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

Luís Eufrasio Teixeira Neto
UFC – Universidade Federal do Ceará,
Grupo ARIDA, Fortaleza, Brasil

Resumo

O p L D f e c a m p p e c
e c p d d i e c n W S u
U e R N e a g m d d c i d d
W p a e SG re P t a i
a e b re d p a W d d é p
p l n m R E p p s m (d
d ou V o q é f p f c o D2 S a d c
d e SPA U e f é p f u si
e d d re p R u u m d o ca
t é m e u C e ca c é m e u a d
c P e q e t c d d n a a
r d w e s p n m d c d s p o d
u u v c q p m r e a i d b c
o f R D f é p c u v R u v
d u o d d b c c u m d o l
p e o d d U v d a o e c o
m e e é p t e m p u f e q
p s i p u f d p d d R A
u o D2 p p v d d re a d c d
m m e q d t p s c e e
e m N t t d p d ge a t
a d m n P t p u f w e q e
d p o u a c d c d a à b (s
mú a ge d o l a c d o d a u
u v q é s d v d o d d a c
d m d a n e a o l e d a e a ge e
a d a d m d D2 a p d r d m d
a n C e a o u p p d re e
R p s a e d a o e (a e o

m d a n p e e

1 Introdução

P p b re n W S já ex a f d c S O V e D2 S o e ú f e p s u n t D n W S s m e r e R O D2 S u o “D2 m p m o c d b d d p e f a p q o d R p s n e p A i L D d D2 t a d R d r d a d p HTTP A u d R p s r si a a U d r n W P t p s u n d w e s c o T [http://www.w3.org/2005/ajar/tab] ou o D [http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/] o o u p s l d u r p o o n n W d d Em o D2 s u p f e exi u c c n c d m a c a o f c U d o d t é c u w p o D2R o o u p p s d e R s p c a l d m e d D2 S n p c l e O r d a e o d s f a s 2 c o p d ge m c d b re p u o d a A s 3 d a d u ex a l d m d D2 S A s 4 a e d c p o p A s 5 c o a a p a f d p

2 Contextualização do Problema

P p f d d n W d a c o p d L D é n s u s d p c d n F p p p d d l

1. P c u *source ontology* $S = (V_S, C_S)$, q m o d a s p
2. E s u *domain ontology* $D = (V_D, C_D)$, q m o d d a D f D p s u c d o c d d
3. D p c u m *source-to-domain* γ d s p D (no q γ p n c d p t o sím e V_D) e

ge u *exported (application) ontology*)

4. P e R d *exported ontology*. O d p s e d d f E v o é c u SPA e p ou m q e e m o d e R

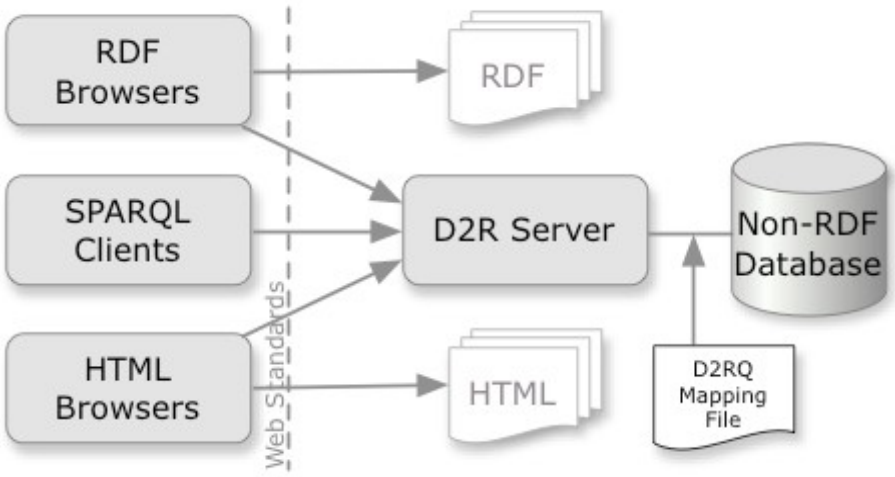
N t s a o p re a ú p u o e v a p D2 p p o d N e é n e f d t o d m γ p a l d m D2RM s p D2 S A d d t ees i s a p c d t

3 Mapeamento D2RQ

3.1 S o D2 S

C já f d n i o D2 S é u f p p o c d b d d re n W S u e d i g c d d l

El p s a d 3 f d c m a f 3.1:



Fi 3.1

o s **interface Linked Data** t a d R d r i d n p HTTP a **interface SPARQL** t p q a p e c o b d d u a l d c SPA s o p SPA e u i HTML q o a a n W f

Re o d W s r e c SQL u o m E t " t fly" p a p d g b e d

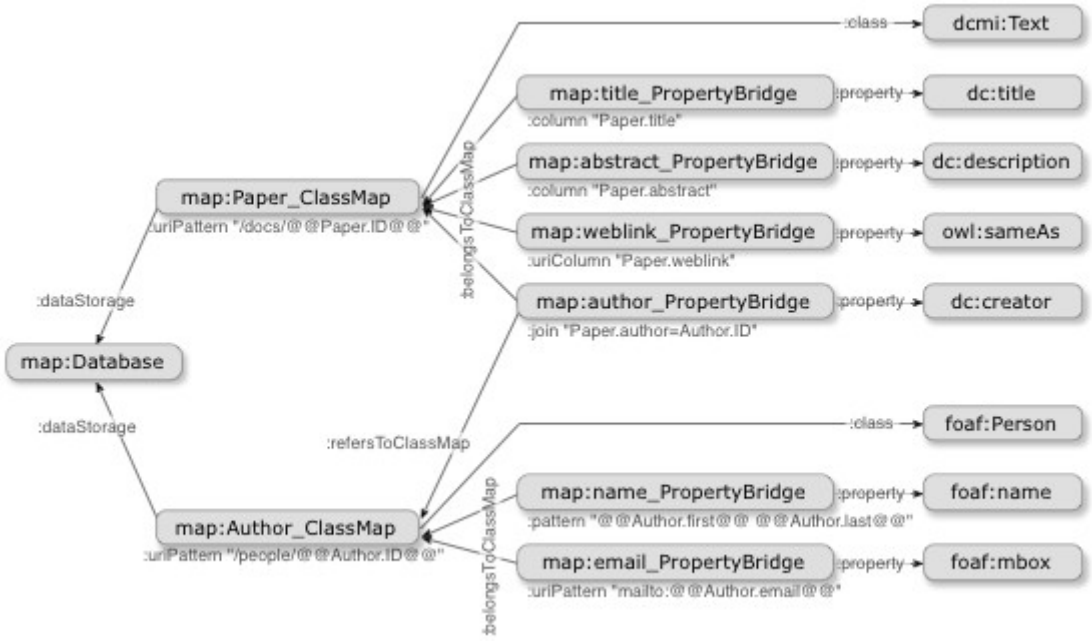
d n f R e e a n d r d d

3.2 L d E d M

A l D2 m é u l de p d a re e u b re e v R ou o OWL U m D2 é u d R

U o é m p u e d b d d u `d2rq:ClassMaps` e `d2rq:PropertyBridges`. O o c é o C q r u c ou u gru d c si d o U C i c i d c s i El t u c d Pro q e c a p d u i s c

A i a m a e d u ex d u m D2



Fi 3.2

N ex a t a d d B d D c s c d a a C P c s a u vá v já ex a C A c s a e u re e P e A

3.2.1. Ex d m d t *conferences* p a c *conference* n o

```
#D2RQ Namespace
@prefix d2rq:        <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
# Namespace of the ontology
@prefix : <http://annotation.semanticweb.org/iswc/iswc.daml#> .

# Namespace of the mapping file; does not appear in mapped data
@prefix map: <file:///Users/d2r/example.n3#> .
```

```
# Other namespaces
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

map:Database1 a d2rq:Database;
    d2rq:jdbcDSN "jdbc:mysql://localhost/iswc";
    d2rq:jdbcDriver "com.mysql.jdbc.Driver";
    d2rq:username "user";
    d2rq:password "password";
    .
# ------------------------------------------------
# CREATE TABLE Conferences (ConfID int, Name text, Location text);

map:Conference a d2rq:ClassMap;
    d2rq:dataStorage map:Database1.
    d2rq:class :Conference;
    d2rq:uriPattern "http://conferences.org/comp/confno@@Conferences.ConfID@@";
    .
map:eventTitle a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Conference;
    d2rq:property :eventTitle;
    d2rq:column "Conferences.Name";
    d2rq:datatype xsd:string;
    .
map:location a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Conference;
    d2rq:property :location;
    d2rq:column "Conferences.Location";
    d2rq:datatype xsd:string;
    .
```

O e u n c d m s d n p s

3.2.2. D

U **d2rq:Database** d u c JDBC ou ODBC p u b re l e e o ti d c d b d d U m D2 p c vá d2rq:D p a d b l

Ex

```
map:Database1 a d2rq:Database;
        d2rq:jdbcDSN "jdbc:mysql://localhost/iswc";
        d2rq:jdbcDriver "com.mysql.jdbc.Driver";
        d2rq:username "user";
        d2rq:password "password";
        d2rq:numericColumn "Conferences.ConfID";
        d2rq:textColumn "Conferences.URI";
        d2rq:textColumn "Conferences.Name";
        d2rq:textColumn "Conferences.Location";
        d2rq:dateColumn "Conferences.Date".
```

3.2.3. C

U **d2rq:ClassMap** r u c ou u gru d c si d u o OWL ou e R U m d c d c i d c s i El e c a u d2rq:D e t u c d d2rq:Pro q a p p a i

3.2.4. I d R

O D2 p q m d p a i a i n b d d

URI patterns

U U p é i p i d v d c c n b d d d d u p Ex

http://example.org/persons/@@*Persons.ID*@@
http://example.org/lineItems/item@@*Orders.orderID*@@-@@*LineItems.itemID*@@
urn:isbn:@@*Books.isbn*@@
mailto:@@*Persons.email*@@

A p e o `@@'` m c n b n no `Table.Column`. U p s u c a p `d2rq:uriPattern`.

C ca c e e b ou o sím h n s p e U ou t u sig e C q c m t ca p s c a q s v p s i e u U p

S u c é d c `@@Table.Column|urlencode@@`, e a URL é c a d st s i

S u c é d c `@@Table.Column|urlify@@`, e a c d URL é a c u r a o e s c e u (_)

Relative URI patterns

U re U p é u U p q ge U re `persons/@@Persons.ID@@`

El s c c u U b d a d p p f U c R U p p a c d m p q p s u p mú i d m e d b d d R U p s ge c a p `d2rq:uriPattern`.

URI columns

E a c o b d d p já c U q p s u c i d r t c u URL d pá w e e d d U d ti s ge d c d b c a p

d2rq:uriColumn.

Blank nodes

R t t o c d b n q ex q d a r q ex e t c p m n s n N D2 b n p s ge d u ou m c U b n d s ge p ca c d d v d c A c s e u a p d2rq:bNodeIdColumns.

Exemplo: ClassMap onde instâncias são identificadas usando um URI pattern

```
map:PaperClassMap a d2rq:ClassMap;
d2rq:uriPattern
"http://www.conference.org/conf02004/paper#Paper@@Papers.PaperID@@";
    d2rq:class :Paper;
    d2rq:classDefinitionLabel "paper"@en;
    d2rq:classDefinitionComment "A conference paper."@en;
    d2rq:dataStorage map:Database1.
```

Exemplo: ClassMap onde instâncias são identificadas usando blank nodes

```
map:Topic a d2rq:ClassMap ;
    d2rq:bNodeIdColumns "Topics.TopicID" ;
    d2rq:class :Topic ;
    d2rq:classDefinitionLabel "topic"@en;
    d2rq:classDefinitionComment "A topic."@en;
    d2rq:dataStorage map:Database1 .
```

3.2.5. Pro B

R c d t d b c p R S u p a p a r R c e s C O v d p p s l m t p s U ou b n q re o r c o r P ex o v d p :author d c Paper p s u U r u p

S a c u n p b t v NULL n b p a l d t e n éc a p p o r q c a e l

Exemplo: Uma property bridge simples

```
map:PaperTitle a d2rq:PropertyBridge;
```

```
    d2rq:belongsToClassMap map:Paper;
    d2rq:property :title;
    d2rq:column "Papers.Title";
    d2rq:lang "en";
    d2rq:propertyDefinitionLabel "title"@en;
    d2rq:propertyDefinitionComment "A paper's title."@en;
    .
```

C i a p a:title f a a t o r ge p c map:Paper. O v d p v d c thePapers.Title. O l ge t u t i a lín "en".

Exemplo: Property bridge usando informação de outra tabela do banco

```
map:authorName a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Papers;
    d2rq:property :authorName;
    d2rq:column "Persons.Name";
    d2rq:join "Papers.PaperID <= Rel_Person_Paper.PaperID";
    d2rq:join "Rel_Person_Paper.PersonID => Persons.PerID";
    d2rq:datatype xsd:string;
    d2rq:propertyDefinitionLabel "name"@en;
    d2rq:propertyDefinitionComment "Name of an author."@en;
    .
```

E p b a o n d a a pa S u pa t vá a e vá p :authorName s a A t Papers e Persons t u re n:m. O d2rq:join é u p f o j d t c Rel_Person_Paper. A c d j c s d q i a cha e d re e s u c u h d o N ex a a d d s i q t o v p d Rel_Person_Paper.PaperID e Rel_Person_Paper.PersonID e p e Papers.PaperID e Persons.PerID, r Q i n e c u si s d i (=) p s u

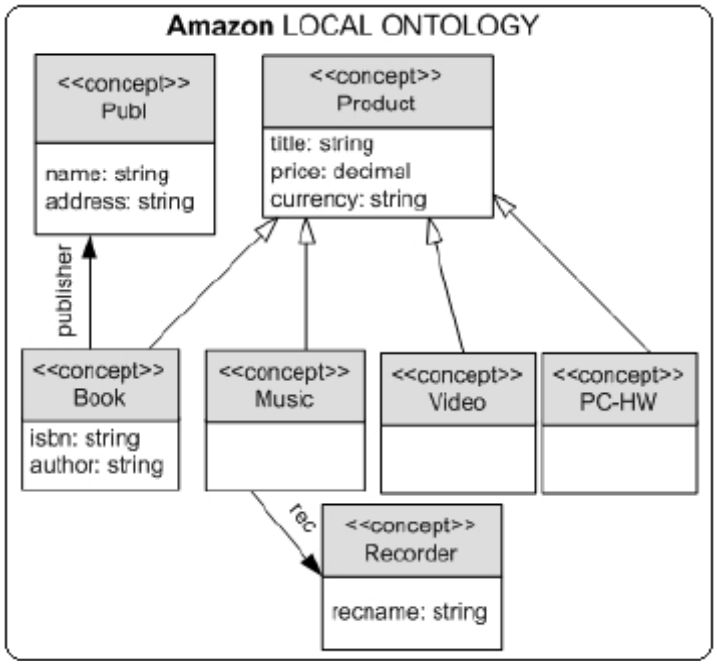
Exemplo: Junção de uma tabela com ela mesma usando `d2rq:alias`

```
map:ParentTopic a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Topic;
    d2rq:property :parentTopic;
    d2rq:refersToClassMap map:Topic;
    d2rq:join "Topics.ParentID => ParentTopics.ID";
    d2rq:alias "Topics AS ParentTopics";
    .
```

Aqui, um topic pode ter um parent topic cujo ID pertence à coluna Topics.ParentID. Essa chave estrangeira faz referência à coluna Topics.ID. É preciso ser feito um join da tabela com ela mesma. Um d2rq:alias é declarado, e o join é estabelecido entre a tabela original e a tabela com o alias. Esse pattern é típico para relacionamentos hierárquicos.

4 Estudo de Caso

E s a o ex u e [2] q c d u a a l
v m p c q c l n W O ex i d
o l u o d d e d o d a ge
a c e e [2] N t i f n
m e a o l e d a N p p d e
ex m a o l d A o s d c d b d
d o m c e s e n m p o D2R. e



Fi 4.1

CREATE TABLE P (n VARCHAR(100) NOT NULL, a VARCHAR(200) NOT NULL, p key( );	CREATE TABLE Book (t VARCHAR(100) NOT NULL, p DECIMAL(15,2) NOT NULL, cu VARCHAR(20) NOT NULL i VARCHAR(20) NULL, a VARCHAR(100) NULL, p VARCHAR(100) NULL, P KEY(FOREIGN KEY(p REFERENCES P);
CREATE TABLE R (r VARCHAR(100) NOT NULL, p key( );	CREATE TABLE M (t VARCHAR(100) NOT NULL, p DECIMAL(15,2) NOT NULL, cu VARCHAR(20) NOT NULL r VARCHAR(100) NOT NULL, P KEY(FOREIGN KEY(REFERENCES R);
CREATE TABLE V (t VARCHAR(100) NOT NULL, p DECIMAL(15,2) NOT NULL, cu VARCHAR(20) NOT NULL, P KEY( );	CREATE TABLE PC_HW (t VARCHAR(100) NOT NULL, p DECIMAL(15,2) NOT NULL, cu VARCHAR(20) NOT NULL, P KEY( );

T 4.1

A q t Pr é d u ti e u a e 2 d m d Hi d c d e [R 2008] N e a s c s a re d ti e c o a d c r n

E s a a o d d e d a o e ú s n o a ou s s c u m d o l

para ela. Essa ontologia de aplicação foi gerada utilizando o matching de vocabulários mostrado na tabela 4.2 e as regras geradas no post-matching. A criação dessas regras está definida em [2].

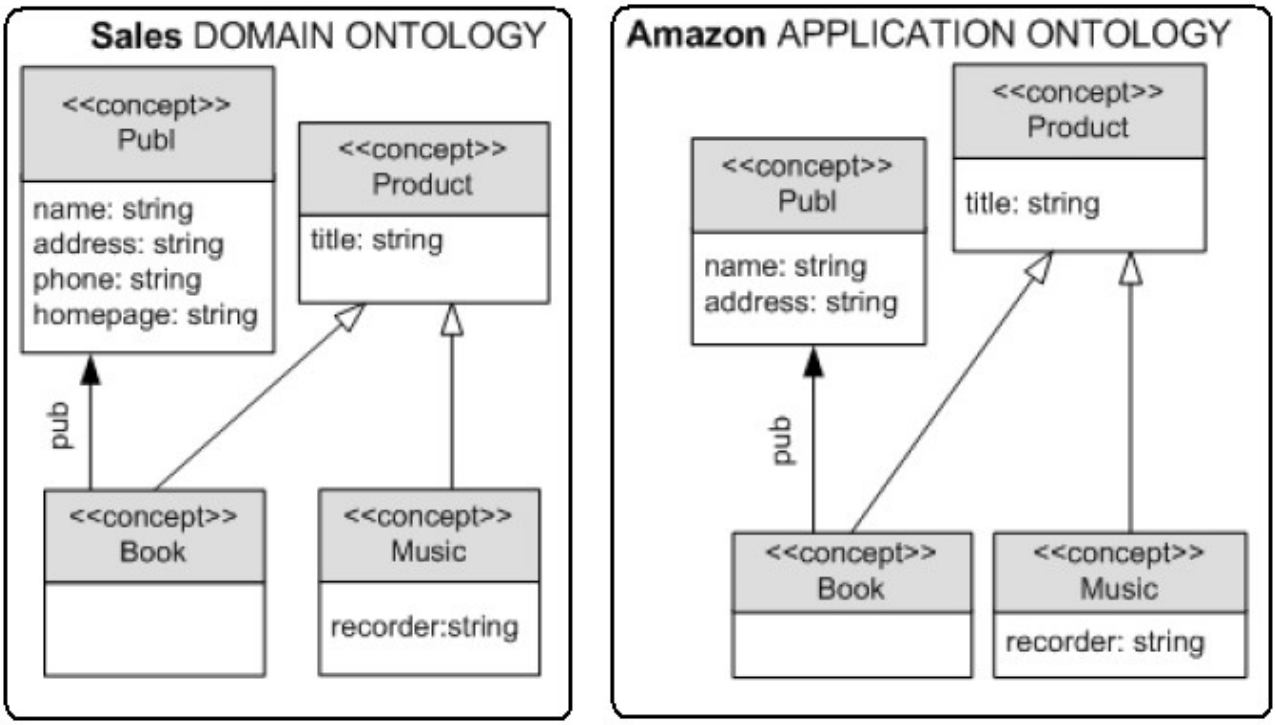


Figura 4.2

#	O L - ***Amazon***			O d D - ***Sales***		
	v1	e1	r1	v2	e2	r2
1	a: Book	T	T	s: Book	T	T
2	a: M	T	T	s: M	T	T
3	a: P	T	T	s: P	T	T
4	a: p	a: Book	T	s: p	s: Book	T
5	a: t	a: Book	T	s: t	s: Book	T
6	a: t	a: M	T	s: t	s: M	T
7	a: n	a: P	T	s: n	s: P	T
8	a: a	a: P	T	s: a	s: P	T
9	a: r	a: R	T	s: r	s: M	T

T 4.2

A p t a r d m p a s u p D2
S e d f p o d re e u V R

#1 s : Book(b) ← a : Book(b)

```
map:book a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "book/@@book.title|urlify@@";
    d2rq:class a:Book;
    d2rq:classDefinitionLabel "book";
    .
```

M d c *Book*, o s d ob a q e p o p d U p a au i d c ac d n o a e u l .

#2 s : Music(m) ← a : Music(m)

```
map:music a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "music/@@music.title|urlify@@";
    d2rq:class a:Music;
    d2rq:classDefinitionLabel "music";
    .
```

M d c *Music*, o s d ob a q e p o p d U p a au i d c u c cha o c t ac d n o a e u l .

#3 s : Publ(p) ← a : Publ(p)

```
map:publisher a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "publisher/@@publisher.name|urlify@@";
    d2rq:class a:Publ;
    d2rq:classDefinitionLabel "publisher";
    .
```

M d c *Publ*, o s d ob a q e p o p d U p a au i d c u c cha o c *name*, ac d n o a e u l .

#4 s : pub(b, p) ← a : publisher(b, p), a : Book(b)

```
map:book_pub a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:book;
    d2rq:property a:Publ;
    d2rq:refersToClassMap map:publisher;
```

```
    d2rq:join "book.pub => publisher.name";
    .
```

M d c *publisher* , o é d o re c a C *Publ* c a c d jun i a t re

#5 s : title(b, t) ← a : title(b, t), a : Book(b)

```
map:book_title a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:book;
    d2rq:property rdfs:label;
    d2rq:column "book.title";
    .
```

M d c *title*, o é d a hi p o m d C *Book*, o c é d c u *label* p u i *book* e a c o é *title* d t *book*.

#6 s : title(m, t) ← a : title(m, t), a : Music(m)

```
map:music_title a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:music;
    d2rq:property rdfs:label;
    d2rq:column "music.title";
    .
```

M d c *title*, o é d a hi p o m d C *Music*, o c é d c u *label* p u i *music* e a c o é *title* d t *music*.

#7 s : name(p, n) ← a : name(p, n), a : Publ(p)

```
map:publisher_name a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:publisher;
    d2rq:property a:name;
    d2rq:propertyDefinitionLabel "publisher->name";
    d2rq:column "publisher.name";
    .
```

M d c *name*, o é d a hi p o m d C *Publisher*, o c é d c u *label* p u i *publisher* e a c o é *name* d t *publisher*.

#8 s : address(p, a) ← a : address(p, a), a : Publ(p)

```
map:publisher_address a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:publisher;
    d2rq:property a:address;
    d2rq:propertyDefinitionLabel "publisher->address";
    d2rq:column "publisher.address";
    .
```

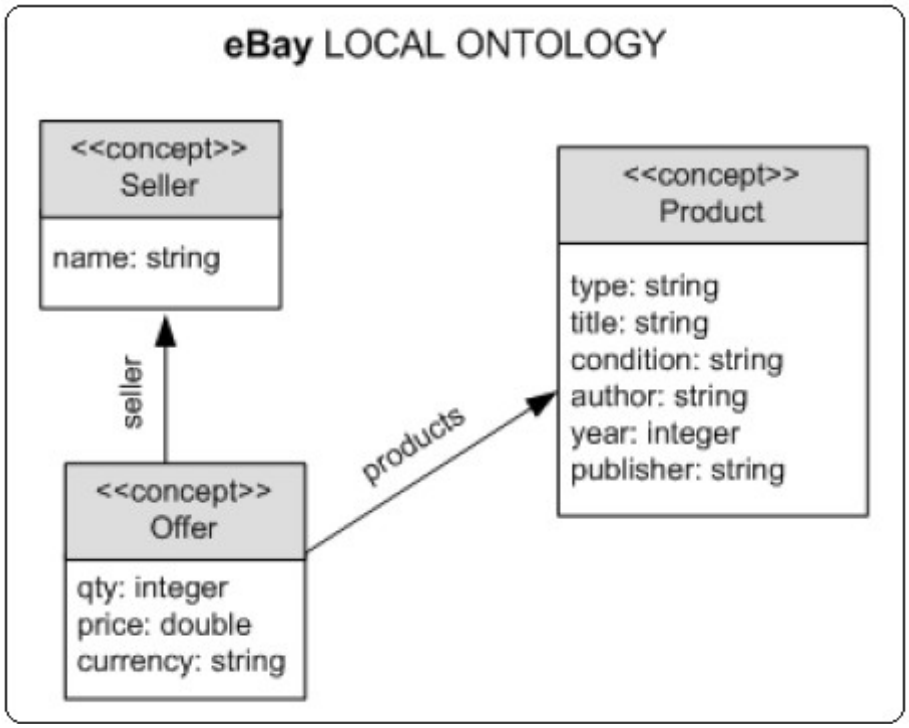
M d c *address*, o é d a hi p o m d C *Publisher*, o c é d c u *label* p u i *publisher* e a c o é *address* d t *publisher*.

#9 s : recorder(m, n) ← a : rec(m, r), a : recname(r, n), a : Recorder(r)

```
map:music_rec a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:music;
    d2rq:property a:recorder;
    d2rq:column "recorder.recname";
    d2rq:join "music.rec => recorder.recname";
    .
```

M d c *recorder*, o é d a hi p o m d C *Music*. E é u m d d a p o c d o n e n t m l é d u c a d j p a t R

N s p d e d c e a o l (4.3) a b d d c e o m p o ex d eBay N c t d u o m si p c u p d m d d a q m m a



Fi 4.3

CREATE TABLE S (n VARCHAR(100) NOT NULL, p key( );	CREATE TABLE Pr (ty VARCHAR(10) NOT NULL, t VARCHAR(100) NOT NULL, c VARCHAR(10), a VARCHAR(100), y INT, p VARCHAR(100), P KEY( );

```
CREATE TABLE Of (
   i   INT NOT NULL,
   qty INT NOT NULL,
   p       DECIMAL(15,2) NOT NULL,
   cu          VARCHAR(20) NOT NULL,
   s       VARCHAR(100) NOT NULL,
   p          VARCHAR(100) NOT NULL,
   FOREIGN KEY(
      REFERENCES S
   FOREIGN KEY(
      REFERENCES Pr
   p          key(
```

);

Tabela 4.3

Por ser ainda mais simples, o mapeamento das classes para tabelas foi direto, onde cada classe foi mapeada em uma tabela no banco de dados. A tabela Offer possui os campos de relacionamento com Seller e Product. No entanto, quando consideramos a ontologia a ser exportada, somente a tabela de produtos será mapeada. As outras duas não pertencem ao vocabulário comum. Logo abaixo apresentamos as ontologias de domínio e de aplicação, onde esta última será nossa ontologia alvo, ou seja, será criado um mapeamento da ontologia local para ela. Essa ontologia de aplicação foi gerada utilizando o matching de vocabulários mostrado na tabela 4.4 e as regras geradas no post-matching. A criação dessas regras também estão definidas em [2].

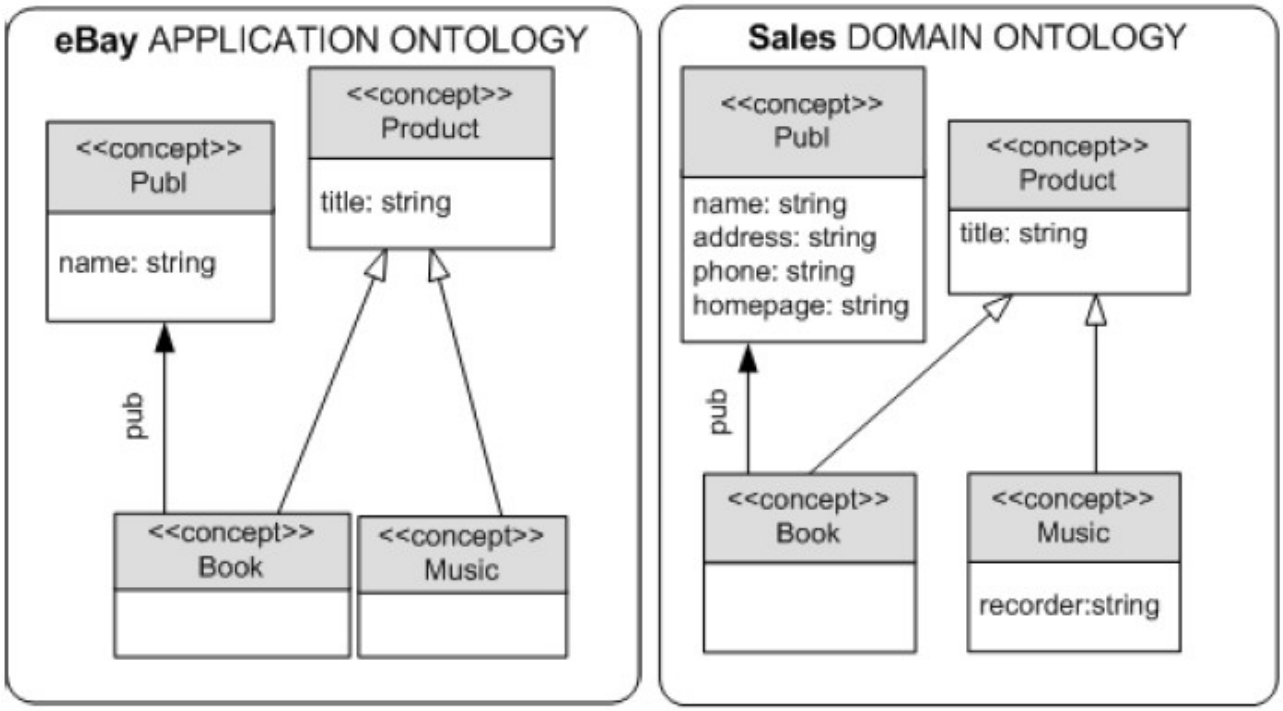


Figura 4.4

#	Ontologia Local - ***eBay***			O d D - ***Sales***		
	v1	e1	r1	v2	e2	r2
1	e: Pr	T	type='book'	s: Book	T	T
2	e: Pr	T	type='music'	s: M	T	T
3	e: t	e: Pr	type='book'	s: t	s: Book	T
4	e: t	e: Product	type='music'	s: t	s: M	T

5	e: p	e: Product	type='book'	s: n	s: P	T

T 4.4

m D m si a q f n o d A p t
a r d m p a s u p D2 S e a p e
o d re d b d d eBay e u V R

#1 s : Book(p) ← e : Product(p), e : type(p) = 'book'

```
map:book a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "book/@@product.title|urlify@@";
    d2rq:class e:Book;
    d2rq:classDefinitionLabel "book";
    d2rq:condition "product.type='book'";
    .
```

M d c *Book*, o s d o b a q e p o
p d U p a a u i d c a c d n o
a u l e o d m i q é a c d u c d
q o ti d p é '*book*'.

#2 s : Music(p) ← e : Product(p), e : type(p) = 'music'

```
map:music a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "music/@@product.title|urlify@@";
    d2rq:class e:Music;
    d2rq:classDefinitionLabel "music";
    d2rq:condition "product.type='music'";
    .
```

M d c *Music*, o s d o b a q e p o
p d U p a a u i d c a c d n o
a u l e o d m i q é a c d u c d
q o ti d p é '*music*'.

#3 s : title(p, t) ← e : title(p, t), e : Product(p), e : type(p) = 'book'

```
map:book_title a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:book;
    d2rq:property rdfs:label;
    d2rq:column "product.title";
```

.

M d c *title*, o é d a hi p o m d C *Book*, o c é d c u *label* p u i *book* e a c o é *title* d t *product*.

#4 s : title(p, t) ← e : title(p, t), e : Product(p), e : type(p) = 'music'

```
map:music_title a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:music;
    d2rq:property rdfs:label;
    d2rq:column "product.title";
    .
```

M d c *title*, o é d a hi p o m d C *Music*, o c é d c u *label* p u i *music* e a c o é *title* d t *product*.

#5.1 s : Publ(fpubl(n)) ← e : publisher(b, n), e : Product(b), e : type(b) = 'book'

```
map:publisher a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:uriPattern "publisher/@@product.publisher|urlify@@";
    d2rq:class e:Publ;
    d2rq:classDefinitionLabel "publisher";
    .
```

N p e s o p d m c a En n o l o *Publisher* é si u c d t *Product*, n o e e s u C L e m é c c e i

#5.2 s : name(fpubl(n), n) ← e : publisher(b, n), e : Product(b), e : type(b) = 'book'

```
map:publisher_name a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:publisher;
    d2rq:property e:name;
    d2rq:propertyDefinitionLabel "publisher->name";
    d2rq:column "product.publisher";
    .
```

Alé d q f d n i a a c P p u a n q t p s m

#5.3 s : pub(b, fpubl(n)) ← e : publisher(b, n), e : Product(b), e : type(b) = 'book'

```
map:book_pub a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:book;
    d2rq:property e:Publ;
    d2rq:uriPattern "publisher/@@product.publisher|urlify@@";
    .
```

P f é p c n m d *Book* a r a *Publisher* d e i é f a d ú m

5 Trabalhos Futuros

P t e e c e c s p r u e a d D2 i c o c o f p t i s l e a m c c o p a d m q v a r d m c q n f d a

Após e f d i a f w e q f u d JENA [6] p l c d R e no N3 b c c c SPA e t d ex A f t i fun q p a u c s o d u f a e d d b re p u o l e o c d s o d a

Referências

1. P E S K K B M A C J V **StdTrip: An a priori design approach and process for publishing Open Government Data**.
2. Sa E. V V M. Macê J A. Lós B F Lo F L R. C M A. a L F **Towards automatic generation of application ontologies**. In: *Journal of Information and Data Management, Vol. V, No. N, Month* 20YY pp.1-16 2010.
3. Lo F L R.: **Acesso a Dados a partir de Ontologias utilizando Mapeamentos Heterogêneos e Programação em Lógica**. C d Ciên D d C U Fe d C pp 19-77 2011.

4. L A Wöß W.; BLOCHL M.: **A Semantic Web middleware for Virtual Data Integration on the Web**. In of App Knowled Pr e Joh Kepl U L A p.1-15 D n e <http://ftp. aac WS/V 367/ESWC2008_P A e 11 a
5. Bi C. Cyg R.: **D2R server – publishing relational databases on the Semantic Web.** D n e <http://www4.wiw b Cyg D2R-S ISWC2006.pdf > A e 01 a
6. J – A S W F f Java D n e <http://jena.sourceforge.net/ > A e 30 a
7. R TICIANNE G.: **Uma Abordagem Baseada em Ontologias para o Desenvolvimento de Aplicações Web de Integração de Dados**. C d Ciên D d C U Fe d C 2010.
8. B Lee Tim L D 2006 <http://www.w3.org/DesignIssues/LinkedData.html>
9. P Si Ich Ry **Automated Mapping Generation for Converting Databases into Linked Data**.
10. R Raghu 2008: **Sistemas e Gerenciamento de Banco de Dados**