Direct Mapping

Werner Nutt

Acknowledgment

These slides are based on a slide set by Mariano Rodriguez

Reading Material/Sources

- R2RML specification by W3C
  http://www.w3.org/TR/r2rml/

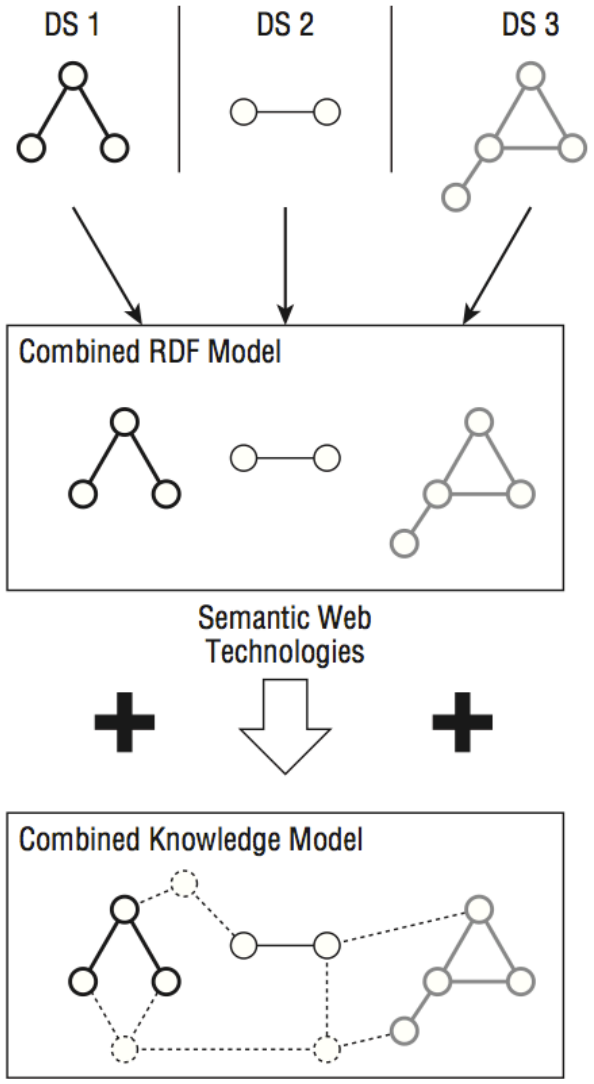
- R2RML specification byW3C
  http://www.w3.org/2001/sw/rdb2rdf/test-cases/

- Idea: Combining Information
- Direct Mapping: Details
- Exercises

- Idea: Combining Information
- Direct Mapping: Details
- Exercises

Goal: Combining Information

- Standard formats for storing information
  - Relational DBs
  - XML
  - Comma/tab delimited files, spreadsheets
  - Proprietary file formats
- Common Data Model: RDF (+ other Semantic Web Technologies) can make it easier to integrate across all formats (not replace)
- Knowledge Model: RDFS (+ OWL) can make it easier to integrate under a common knowledge model



Combining Information

Techniques for relational DBMSs

- Direct Mapping (studied now!)
- R2RML (topic of the next lectures)

Tools

- D2RQ Server
  - developed @ FU Berlin, DERI Ireland
- ontop
  - developed @ UNIBZ

Standards and Tools

Mapping languages

- Standards by RDB2RDF working group (W3C)
  - Direct Mapping
  - R2RML
- Proprietary

Tools

- Free, academic: D2R, Morph, r2rml4net, db2triples, ontop
- Commercial: Virtuoso, ultrawrap, Oracle SW

- Idea: Combining Information
- Direct Mapping: Details
- Exercises

RDF2RF: Direct Mapping

"The direct mapping defines a simple transformation, providing a basis for defining and comparing more intricate transformations.
It can also be used to materialize RDF graphs or define virtual graphs, which can be queried by SPARQL or traversed by an RDF graph API."

Source: A Direct Mapping of Relational Data to RDF
W3C Recommendation 27 September 2012

Usage

- Approach 1: ETL (= Extract Transform Load)
  - Connect the database to a Direct Mapping engine
  - Transform the data into RDF using the engine
  - Load the RDF data into a triple store
- Approach 2: Virtual RDF Graphs
  - Connect the database to a Direct Mapping engine with support for Virtual RDF graphs
  - Start the engine's SPARQL end-point
  - Query the (virtual) RDF Graph using the vocabulary defined by the Direct Mapping transformation

Direct Mapping Idea

People

PK		→ Addresses(ID)
ID	fname	addr
7	Bob	18
8	Sue	NULL

Addresses

PK		
ID	City	State
18	Cambridge	Ma

Convert relational data into RDF, by making explicit the semantics encoded in the relational schema.

Create URIs following some simple rules: Map

- table to class
- column to property
- row to resource
- cell to literal value
- in addition cell to URI
  - if there is a foreign key constraint

Direct Mapping: Consequences

People

PK		→ Addresses(ID)
ID	fname	addr
7	Bob	18
8	Sue	NULL

Addresses

PK		
ID	City	State
18	Cambridge	Ma

We need IRIs for identifying

- the resource class corresponding to a table
- the resources represented by the table rows
- the properties of the resources corresponding to table cells
- the references due to foreign keys

Direct Mapping: Encoding Principles

People

PK		→ Addresses(ID)
ID	fname	addr
7	Bob	18
8	Sue	NULL

Addresses

PK		
ID	City	State
18	Cambridge	Ma

- Base IRI for the whole graph/dataset, e.g.
  @base <http://foo.example/DB/> .
- Table name ➔ Class name, e.g.
  People ➔ <People>
- Row with PK ➔ Resource with PK, e.g,
  <People/ID=7>
- Table row ➔ Property, e.g.,
  <People#ID>
  <People#addr>
- Table cells: what if NULL?
- Foreign key reference ➔ additional property, e.g.,
  <People#ref-addr>

Example: RDF Resulting from Direct Mapping

People

PK		→ Addresses(ID)
ID	fname	addr
7	Bob	18
8	Sue	NULL

Addresses

PK		
ID	City	State
18	Cambridge	Ma

Provide a base IRI http://foo.example/DB/ !

```
@base <http://foo.example/DB/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<People/ID=7> rdf:type <People> .
<People/ID=7> <People#ID> 7 .
<People/ID=7> <People#fname> "Bob" .
<People/ID=7> <People#addr> 18 .
<People/ID=7> <People#ref-addr> <Addresses/ID=18> .

<People/ID=8> rdf:type <People> .
<People/ID=8> <People#ID> 8 .
<People/ID=8> <People#fname> "Sue" .

<Addresses/ID=18> rdf:type <Addresses> .
<Addresses/ID=18> <Addresses#ID> 18 .
<Addresses/ID=18> <Addresses#city> "Cambridge" .
<Addresses/ID=18> <Addresses#state> "MA" .
```

Tables with Primary Keys

- In this expression, each row, e.g. (7, "Bob", 18), produces a set of triples with a common subject. The subject is an IRI formed from the concatenation of the base IRI, table name (People), primary key column name (ID) and primary key value (7). The predicate for each column is an IRI formed from the concatenation of the base IRI, table name and the column name.
- The values are RDF literals formed from the lexical form of the column value. Each foreign key produces a triple with a predicate composed from the foreign key column names, the referenced table, and the referenced column names.
- The object of these triples is the row identifier (<Addresses/ID=18>) for the referenced triple. Note that these reference row identifiers must coincide with the subject used for the triples generated from the referenced row.
- The direct mapping does not generate triples for NULL values. Note that it is not known how to relate the behavior of the obtained RDF graph with the standard SQL semantics of the NULL values of the source RDB.

Source: A Direct Mapping of Relational Data to RDF
W3C Recommendation 27 September 2012

Percent-Encoding

Definition of percent-encoding:

- Replace the string with the “IRI-safe form” per section 7.3 of [R2RML].

String	IRI-safe version
42	42
Hello World!	Hello%20World%21
2011-08-23T22:17:00Z	2011-08-23T22%3A17%3A00Z
~A_17.1-2	~A_17.1-2
葉篤正	葉篤正

Percent-Encoding

A percent-encoding mechanism is used to represent a data octet in a component when that octet's corresponding character is outside the allowed set or is being used as a delimiter of, or within, the component. A percent-encoded octet is encoded as a character triplet, consisting of the percent character "%" followed by the two hexadecimal digits representing that octet's numeric value. For example, "%20" is the percent-encoding for the binary octet "00100000" (ABNF: %x20), which in US-ASCII corresponds to the space character (SP).

Source: RFC 3986, Section 2.1

Foreign Keys to Composite Keys

People

PK		→ Addresses(ID)	→ Department(name, city)	
ID	fname	addr	deptName	deptCity
7	Bob	18	accounting	Cambridge
8	Sue	NULL	NULL	NULL

Department

PK	Unique Key		
ID	name	city	manager
23	accounting	Cambridge	8

Addresses

PK	→ People(ID)	
ID	City	State
18	Cambridge	Ma

RDF

```
@base <http://foo.example/DB/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<People/ID=7> rdf:type <People> .
<People/ID=7> <People#ID> 7 .
<People/ID=7> <People#fname> "Bob" .
<People/ID=7> <People#addr> 18 .
<People/ID=7> <People#ref-addr> <Addresses/ID=18> .
<People/ID=7> <People#deptName> "accounting" .
<People/ID=7> <People#deptCity> "Cambridge" .
<People/ID=7> <People#ref-deptName;deptCity> <Department/ID=23> .
<People/ID=8> rdf:type <People> .
<People/ID=8> <People#ID> 8 .
<People/ID=8> <People#fname> "Sue" .

<Addresses/ID=18> rdf:type <Addresses> .
<Addresses/ID=18> <Addresses#ID> 18 .
<Addresses/ID=18> <Addresses#city> "Cambridge" .
<Addresses/ID=18> <Addresses#state> "MA" .

<Department/ID=23> rdf:type <Department> .
<Department/ID=23> <Department#ID> 23 .
<Department/ID=23> <Department#name> "accounting" .
<Department/ID=23> <Department#city> "Cambridge" .
<Department/ID=23> <Department#manager> 8 .
<Department/ID=23> <Department#ref-manager> <People#ID=8> .
```

In reference property, concatenate the names of the referring attributes with ";"

Multi-Column Primary Keys

People

PK		→ Addresses(ID)	→ Department(name, city)	
ID	fname	addr	deptName	deptCity
7	Bob	18	accounting	Cambridge
8	Sue	NULL	NULL	NULL

Department

PK		
name	city	manager
accounting	Cambridge	8

Addresses

PK		
ID	City	State
18	Cambridge	Ma

Multi-Column Primary Keys/2

Primary keys may also be composite. If, in the above example, the primary key for Department were (name, city) instead of ID, the identifier for the only row in this table would be

<Department/name=accounting;city=Cambridge>.

The triples involving <Department/ID=23> would be replaced with the following triples:

<People/ID=7> <People#ref-deptName;deptCity> <Department/name=accounting;city=Cambridge> .
<Department/name=accounting;city=Cambridge> rdf:type <Department> .
<Department/name=accounting;city=Cambridge> <Department#ID> 23 .
<Department/name=accounting;city=Cambridge> <Department#name> "accounting" .
<Department/name=accounting;city=Cambridge> <Department#city> "Cambridge" .

Empty (Non-existing) Primary Keys

Tweets

→ People(ID)		
tweeter	when	text
7	2010-08-30T01:33	I really like lolcats.
7	2010-08-30T09:01	I take it back.

Table Tweets does not have a primary key …

➔ How can we represent rows as resources?

Empty (Non-existing) Primary Keys

Tweets

→ People(ID)		
tweeter	when	text
7	2010-08-30T01:33	I really like lolcats.
7	2010-08-30T09:01	I take it back.

If there is no primary key, each row determines a set of triples with a shared subject, but that subject is a blank node.

```
@base <http://foo.example/DB/>
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

_:a rdf:type <Tweets> .
_:a <Tweets#tweeter> "7" .
_:a <Tweets#ref-tweeter> <People/ID=7> .
_:a <Tweets#when>
                "2010-08-30T01:33"^^xsd:dateTime .
_:a <Tweets#text> "I really like lolcats." .

_:b rdf:type <Tweets> .
_:b <Tweets#tweeter> "7" .
_:b <Tweets#ref-tweeter> <People/ID=7> .
_:b <Tweets#when>
                 "2010-08-30T09:01"^^xsd:dateTime .
_:b <Tweets#text> "I take it back." .
```

Summary: Row Nodes

- If the table has a primary key, the row node is a relative IRI obtained by concatenating:
  - the percent-encoded form of the table name,
  - the SOLIDUS character '/',
  - for each column in the primary key, in order:
    - the percent-encoded form of the column name,
    - an EQUALS SIGN character '='
    - the percent-encoded lexical form of the canonical RDF literal representation of the column value as defined in R2RML section 10.2 Natural Mapping of SQL Values [R2RML],
    - if it is not the last column in the primary key, a SEMICOLON character ';'
- If the table has no primary key, the row node is a fresh blank node that is unique to this row.

Source: A Direct Mapping of Relational Data to RDF
W3C Recommendation 27 September 2012

Referencing Tables with Empty Primary Keys

Rows in tables with no primary key may still be referenced by foreign keys. (Relational database theory tells us that these rows must be unique as foreign keys reference candidate keys and candidate keys are unique across all the rows in a table.) References to rows in tables with no primary key are expressed as RDF triples with blank nodes for objects, where that blank node is the same node used for the subject in the referenced row.

Source: A Direct Mapping of Relational Data to RDF
W3C Recommendation 27 September 2012

Referencing Tables with Empty Primary Keys

Projects

Unique Key			
	Unique Key		
→ People(ID)		→ Department(name,city)	
lead	name	deptName	deptCity
8	pencil survey	accounting	Cambridge
8	eraser survey	accounting	Cambridge

TaskAssignments

PK			
	→ Projects(name, deptName, deptCity)		
→ People(ID)		→ Department(name,city)	
worker	project	deptName	deptCity
7	pencil survey	accounting	Cambridge

```
@base <http://foo.example/DB/>
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

_:c rdf:type <Projects> .
_:c <Projects#lead> <People/ID=8> .
_:c <Projects#name> "pencil survey" .
_:c <Projects#deptName> "accounting" .
_:c <Projects#deptCity> "Cambridge" .
_:c <Projects#ref-deptName;deptCity> <Department/ID=23> .

_:d rdf:type <Projects> .
_:d <Projects#lead> <People/ID=8> .
_:d <Projects#name> "eraser survey" .
_:d <Projects#deptName> "accounting" .
_:d <Projects#deptCity> "Cambridge" .
_:d <Projects#ref-deptName;deptCity> <Department/ID=23> .

<TaskAssignments/worker=7;project=pencil%20survey> rdf:type <TaskAssignments> .
<TaskAssignments/worker=7;project=pencil%20survey> <TaskAssignments#worker> 7 .
<TaskAssignments/worker=7;project=pencil%20survey> <TaskAssignments#ref-worker> <People/ID=7> .
<TaskAssignments/worker=7;project=pencil%20survey> <TaskAssignments#project> "pencil survey" .
<TaskAssignments/worker=7;project=pencil%20survey> <TaskAssignments#deptName> "accounting" .
<TaskAssignments/worker=7;project=pencil%20survey> <TaskAssignments#deptCity> "Cambridge" .
<TaskAssignments/worker=7;project=pencil%20survey> <TaskAssignments#ref-deptName;deptCity>
                                                   <Department/ID=23> .
<TaskAssignments/worker=7;project=pencil%20survey> <TaskAssignments#ref-project;deptName;deptCity> _:c .
```

- Idea: Combining Information
- Direct Mapping: Details
- Exercises

Exercises

… based on

R2RML and Direct Mapping Test Cases
W3C Editor's Draft 24 July 2012

http://www.w3.org/2001/sw/rdb2rdf/test-cases/

Case 1: Referencing a Table with Primary Key

Source

PK		
	→ Target(key2attr2, key2attr1)	
ID	attrA	attrB
1100	K2A2	K2A1

Target

PK	Unique (key1attr1, key1attr2)		Unique (key2attr2, key2attr1)	
PK	key1attr1	key1attr2	key2attr1	key2attr2
1010	K1A1	K1A2	K2A1	K2A2

Case 1: Solution Steps

- Fix a base URI
- Encode the row in the table Target
  - Introduce a URI/blank node for the row
  - Introduce a suitable class
  - Encode the components of the row,
    using suitable URIs for encoding the attributes of Source
- Encode the row in the table Source
  - First, proceed as for Target
  - Encode the foreign key reference

Case 1: Solution

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://example.com/base/> .

<Target/PK=1010> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Target> .
<Target/PK=1010> <Target#PK> 1010 .
<Target/PK=1010> <Target#key1attr1> "K1A1" .
<Target/PK=1010> <Target#key1attr2> "K1A2" .
<Target/PK=1010> <Target#key2attr1> "K2A1" .
<Target/PK=1010> <Target#key2attr2> "K2A2" .

<Source/ID=1100> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Source> .
<Source/ID=1100> <Source#ID> 1100 .
<Source/ID=1100> <Source#attrA> "K2A2" .
<Source/ID=1100> <Source#attrB> "K2A1" .
<Source/ID=1100> <Source#ref-attrA;attrB> <Target/PK=1010> .

Case 2: Referencing a Table with Empty Primary Key

Source

PK		
	→ Target(key2attr2, key2attr1)	
ID	attrA	attrB
1100	K2A2	K2A1

Target

	Unique (key1attr1, key1attr1)		Unique (key2attr2, key2attr1)	
litattr1	key1attr1	key1attr2	key2attr1	key2attr2
1010	K1A1	K1A2	K2A1	K2A2

Case 2: Solution Steps

- Fix a base URI
- Encode the row in the table Target
  - Introduce a URI/blank node for the row
  - Introduce a suitable class
  - Encode the components of the row,
    using suitable URIs for encoding the attributes of Source
- Encode the row in the table Source
  - First, proceed as for Target
  - Encode the foreign key reference

Case 2: Solution

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://example.com/base/> .

_:a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Target> .
_:a <Target#litattr1> 1010 .
_:a <Target#key1attr1> "K1A1" .
_:a <Target#key1attr2> "K1A2" .
_:a <Target#key2attr1> "K2A1" .
_:a <Target#key2attr2> "K2A2" .

<Source/ID=1100> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Source> .
<Source/ID=1100> <Source#ID> 1100 .
<Source/ID=1100> <Source#attrA> "K2A2" .
<Source/ID=1100> <Source#attrB> "K2A1" .
<Source/ID=1100> <Source#ref-attrA;attrB> _:a .
```

Case 3: Foreign Key to Row with Nulls

PK		
	→ Target(key2attr2, key2attr1)	
ID	attrA	attrB
1100	K2A21	K2A11
1100	K2A22	NULL

PK	Unique (key1attr1, key1attr1)		Unique (key2attr2, key2attr1)	
PK	key1attr1	key1attr2	key2attr1	key2attr2
1010	K1A11	K1A21	K2A11	K2A21
1011	K1A12	K1A22	NULL	K2A22

Case 3: Solution

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://example.com/base/> .

<Target/PK=1010> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Target> .
<Target/PK=1010> <Target#PK> 1010 .
<Target/PK=1010> <Target#key1attr1> "K1A11" .
<Target/PK=1010> <Target#key1attr2> "K1A21" .
<Target/PK=1010> <Target#key2attr1> "K2A11" .
<Target/PK=1010> <Target#key2attr2> "K2A21" .

<Target/PK=1011> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Target> .
<Target/PK=1011> <Target#PK> 1011 .
<Target/PK=1011> <Target#key1attr1> "K1A12" .
<Target/PK=1011> <Target#key1attr2> "K1A22" .
<Target/PK=1011> <Target#key2attr2> "K2A22" .

Case 3: Solution (cont.)

<Source/ID=1100> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Source> .
<Source/ID=1100> <Source#ID> 1100 .
<Source/ID=1100> <Source#attrA> "K2A21" .
<Source/ID=1100> <Source#attrB> "K2A11" .
<Source/ID=1100> <Source#ref-attrA;attrB> <Target/PK=1010> .

<Source/ID=1101> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <Source> .
<Source/ID=1101> <Source#ID> 1101 .
<Source/ID=1101> <Source#attrA> "K2A22" .