RDF and Relational Databases

Mapping Relational data to RDF

Suppose we have data in a relational database that we want to export as RDF:
1. Choose an RDF vocabulary to represent the data
2. Define a mapping from the relational tables to RDF
Then either:
   a) Materialize the RDF triples from the database using the mappings
   b) Use a server to dynamically access the relational data given a SPARQL query
   c) Use a DBMS that directly supports RDF (e.g., Oracle 11g, DB2)

D2RQ

- D2RQ exposes relational data as RDF
- see http://d2rq.org/

- D2RQ mapping language file describes the relation between ontology and RDB
- D2RQ server provides HTML and linked data views and a SPARQL 1.1 endpoint
- D2RQ engine uses mappings to rewrite Jena & Sesame API calls to SQL queries and generates RDF dumps in various formats

D2RQ Features

- Browsing database contents: Web interface for navigation through the RDF contents for people
- Resolvable URIs: D2R Server assigns a resolvable URI to each entity in the database
- Content negotiation: HTML & RDF versions share URIs; HTTP content negotiation fixes version
- SPARQL: Both an endpoint & explorer provided
- BLOBs and CLOBs: Support for serving up values as files (e.g., PDFs, images)
- Not surprisingly, no inferencing
D2RQ Mapping Language

- The mapping is defined in RDF
- D2RQ can generate a default mapping using a standard heuristic
  - Each database table has information about one type of thing
  - Each row in a table represents one object
  - The first column is the key => defines the object
  - The other columns represent properties
- You can edit the default mapping or create your own by hand

A simple database

```sql
mysql> use lab; show tables;
+---------------+
| Tables_in_lab |
+---------------+
| people        |
+---------------+
```

```sql
mysql> desc people;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Name   | varchar(50) | NO   | PRI |         |       |
| Age    | int(11)     | YES  |     | NULL    |       |
| Mobile | varchar(50) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

```sql
mysql> select * from people;
+---------------+------+--------------+
| Name          | Age  | Mobile       |
+---------------+------+--------------+
| Al Turing     | 32   | 443-253-3863 |
| Don Knuth     | 25   | 410-228-6282 |
| Chuck Babbage | 38   | 410-499-1282 |
+---------------+------+--------------+
```

The default model

- The people table has info of things of type people
  `<http://ebiq.org/o/labvocab/resource/people>`
- Each row in the table has information about one instance of a person
- The first column is the key and is used both
  - As the identifier for a person instance
    `<http://localhost/people/Chuck_Babbage>`
  - For the rdf:label for a person instance
- Properties of a person are: name, age & mobile
  `<http://ebiq.org/o/labvocab/resource/people_Age>`

The database table

```sql
mysql> use lab; show tables;
+---------------+
| Tables_in_lab |
+---------------+
| people        |
+---------------+
```

```sql
mysql> desc people;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Name   | varchar(50) | NO   | PRI |         |       |
| Age    | int(11)     | YES  |     | NULL    |       |
| Mobile | varchar(50) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

```sql
mysql> select * from people;
+---------------+------+--------------+
| Name          | Age  | Mobile       |
+---------------+------+--------------+
| Al Turing     | 32   | 443-253-3863 |
| Don Knuth     | 25   | 410-228-6282 |
| Chuck Babbage | 38   | 410-499-1282 |
+---------------+------+--------------+
```
Generating RDF mappings

- D2RQ can generate a default mapping directly from the database
  % generate-mapping –u demo –p demo \
  -b [http://ebiq.org/o/lab] \
  'jdbc:mysql://127.0.0.1/lab’
- The –b arg is the base url for the RDF vocabulary used in publishing the table
- The last argument is the string that JDBC uses to reference the database table
- The resulting mapping can be edited as desired

The Default D2RQ mapping

```
@prefix ... 
Map:database a d2rq:Database; 
d2rq:jdbcDriver "com.mysql.jdbc.Driver"; 
d2rq:jdbcDriver "jdbc:mysql://127.0.0.1/lab"; 
d2rq:username "demo"; 
d2rq:password "demo"; 
d2rq:autoReconnect "true"; 
d2rq:zeroDateTimeBehavior "convertToNull"; 
map:people a d2rq:ClassMap; 
d2rq:dataStorage map:database; 
d2rq:uriPattern "people/@@people.Name|urlify@@"; 
d2rq:class vocab:people; 
d2rq:classDefinitionLabel "people"; 
map:people__label a d2rq:PropertyBridge; 
d2rq:belongsToClassMap map:people; 
d2rq:property rdfs:label; 
d2rq:pattern "people #@@people.Name@@"; 
map:people_Name a d2rq:PropertyBridge; 
d2rq:belongsToClassMap map:people; 
d2rq:property vocab:people_Name; 
d2rq:propertyDefinitionLabel "people Name"; 
d2rq:datatype xsd:int; 
map:people_Mobile a d2rq:PropertyBridge; 
d2rq:belongsToClassMap map:people; 
d2rq:property vocab:people_Mobile; 
d2rq:propertyDefinitionLabel "people Mobile"; 
```

Run the D2RQ Server

d2r-server -p 8080 ../mapping-lab.n3

Access via D2R server

- Explore via HTML
- Via SPARQL endpoint

```
D2R Server
Running at http://localhost:8080/
```

1. HTML View
   You can use the navigation links at the top of this page to explore the database.

2. RDF View
   You can also explore this database with Semantic Web browsers like Tabulator or Dap
   To start browsing, open this entry point URL in your Semantic Web browser:
   http://localhost:8080/0000

3. SPARQL Endpoint
   SPARQL clients can query the database at this SPARQL endpoint:
   http://localhost:8080/javax_sparql_endpoint
   The database can also be explored using this AJAX-based SPARQL Explorer.

---

3
Access via D2R server
Via SPARQL endpoint

Content Negotiation
- D2RQ automatically recognizes URIs for
  - Entities (e.g., an RDF object like a class or instance)
    http://localhost:8080/resource/people/AI_Turing
  - RDF representations
    http://localhost:8080/data/people/AI_Turing
  - HTML representations
    http://localhost:8080/page/people/AI_Turing
- The HTTP protocol supports content negotiation
- A get request can specify what kind of content it wants, e.g., HTML or RDF

Resources and 303 redirects
- Asking for a raw resource doesn’t make sense – it’s just an identifier
- But we can specify in the HTTP header what kind of content we want, e.g. HTML or RDF
- If client gets a 303 (redirect) it knows where to go
- For example:
  303 See Other: For a description of this item, see http://localhost:8080/page/people/AI_Turing

  303 See Other: For a description of this item, see http://localhost:8080/data/people/AI_Turing

URIs should be de-referenceable
- Linked Data best practice says that LOD URIs should be dereferenceable
- Doing a GET on one should always yield useful information
Asking for RDF data

```ruby
% curl http://localhost:8080/data/people/Al_Turing
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix vocab: <http://ebiq.org/o/labvocab/resource/> .
<http://localhost:8080/data/people/Al_Turing>
  rdfs:label "RDF Description of people #Al Turing" ;
vocab:people
<http://localhost:8080/resource/people/Al_Turing>
  a vocab:people ;
  vocab:people_Age "32"^^xsd:int ;
  vocab:people_Mobile "443-253-3863" ;
  vocab:people_Name "Al Turing" .
```

Asking for HTML

```ruby
% curl http://localhost:8080/page/people/Al_Turing
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title> people #Al Turing | D2R Server </title>
<link rel="stylesheet" type="text/css" href="http://localhost:8080/snorql/style.css" />
<link rel="alternate" type="text/rdf+n3" href="http://localhost:8080/data/people/Al_Turing?output=n3" title="This page in RDF (N3)" />
</head>
```

The iswc example

- D2RQ comes with a partial example database and mapping for information about ISWC
  - Stop the server
  - d2r-server -p 8080 ../mapping-iswc.n3
  - Visit http://localhist:8080/

ISWC Database

- The ISWC database has partial information about the 2002 ISWC conference
- It's a richer schema going beyond the simple auto generated mapping
- http://sw.cs.technion.ac.il/d2rq/tutorial had detailed instructions on installing on your computer
- And sample queries you can run

```sql
mysql> use iswc; show tables;
+-------------------------+
| Tables_in_iswc         |
+-------------------------+
| conferences            |
| organizations          |
| papers                 |
| persons                |
| rel_paper_topic        |
| rel_person_organization|
| rel_person_paper       |
| rel_person_topic       |
| topics                 |
+-------------------------+
9 rows in set (0.00 sec)
```
Generating RDF dumps

- Once the mapping is defined, use dump-rdf to for RDF dumps in various formats
- For example:

  % dump-rdf -m ../mapping-iswc.n3 -f N3

Oracle Database Semantic Data Store

- Introduced in Oracle 10g, also in 11g
- An open and persisted RDF data model and analysis platform for semantic applications
- An RDF Data Model with inferencing (RDFS, OWL and user-defined rules)
- Performs SQL-based access to triples and inferred data
- Combines SQL query of relational data with RDF graphs and ontologies
- Scalable: supports large graphs (billion+ triples)
- Support for Special queries

RDB2RDF Working Group

- [http://www.w3.org/2001/sw/rdb2rdf/](http://www.w3.org/2001/sw/rdb2rdf/)
- Mission: standardize languages for mapping relational data and schemas into RDF and OWL
- It is developing two languages: R2RML and Direct Mapping
  - Direct mapping is like D2RQ’s automatic schema
  - R2RML is the language for expressing custom mappings
- Preliminary recommendations for both were published in March, final recommended status expected in Summer 2012