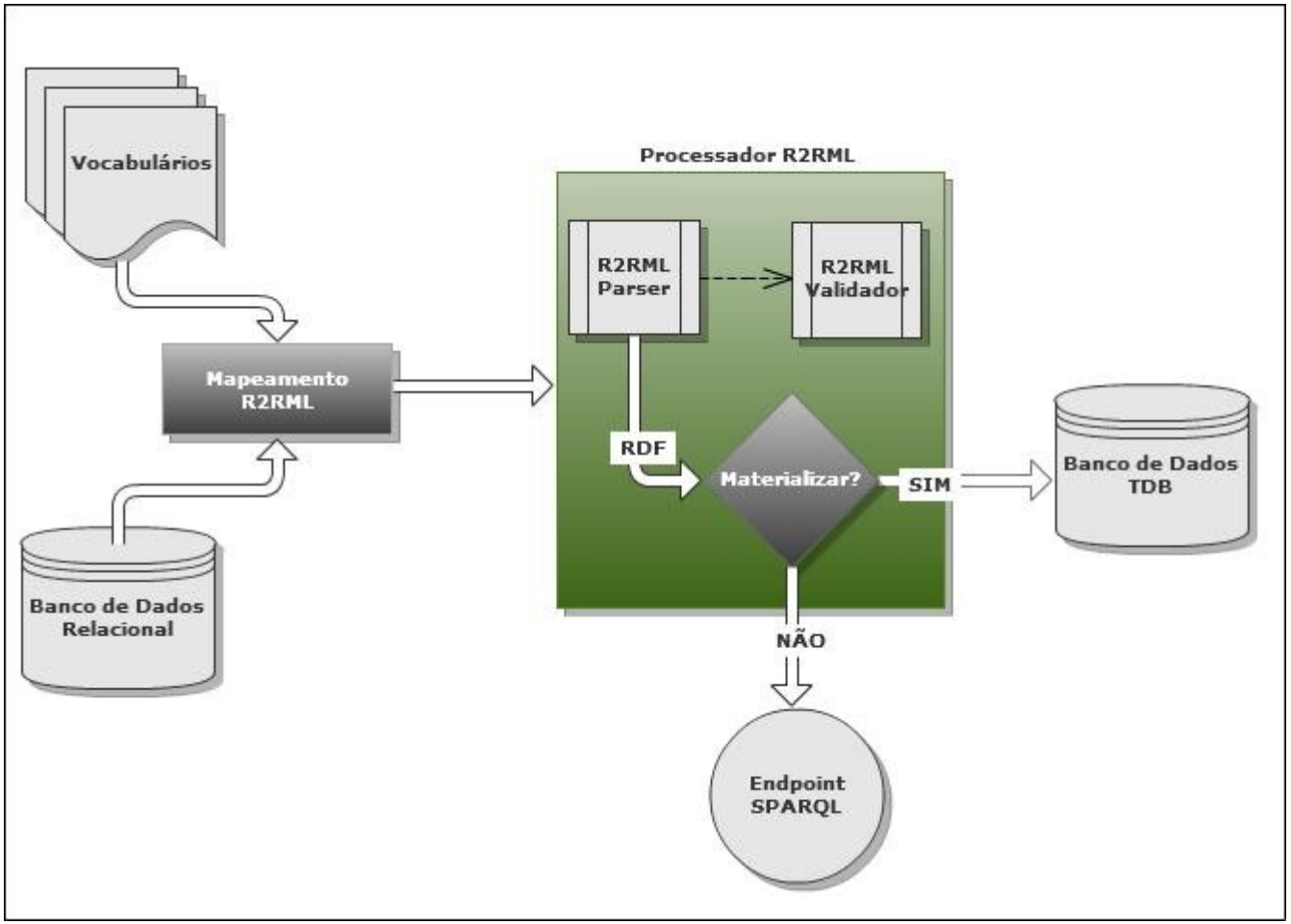
Introdução

- Nesse trabalho apresentamos uma ferramenta para suportar a criação e manutenção de mapeamentos R2RML.
- A ferramenta é baseada na criação de assertivas de correspondência criadas através de uma interface gráfica.
- A interface gráfica induz a utilização de uma metodologia para geração dos mapeamentos.

R2RML

- É uma linguagem para expressar mapeamentos customizados de bases relacionais para datasets RDF.
- Cada mapeamento é criado para um esquema de banco de dados específico e para um vocabulário alvo.

R2RML


Vocabulários
Mapeamento R2RML
Banco de Dados Relacional
Processador R2RML
R2RML Parser
R2RML Validador
RDF
Materializar?
SIM
NÃO
Banco de Dados TDB
Endpoint SPARQL

R2RML

Um mapeamento faz referência a tabelas lógicas que podem ser de 3 tipos:

1. Uma tabela do esquema do banco relacional;
2. Uma visão do esquema do banco relacional;
3. Uma consulta SQL válida chamada de “Visão R2RML”, criada dentro do mapeamento.

R2RML

- Cada tabela lógica é mapeada para RDF pela criação do elemento *TriplesMap*.
- Cada *TriplesMap* é uma regra que mapeia as tuplas da sua tabela lógica para uma ou mais triplas RDF.
- Um *TriplesMap* é composto de um *SubjectMap* e de um ou mais *PredicateObjectMap*.

R2RML

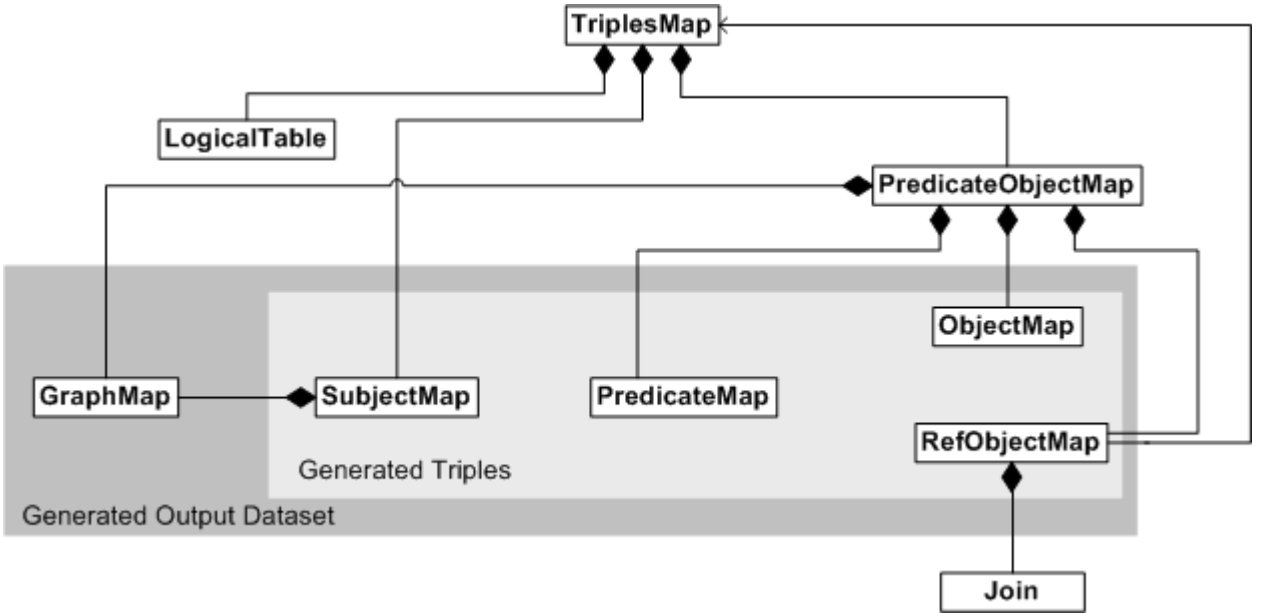
- O *SubjectMap* cria o sujeito das triplas RDF geradas a partir das tuplas da tabela lógica.
- Geralmente é uma *IRI* gerada a partir das colunas que são chave primária da tabela.
- Um *PredicateObjectMap* é composto de um *PredicateMap* e de um *ObjectMap*.

R2RML

- As triplas são produzidas pela combinação de um *SubjectMap*, um *PredicateMap* e de um *ObjectMap*.

```
<#BookTriplesMap>
    rr:logicalTable [ rr:tableName "book" ];
    rr:subjectMap [
        rr:template "book/{title}";
        rr:class sales:Book;
    ];
    rr:predicateObjectMap [
        rr:predicate sales:title;
        rr:objectMap [ rr:column "title" ];
    ];
```

R2RML


TriplesMap
LogicalTable
PredicateObjectMap
ObjectMap
GraphMap
SubjectMap
PredicateMap
RefObjectMap
Generated Triples
Generated Output Dataset
Join

Necessidade da Ferramenta

- Para criar mapeamentos R2RML é preciso ter conhecimentos avançados na linguagem R2RML.
- Para adquirir esse conhecimento e criar manualmente os mapeamentos consumimos bastante tempo.
- Além disso os usuários terão que redefinir os mapeamentos sempre que o esquema do banco relacional muda.

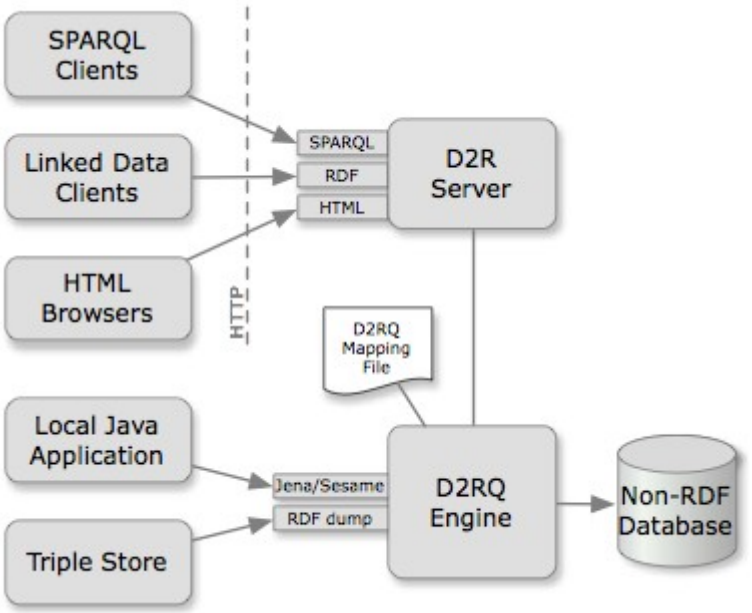
Necessidade da Ferramenta

- É preciso que sejam criadas ferramentas para facilitar a tarefa de criar e manter mapeamentos R2RML.

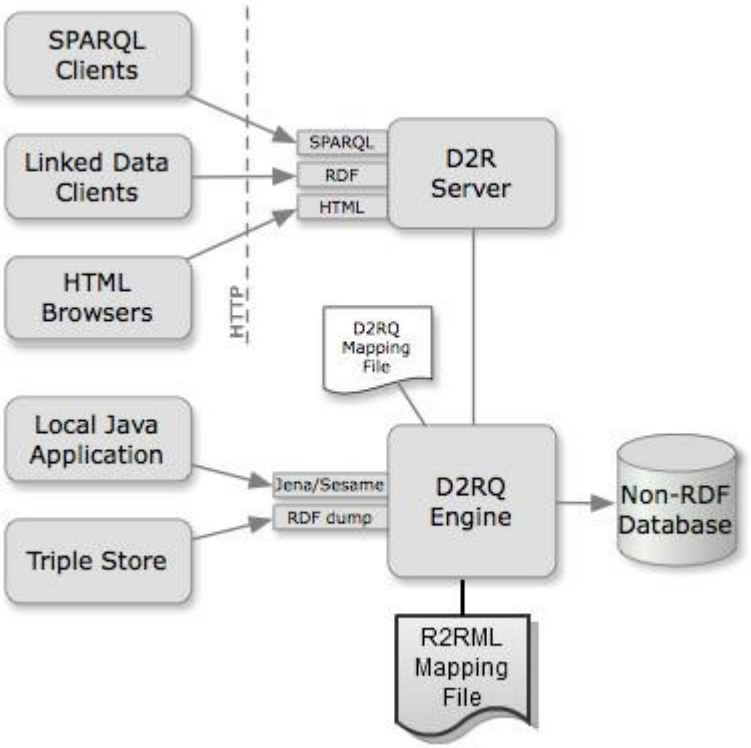
Contribuições

- Como base para a publicação de nossas visões RDF estamos implementando uma mudança evolutiva na ferramenta D2R para suportar mapeamentos R2RML.
- Também implementamos a especificação de geração de mapeamentos diretos de bases relacionais.
- Essas extensões do D2R compõem nossa primeira contribuição.

Contribuições



Contribuições



Contribuições

- Também propomos uma abordagem onde o mapeamento R2RML é derivado de assertivas de correspondência.
- Essas assertivas especificam relacionamentos entre o esquema da ontologia alvo e o esquema do banco de dados relacional.
- Essa é nossa segunda contribuição.

Contribuições

- Nós especificamos formalmente as condições sobre as quais um conjunto de assertivas de correspondência especifica de forma completa uma visão RDF em termos de uma fonte relacional.
- Em seguida mostramos que os mapeamentos definidos pelas assertivas de correspondência podem ser transformados em mapeamentos R2RML.

Contribuições

- Nossa terceira contribuição é um algoritmo que gera os mapeamentos R2RML a partir das assertivas de correspondências criadas.
- A quarta é a ferramenta R2RML-Mapping-By-Assertions que facilita a tarefa de criação e manutenção dos mapeamentos, pois permite que o usuário defina as assertivas de correspondência através de uma interface gráfica.

Visões RDF

- Nesse trabalho adotamos a estratégia de gerar os mapeamentos complexos a partir da criação de visões RDF.
- Essas visões podem ser relacionais criadas na base de dados ou visões R2RML criadas através de consultas SQL dentro do próprio arquivo de mapeamento.

Visões RDF

- As visões relacionais são visões criadas sobre o esquema do banco relacional para expressá-lo em termos do esquema da ontologia alvo.
- A principal vantagem desse tipo de visão é que uma vez que temos todas as visões criadas, podemos criar uma mapeamento direto das visões relacionais para R2RML.
- Dessa forma o mapeamento gerado fica simples, ou seja, todas as tabelas lógicas são visões relacionais.

Visões RDF

- No entanto, as visões relacionais perdem a informação de quais colunas são chaves.
- Os mapeamentos diretos se baseiam na chave primária da tabela para criar a IRI de sua classe correspondente e nas chaves estrangeiras para criar os links entre os indivíduos das classes.

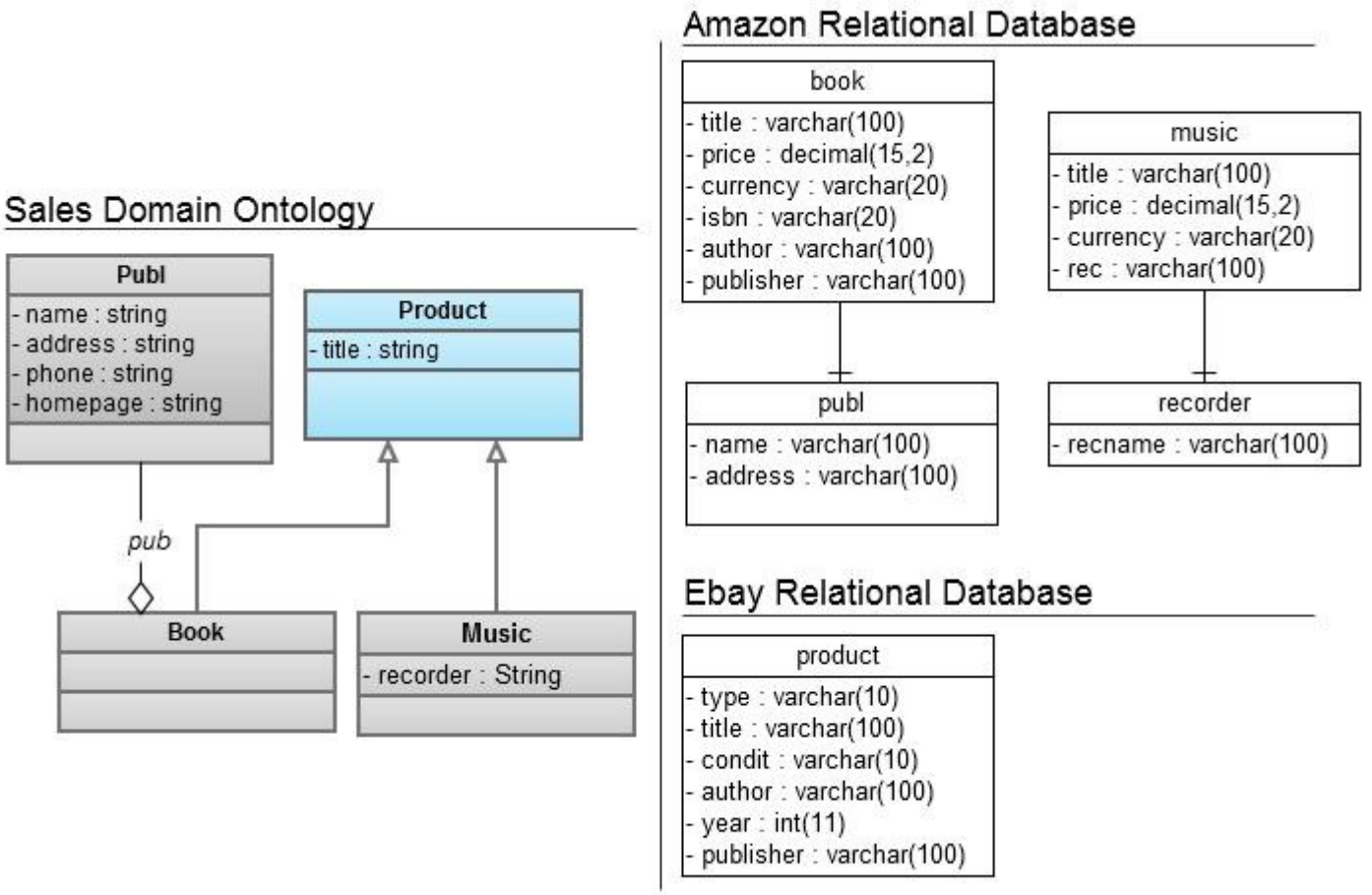
Visões RDF

- Como alternativa podemos colocar as visões dentro do mapeamento ao invés de colocarmos no banco de dados, criando assim visões R2RML.
- A desvantagem é que a geração do arquivo de mapeamento fica mais complexa, pois não podemos usar a abordagem de criação do mapeamento direto.

Visões RDF

- Porém se temos um algoritmo que comprovadamente consegue gerar todos os mapeamentos possíveis, não teremos o problema de ter que ajustar o mapeamento por conta das chaves.
- Entretanto, será preciso alterar a ferramenta D2R para interpretar consultas SQL dentro do mapeamento R2RML. O que seria mais uma potencial contribuição.

Estudo de Caso

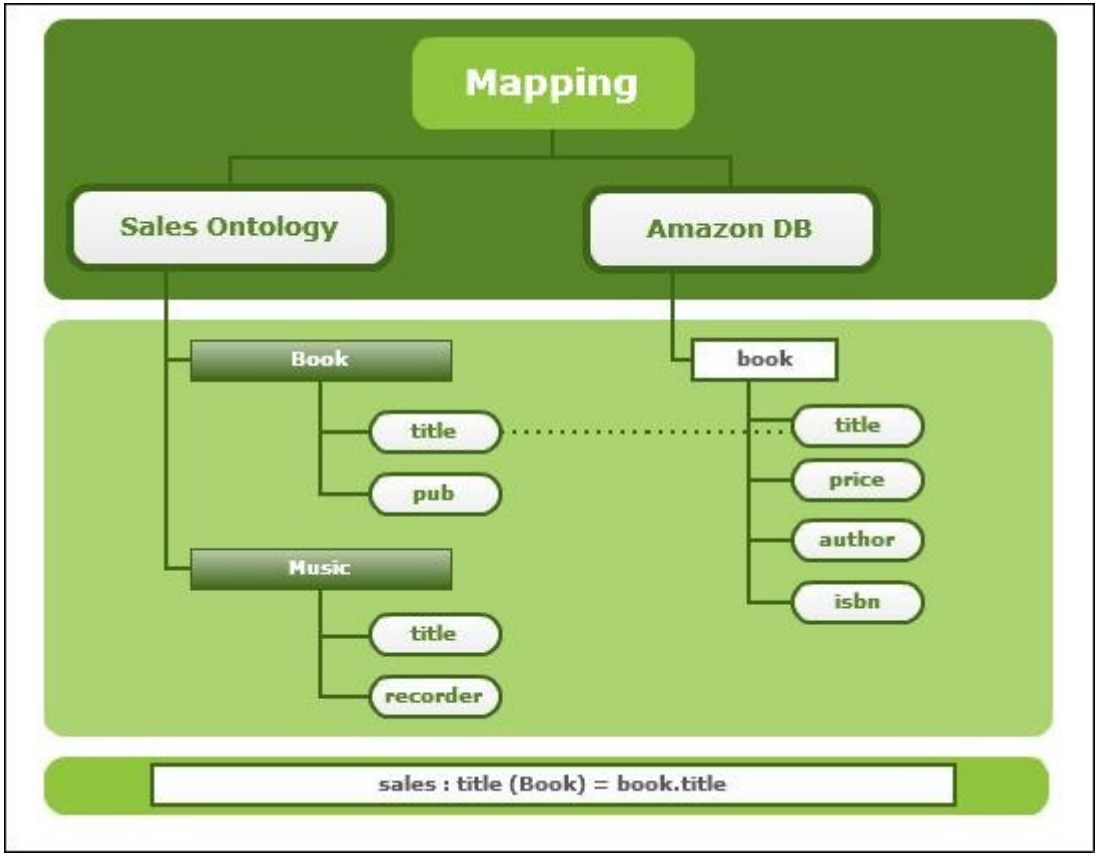


Estudo de Caso

- O primeiro passo para o usuário da ferramenta será escolher a ontologia de domínio e configurar o acesso para a base de dados relacional.
- Digamos que ele escolha a ontologia Sales e a base de dados relacional Amazon.
- Nessa primeira parte vamos utilizar a estratégia de criar visões R2RML para representar os mapeamentos complexos.

Estudo de Caso

- Em seguida o usuário irá utilizar a ferramenta gráfica para criação das assertivas de correspondência:



Estudo de Caso

- Assertivas criadas para mapear tabelas de Amazon em classes de Sales:

```
1. s : Book <= book;
2. s : Music <= music;
3. s : Publ <= publ;
```

- Opcionalmente pode-se definir as colunas que serão usadas para definir a URI. Por Exemplo:

```
s : Book <= book, {iriCols=[title]};
```

- Caso não seja definida nenhuma coluna, a chave primária será utilizada.

Estudo de Caso

- Assertivas simples criadas para mapear colunas de Amazon em propriedades de Sales:

```
4. s : title(Book) <= book.title;
5. s : pub(Book) <= book.publisher;
6. s : title(Music) <= music.title;
7. s : name(Publ) <= publ.name;
8. s : address(Publ) <= publ.address;
```

Estudo de Caso

- Assertiva complexa criada para mapear um caminho em Amazon para uma propriedade de Sales:

```
9.
s : recorder(Music) <=
recorder.recname,
music.rec = recorder.recname;
```

Estudo de Caso

- Geração do mapeamento R2RML a partir das assertivas de correspondência criadas anteriormente.
- O algoritmo deve agrupar as assertivas de propriedades pela classe a qual elas pertencem.
- Se todas as assertivas dentro de um grupo forem simples, então o mapeamento será feito usando uma tabela do esquema como tabela lógica.

Estudo de Caso

- Agrupamento das Assertivas da classe Publ:

```
3. s : Publ <= publ;

7. s : name(Publ) <= publ.name;

8. s : address(Publ) <= publ.address;
```

- Nesse caso, todas as assertivas são simples. Logo o mapeamento R2RML é gerado sem a necessidade de ser criada uma visão R2RML.

Estudo de Caso

```
<#PublTriplesMap>
    rr:logicalTable [ rr:tableName "publ" ];
    rr:subjectMap [
        rr:template "publisher/{name}";
        rr:class s:Publ;
    ];

...
```

Estudo de Caso

```
...
    rr:predicateObjectMap [
        rr:predicate s:name;
        rr:objectMap [ rr:column "name" ];
    ];
    rr:predicateObjectMap [
        rr:predicate s:address;
        rr:objectMap [ rr:column "address" ];
    ].
```

Estudo de Caso

- Agrupamento das Assertivas da classe Book:

```
1. s : Book <= book;

4. s : title(Book) <= book.title;

5. s : pub(Book) <= book.publisher;
```

- Assertiva 5 possui um mapeamento diferente: *s : pub* é uma propriedade que referencia um indivíduo de uma classe e não um literal. Bem como *book.publisher* é uma FK para a tabela *publ*.

Estudo de Caso

```
<#BookTriplesMap>
    rr:logicalTable [ rr:tableName "book" ];
    rr:subjectMap [
        rr:template "book/{title}";
        rr:class s:Book;
    ];
    rr:predicateObjectMap [
        rr:predicate s:title;
        rr:objectMap [ rr:column "title" ];
    ];
...
```

Estudo de Caso

```
...
rr:predicateObjectMap [
    rr:predicate s:pub;
    rr:objectMap [
        rr:parentTriplesMap <#PublTriplesMap>;
        rr:joinCondition [
            rr:child "publisher";
            rr:parent "name";
        ];
    ];
].
```

Estudo de Caso

- Agrupamento das Assertivas da classe Music:

```
2. s : Music <= music;

6. s : title(Music) <= music.title;

9. s : recorder(Music) <=

recorder.recname,

music.rec = recorder.recname;
```

- Como a assertiva 9 não é simples, pois possui um caminho, deverá ser criada uma visão R2RML na tabela lógica do mapeamento da classe Music.

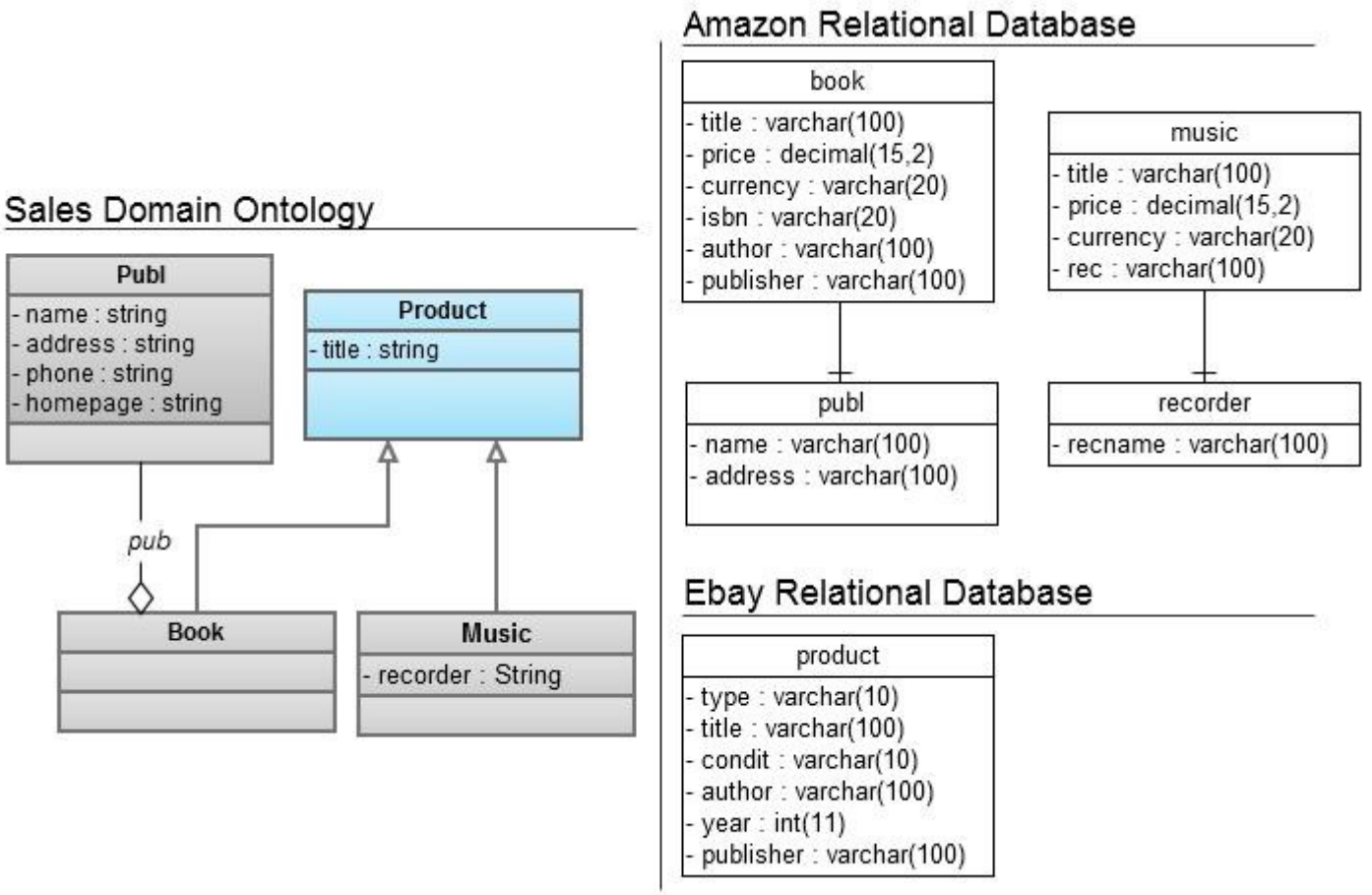
Estudo de Caso

```
<#MusicTriplesMap>
    rr:logicalTable [ rr:sqlQuery """
        SELECT m.title, r.recname
        FROM music m, recorder r
        WHERE m.rec = r.recname
        """];
    rr:subjectMap [
        rr:template "music/{title}";
        rr:class s:Music;
    ];
...
```

Estudo de Caso

```
...
    rr:predicateObjectMap [
        rr:predicate s:title;
        rr:objectMap [ rr:column "title" ];
    ];
    rr:predicateObjectMap [
        rr:predicate s:recorder;
        rr:objectMap [ rr:column "recname" ];
    ].
```

Estudo de Caso



Estudo de Caso

- Na segunda parte vamos escolher a ontologia Sales e a base de dados relacional Ebay.
- Dessa vez vamos utilizar a estratégia de criar visões relacionais para representar os mapeamentos complexos.

Estudo de Caso

- Assertivas criadas para mapear tabelas de Ebay em classes de Sales:

```
1. s : Book <= product, {type='book'};
2. s : Music <= product, {type='music'};
3.
s : Publ <= product,
{type='book', iriCols=[publisher]};
```

Estudo de Caso

- Assertivas criadas para mapear colunas de Ebay em propriedades de Sales:

```
4. s : title(Book) <= product.title;
5. s : title(Music) <= product.title;
6. s : pub(Book) <= product.publisher;
7. s : name(Publ) <= product.publisher;
```

Estudo de Caso

- Novamente agrupamos por Classe para criarmos a visão relacional:

```
1. s : Book <= product, {type='book'};

4. s : title(Book) <= product.title;

6. s : pub(Book) <= product.publisher;
```

Estudo de Caso

- prefixo : classe define o nome da visão: `S_Book`

```
s : Book <= product, {type='book'};
```

- Tabela referenciada na cláusula FROM

```
s : Book <= product, {type='book'};
```

- Restrição incluída na clásula WHERE

```
s : Book <= product, {type='book'};
```

Estudo de Caso

```
CREATE VIEW S_Book AS
SELECT ...
FROM product
WHERE type='book';
```

Estudo de Caso

- prefixo : propriedade define o alias do atributo : `s_title, s_pub`

```
s : title(Book) <= product.title;
s : pub(Book) <= product.publisher;
```

- Colunas incluídas na cláusula SELECT:

```
s : title(Book) <= product.title;
s : pub(Book) <= product.publisher;
```

Estudo de Caso

```
CREATE VIEW S_Book AS
SELECT ...
FROM product
WHERE type='book';
```

Estudo de Caso

```
CREATE VIEW S_Book AS
SELECT title as s_title, publisher as s_pub
FROM product
WHERE type='book';
```

Estudo de Caso

Usando a mesma estratégia, criamos as outras duas visões:

```
CREATE VIEW S_Publ AS
SELECT DISTINCT* publisher as s_name
FROM product
WHERE type='book';
```

* Incluída a cláusula DISTINCT, pois a coluna publisher não possui restrição de chave única.

Estudo de Caso

```
CREATE VIEW S_Music AS
SELECT title as s_title
FROM product
WHERE type='music';
```

Estudo de Caso

```
CREATE VIEW S_Book AS
SELECT title as s_title, publisher as s_pub
FROM product
WHERE type='book';
```

Estudo de Caso

- Estando criadas as visões relacionais, o próximo passo é gerar o mapeamento direto.
- Para realizar essa tarefa utilizamos a implementação da geração de mapeamento direto do D2R seguindo a especificação da w3c.
- Esse mapeamento direto é gerado na linguagem D2RM, pois ainda não foi implementado o mapeamento direto para R2RML.

Estudo de Caso

```
generate_mapping
-d com.mysql.jdbc.Driver
-u root
-p admin
--w3c
-skip-tables offer,product,seller
jdbc:mysql://127.0.0.1/ebay
```

Estudo de Caso

- O detalhe dessa geração do mapeamento direto é que passamos o parâmetro -skip-tables para **não** incluir no mapeamento as tabelas relacionais.
- Dessa forma são criados somente mapeamentos para as visões relacionais, pois esse é o objetivo dessa abordagem.

Estudo de Caso

- Teremos alguns detalhes que precisam ser tratados após a geração do mapeamento.
- O primeiro deles é a geração das URIs.

Estudo de Caso

```
# Table s_book
map:s_book a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    # Sorry, I don't know which columns to put into the uriPattern
    #     for "s_book" because the table doesn't have a primary key.
    #     Please specify it manually.
    d2rq:uriPattern "s_book";
    d2rq:class <s_book>;
    .
```

Estudo de Caso

- Esse problema ocorre pois a visão não possui chave primária.
- Para resolvê-lo temos que alterar o valor da propriedade uriPattern após a geração do mapeamento.
- Nela podemos incluir a chave primária da tabela original ou as colunas referenciadas na assertiva de correspondência criada pelo usuário.

Estudo de Caso

```
# Table s_book
map:s_book a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "product/title-
@@product.title|encode@@";
    d2rq:uriPattern "s_book";
    d2rq:class <s_book>;
    .
```

Estudo de Caso

- Outra situação que precisa ser tratada é a geração das propriedades de objeto.
- Como a visão relacional não possui informação de chave estrangeira, as propriedades de objeto são criadas como propriedades de dados.
- Assim a propriedade que deveria referenciar um indivíduo de outra classe é mapeado para referenciar um literal.

Estudo de Caso

- No exemplo que estamos tratando a situação ainda é mais complexa pois nem a tabela relacional original possui chave estrangeira.

Estudo de Caso

```
map:s_book_s_pub a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:s_book;
    d2rq:property <s_book#s_pub>;
    d2rq:column "s_book.s_pub";
    .
```

Estudo de Caso

- O algoritmo que ajusta o mapeamento precisa ser inteligente para perceber que ali se trata de uma propriedade de objeto que precisa ser mapeado em um indivíduo da classe Publ.
- E então fazer a devida substituição da propriedade `d2rq:column` para a propriedade `d2rq:uriPattern`.

Estudo de Caso

```
map:s_book_s_pub a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:s_book;
    d2rq:property <s_book#s_pub>;
    d2rq:uriPattern
"publisher/@@product.publisher|urlify@@";
    .
```

Trabalhos Futuros

- Formalizar as assertivas de correspondência entre esquemas relacionais e ontologias.
- Decidir qual a melhor abordagem a ser seguida: visões R2RML ou visões relacionais.
- Concluir a implementação do parser R2RML.
- Implementar o mapeamento direto em R2RML.
- Criar a ferramenta gráfica para geração das assertivas de correspondência.

Trabalhos Futuros

- Especificar os algoritmos que transformam as assertivas em visões RDF.
- Implementar esses algoritmos na ferramenta.
- Integrar a ferramenta com D2R para publicação final dos dados relacionais em visões RDF.

Referências

1. Resource Description Framework (RDF) Model and Syntax Specification, W3C Proposed Recommendation 05 January 1999. Disponível em http://www.w3.org/TR/PR-rdf-syntax/. Acessado em 2012.

2. Tim Berners-Lee: Linked Data, Design Issues. Disponível em:

<http://www.w3.org/DesignIssues/LinkedData.html >. Acessado em 2012.

3. A Direct Mapping of Relational Data to RDF, Marcelo Arenas, Eric Prud'hommeaux, Juan Sequeda, Editors. World Wide Web Consortium, 24 de Março de 2011. Disponível em http://www.w3.org/TR/rdb-direct-mapping/.

Referências

4. Percy E. Salas, Karin K. Breitman, Marco A. Casanova, José Viterb: StdTrip: Promoting the Reuse of Standard Vocabularies in Open Government Data.

5. Sacramento, E., Vidal, V. M., Macêdo, J. A., Lóscio, B. F., Lopes, F. L. R., Casanova, M. A., and Lemos, F.: Towards automatic generation of application ontologies. In: Journal of Information and Data Management, Vol. V, No. N, Month 20YY, pp.1-16, 2010.

6. David Beckett, Tim Berners-Lee: Turtle - Terse RDF Triple Language. Disponível em http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/ .

7. M. Dürst and M. Suignard: Internationalized Resource Identifiers (IRIs), Internet-Draft, June 2003, expires December 2003. This document is http://www.w3.org/International/iri-edit/draft-duerst-iri-04.

Referências

8. Hull, R. and Yoshikawa, M. Ilog: declarative creation and manipulation of object identifiers. In Proceedings of the 16th International Conference on Very Large Databases (VLDB). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 455#468, 1990.

9. Leme, L. A. P. P., Casanova, M. A., Breitman, K. K., and Furtado, A. L. Instance-based OWL schema matching. In Proceedings of the 11th International Conference on Enterprise Information Systems. Lecture Notes in Business Information Processing, vol. 24. pp. 14-26, 2009.

10. Das Souripriya, Sundara Seema, Cyganiak Richard (2011), R2RML: RDB to RDF Mapping Language. Disponível em < http://www.w3.org/TR/r2rml/>

Referências

11. Bizer, C., Cyganiak, R.: D2R server – publishing relational databases on the Semantic Web. Disponível em:

http://www4.wiwiss.fu-berlin.de/bizer/pub/Bizer-Cyganiak-D2R-Server-ISWC2

12. Jena - A Semantic Web Framework for Java. Disponível em:

<http://jena.sourceforge.net>

Trabalhos Futuros

- Formalizar as assertivas de correspondência entre esquemas relacionais e ontologias.
- Decidir qual a melhor abordagem a ser seguida: visões R2RML ou visões relacionais.
- Concluir a implementação do parser R2RML.
- Implementar o mapeamento direto em R2RML.
- Criar a ferramenta gráfica para geração das assertivas de correspondência.