A Direct Mapping of Relational Data to RDF [1] [2]

Albin Ahmeti and Wolfgang Fischl

Semantic Web Technologies

July 4th, 2012

Introduction

- Relational data everywhere on the web
- Convert this data into RDF in order to
  - query the data with SPARQL
  - extend the data
  - connect the data

Introduction

- Relational data everywhere on the web
- Convert this data into RDF in order to
  - query the data with SPARQL
  - extend the data
  - connect the data

- create a mapping from relational databases to RDF
- mapping should
  - convert structure and data
  - have some properties

Literature

Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, and Juan F. Sequeda.
A directly mapping of relational data to rdf.
http://www.w3.org/TR/rdb-direct-mapping, May 2012.

Juan F. Sequeda, Marcelo Arenas, and Daniel P. Miranker.
On directly mapping relational databases to rdf and owl.
In Proceedings of the 21st international conference on World Wide Web, WWW '12, pages 649–658, New York, NY, USA, 2012. ACM.

Outline

A first example

Course

PK		→ *Dept(DID)*
CID	**Name**	**Code**
100	SWT	10

Dept

PK	
DID	**Name**
10	DBAI

direct graph

```
<Course/CID=100> rdf:type <Course> .
<Course/CID=100> <Course#CID> 100 .
<Course/CID=100> <Course#Name> "SWT" .
<Course/CID=100> <Course#Code> 10 .
<Course/CID=100> <Course#ref-Code> <Dept/DID=10> .

<Dept/DID=10> rdf:type <Dept> .
<Dept/DID=10> <Dept#DID> 10 .
<Dept/DID=10> <Dept#Name> "DBAI" .
```

Multicolumn primary keys

Student

PK	
SID	**Name**
1	Albin
2	Wolfgang

Enrolled

→ Studen(SID)	→ Course(CID)
PK	
SID	**CID**
1	100
2	100

direct graph

```
<Student/SID=1> rdf:type <Student> .
<Student/SID=1> <Student#SID> 1 .
<Student/SID=1> <Student#Name> Albin .
<Student/SID=2> rdf:type <Student> .
<Student/SID=2> <Student#SID> 2 .
<Student/SID=2> <Student#Name> Wolfgang .

<Enrolled/SID=1;CID=100> rdf:type <Enrolled> .
<Enrolled/SID=1;CID=100> <Enrolled#SID> 1 .
<Enrolled/SID=1;CID=100> <Enrolled#CID> 100 .
<Enrolled/SID=1;CID=100> <Enrolled#ref-SID>
                          <Student/SID=1> .
<Enrolled/SID=1;CID=100> <Enrolled#ref-CID>
                          <Course/CID=100> .
<Enrolled/SID=2;CID=100> rdf:type <Enrolled> .
<Enrolled/SID=2;CID=100> <Enrolled#SID> 2 .
<Enrolled/SID=2;CID=100> <Enrolled#CID> 100 .
<Enrolled/SID=2;CID=100> <Enrolled#ref-SID>
                          <Student/SID=2> .
<Enrolled/SID=2;CID=100> <Enrolled#ref-CID>
                          <Course/CID=100> .
```

Empty primary key

Tweets

→ *Student(SID)*		
tweeter	**when**	**text**
1	2010-08-30T01:33	I really like RDF!
1	2010-08-30T09:01	RDB2RDF is even more fun!

direct graph

```
_:a rdf:type <Tweets> .
_:a <Tweets#tweeter> 1 .
_:a <Tweets#ref-tweeter> <Student/SID=1> .
_:a <Tweets#when> "2010-08-30T01:33"^^xsd:dateTime .
_:a <Tweets#text> "I really like RDF!" .

_:b rdf:type <Tweets> .
_:b <Tweets#tweeter> 1 .
_:b <Tweets#ref-tweeter> <Student/SID=1> .
_:b <Tweets#when> "2010-08-30T09:01"^^xsd:dateTime .
_:b <Tweets#text> "RDB2RDF is even more fun!" .
```

Basic Definitions

percent-encode

Replace a string with the IRI-safe form
Example: "Hello World!" → "Hello%20World%21"

row node

- Table has a primary key: the row node is a relative IRI containing the table name and all primary key columns
- Table has no primary key: a fresh blank node

Example: `<Student/SID=1>`

table IRI

the relative IRI consisting of the percent-encoded form of the table name
Example: `<Student>`

Basic Definitions

literal property IRI

concatenation of the percent-encoded form of the table name, the hash character '#' and the percent-encoded form of the column name
Example: `<Student#SID>`

reference property IRI

concatenation of the percent-encoded form of the table name, the string "ref-", for each column in the foreign key, the percent-encoded form of the column names
Example: `<Course#ref-Code>`

Definition Direct Graph

direct graph

the union of the table graphs for each table in a database schema

table graph

the union of the row graphs for each row in a table

row graph

- the row type triple
- a reference triple
- a literal triple

Definition Direct Graph

row type triple

- *subject*: the row node of the row
- *predicate*: the RDF IRI `rdf:type`
- *object*: the table IRI for the table name

Example

```
<Student/SID=1> rdf:type <Student> .
```

Definition Direct Graph

literal triple

- *subject*: the row node of the row
- *predicate*: the literal property IRI for the column
- *object*: the RDF literal representation of the columns value

Example

```
<Student/SID=1> <Student#SID> 1 .
```

Definition Direct Graph

reference triple

- *subject*: the row node of the row
- *predicate*: the reference property IRI for the columns
- *object*: the row node of the referenced row

Example

```
<Course/CID=100> <Course#ref-Code> <Dept/DID=10> .
```

Direct Mapping - $\mathcal{DM}$

- Mapping RDB (+schema constraints) to OWL DL in automatic way using finite Datalog rules
- Inspired by the draft of W3C Direct Mapping standard
- Imposes some properties:
  - **Fundamental properties**
    - Information preservation
    - Query preservation
  - **Desirable properties**
    - Monotonicity
    - Semantic preservation

A first example - Recap

Course

PK		→ *Dept(DID)*
CID	**Name**	**Code**
100	SWT	10

Dept

PK	
DID	**Name**
10	DBAI

direct graph

```
<Course/CID=100> rdf:type <Course> .
<Course/CID=100> <Course#CID> 100 .
<Course/CID=100> <Course#Name> "SWT" .
<Course/CID=100> <Course#Code> 10 .
<Course/CID=100> <Course#ref-Code> <Dept/DID=10> .

<Dept/DID=10> rdf:type <Dept> .
<Dept/DID=10> <Dept#DID> 10 .
<Dept/DID=10> <Dept#Name> "DBAI" .
```

A first example - $\mathcal{DM}$

direct graph

```
<Course/CID=100> rdf:type <Course> .
<Course/CID=100> <Course#CID> 100 .
<Course/CID=100> <Course#Name> "SWT" .
<Course/CID=100> <Course#Code> 10 .
<Course/CID=100> <Course#ref-Code> <Dept/
      DID=10> .

<Dept/DID=10> rdf:type <Dept> .
<Dept/DID=10> <Dept#DID> 10 .
<Dept/DID=10> <Dept#Name> "DBAI" .
```

$\mathcal{DM}$

```
<Course> rdf:type owl:Class .
<Course#CID> rdf:type owl:DatatypeProperty .
<Course#CID> rdfs:domain <Course> .
<Course#Name> rdf:type owl:DatatypeProperty .
<Course#Name> rdfs:domain <Course> .
<Course#Code> rdf:type owl:ObjectProperty .
<Course#Code> rdfs:domain <Course> .
<Course#Code> rdfs:range <Dept> .

<Dept> rdf:type owl:Class .
<Dept#DID> rdf:type owl:DatatypeProperty .
<Dept#DID> rdfs:domain <Dept> .
<Dept#Name> rdf:type owl:DatatypeProperty .
<Dept#Name> rdfs:domain <Dept> .


<Course/CID=100> rdf:type <Course> .
<Course/CID=100> <Course#CID> 100 .
<Course/CID=100> <Course#Name> "SWT" .
<Course/CID=100> <Course#Code> <Dept/DID=10> .

<Dept/DID=10> rdf:type <Dept> .
<Dept/DID=10> <Dept#DID> 10 .
<Dept/DID=10> <Dept#Name> "DBAI" .
```

Binary relations in $\mathcal{DM}$

Student

PK	
SID	**Name**
1	Albin
2	Wolfgang

Enrolled

→ Studen(SID)	→ Course(CID)
PK	
SID	**CID**
1	100
2	100

direct graph

```
<Student/SID=1> rdf:type <Student> .
<Student/SID=1> <Student#SID> 1 .
<Student/SID=1> <Student#Name> Albin .
<Student/SID=2> rdf:type <Student> .
<Student/SID=2> <Student#SID> 2 .
<Student/SID=2> <Student#Name> Wolfgang .

<Enrolled/SID=1;CID=100> rdf:type <Enrolled> .
<Enrolled/SID=1;CID=100> <Enrolled#SID> 1 .
<Enrolled/SID=1;CID=100> <Enrolled#CID> 100 .
<Enrolled/SID=1;CID=100> <Enrolled#ref-SID>
                          <Student/SID=1> .
<Enrolled/SID=1;CID=100> <Enrolled#ref-CID>
                          <Course/CID=100> .
<Enrolled/SID=2;CID=100> rdf:type <Enrolled> .
<Enrolled/SID=2;CID=100> <Enrolled#SID> 2 .
<Enrolled/SID=2;CID=100> <Enrolled#CID> 100 .
<Enrolled/SID=2;CID=100> <Enrolled#ref-SID>
                          <Student/SID=2> .
<Enrolled/SID=2;CID=100> <Enrolled#ref-CID>
                          <Course/CID=100> .
```

Binary relations in $\mathcal{DM}$

direct graph

```
<Student/SID=1> rdf:type <Student> .
<Student/SID=1> <Student#SID> 1 .
<Student/SID=1> <Student#Name> Albin .
<Student/SID=2> rdf:type <Student> .
<Student/SID=2> <Student#SID> 2 .
<Student/SID=2> <Student#Name> Wolfgang .

<Enrolled/SID=1;CID=100> rdf:type <Enrolled> .
<Enrolled/SID=1;CID=100> <Enrolled#SID> 1 .
<Enrolled/SID=1;CID=100> <Enrolled#CID> 100 .
<Enrolled/SID=1;CID=100> <Enrolled#ref-SID>
                         <Student/SID=1> .
<Enrolled/SID=1;CID=100> <Enrolled#ref-CID>
                         <Course/CID=100> .
<Enrolled/SID=2;CID=100> rdf:type <Enrolled> .
<Enrolled/SID=2;CID=100> <Enrolled#SID> 2 .
<Enrolled/SID=2;CID=100> <Enrolled#CID> 100 .
<Enrolled/SID=2;CID=100> <Enrolled#ref-SID>
                         <Student/SID=2> .
<Enrolled/SID=2;CID=100> <Enrolled#ref-CID>
                         <Course/CID=100> .
```

$\mathcal{DM}$

```
<Student> rdf:type owl:Class .
<Student#SID> rdf:type owl:DatatypeProperty .
<Student#SID> rdfs:domain <Student> .
<Student#Name> rdf:type owl:DatatypeProperty .
<Student#Name> rdfs:domain <Student> .

<Enrolled#SID,CID,SID,CID> rdf:type
                           owl:ObjectProperty .
<Enrolled#SID,CID,SID,CID> rdfs:domain
                           <Student> .
<Enrolled#SID,CID,SID,CID> rdfs:range
                           <Course> .


<Student/SID=1> rdf:type <Student> .
<Student/SID=1> <Student#SID> 1 .
<Student/SID=1> <Student#Name> Albin .
<Student/SID=1> rdf:type <Student> .
<Student/SID=1> <Student#SID> 2 .
<Student/SID=1> <Student#Name> Wolfgang .

<Student/SID=1> <Enrolled#SID,CID,SID,CID>
                <Course/CID=100> .
<Student/SID=2> <Enrolled#SID,CID,SID,CID>
                <Course/CID=100> .
```

Query Preservation

- Translating SQL to SPARQL seamlessly, i.e. query preservation between a database and an ontology
- Let's do it by Example
- Relational algebra query:

$$\sigma_{Name=Juan}(STUDENT) \bowtie ENROLLED$$

- Translated SPARQL query ?
- Done inductively!

Query Preservation (cont.)

$$\sigma_{Name=Juan}(STUDENT) \bowtie ENROLLED$$

- Relation STUDENT is non-binary relation, hence the following SPARQL construct is used:

```
SELECT ?SID, ?NAME
  WHERE { ?X rdf:type :Student
          OPT { ?X :Student#SID ?SID }
          OPT { ?X :Student#Name ?NAME }
        }
```

Query Preservation (cont.)

$$\sigma_{Name=Juan}(STUDENT) \bowtie ENROLLED$$

- Filter tuples that fulfill the restriction: Name=Juan

```
SELECT ?SID, ?NAME
  WHERE { ?X rdf:type :Student
          OPT { ?X :Student#SID ?SID }
          OPT { ?X :Student#Name ?NAME }
          FILTER ( ?NAME = Juan )
        }
```

Query Preservation (cont.)

$$\sigma_{Name=Juan}(STUDENT) \bowtie ENROLLED$$

- Relation ENROLLED is a binary relation, hence the following SPARQL construct is used:

```
SELECT ?SID, ?CID
  WHERE { ?T1 :Enrolled#SID,CID,SID,CID ?T2 AND
          ?T1 :Student#SID ?SID AND
          ?T2 :Course#CID ?CID AND
        }
```

Query Preservation (cont.)

$$\sigma_{Name=Juan}(STUDENT) \bowtie ENROLLED$$

- Join operator, performing AND among the obtained SPARQL queries

$\sigma_{Name=Juan}(STUDENT) \bowtie ENROLLED$

```
SELECT ?SID, ?NAME, ?CID
WHERE {
  { SELECT ?SID, ?NAME
    WHERE { ?X rdf:type :Student
            OPT { ?X :Student#SID ?SID }
            OPT { ?X :Student#Name ?NAME }
            FILTER ( ?NAME = Juan )
          }
    FILTER (bound(?SID))
  }
    AND
  {
    SELECT ?SID, ?CID
    WHERE { ?T1 :Enrolled#SID,CID,SID,CID ?T2 AND
            ?T1 :Student#SID ?SID AND
            ?T2 :Course#CID ?CID AND
          }
    FILTER (bound(?SID))
  }
}
```

Conclusion

- We've presented two approaches of directly mapping RDBs to ontologies
  - In the first approach RDBs are mapped to RDF following RDB2RDF Working group
  - In the second approach RDBs are mapped to OWL DL following Sequeda et al. paper
- In both cases integrity constraints are taken into consideration
- The second approach has its advantage as
  - It uses the expressivity of OWL DL
    - Classes,
    - ObjectProperties with Domain and Range restriction,
    - DatatypeProperties with Domain restriction
  - Further fundamental and desirable properties are elaborated
- We've showed here the query preservation property

The End

The End

Thank you for your attention!