

# 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

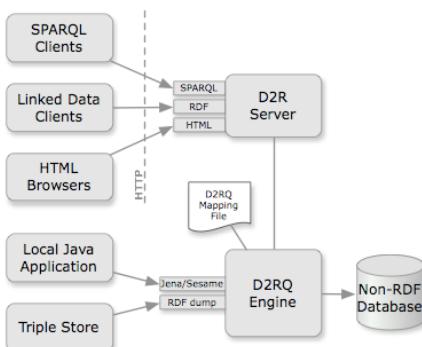
Then either:

- 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
- *D2R 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

```
mysql> use lab; show tables;
+-----+
| Tables_in_lab |
+-----+
| people       |
+-----+
mysql> desc people;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Name  | varchar(50) | NO  | PRI |          |       |
| Age   | int(11)    | YES |     | NULL   |       |
| Mobile | varchar(50) | YES |     | NULL   |       |
+-----+-----+-----+-----+-----+-----+
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>](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>](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>](http://ebiq.org/o/labvocab/resource/people_Age)

## The database table

```
mysql> use lab; show tables;
+-----+
| Tables_in_lab |
+-----+
| people       |
+-----+
mysql> desc people;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Name  | varchar(50) | NO  | PRI |          |       |
| Age   | int(11)    | YES |     | NULL   |       |
| Mobile | varchar(50) | YES |     | NULL   |       |
+-----+-----+-----+-----+-----+-----+
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:jdbcDSN "jdbc:mysql://127.0.0.1/lab";  
d2rq:username "demo";  
d2rq:password "demo";  
jdbc:autoReconnect "true";  
jdbc:zeroDateTimeBehavior "convertToNull";.  
map:people a d2rq:ClassMap;  
d2rq:dataStorage map:database;  
d2rq:uriPattern "people/@@{people.Name}|urify@@";  
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:column "people.Name";.  
map:people_Age a d2rq:PropertyBridge;  
d2rq:belongsToClassMap map:people;  
d2rq:property vocab:people_Age;  
d2rq:propertyDefinitionLabel "people Age";  
d2rq:column "people.Age";.  
d2rq:datatype xsd:int;.  
map:people_Mobile a d2rq:PropertyBridge;  
d2rq:belongsToClassMap map:people;  
d2rq:property vocab:people_Mobile;  
d2rq:propertyDefinitionLabel "people Mobile";  
d2rq:column "people.Mobile";.
```

## Run the D2RQ Server

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

## Access via D2R server

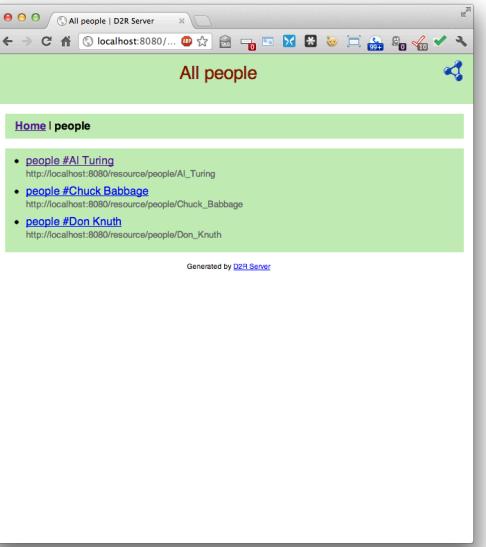
- Explore via HTML
- Via SPARQL endpoint

The screenshot shows a web browser window titled 'D2R Server' with the URL 'localhost:8080'. The page has a green header bar with the title 'D2R Server' and the subtext 'Running at http://localhost:8080/'. Below the header, there's a navigation bar with links for 'Home', 'people', and other options. The main content area is divided into sections:

- 1. HTML View:** Text: 'This is a database published with D2R Server. It can be accessed using  
1. your plain old web browser  
2. Semantic Web browsers  
3. SPARQL clients.'
- 2. RDF View:** Text: 'You can also explore this database with **Semantic Web browsers** like [Tabulator](#) or [Disco](#). To start browsing, open this entry point URL in your Semantic Web browser:  
<http://localhost:8080/all>
- 3. SPARQL Endpoint:** Text: 'SPARQL clients can query the database at this SPARQL endpoint:  
<http://localhost:8080/sparql>  
The database can also be explored using this [AJAX-based SPARQL Explorer](#).

**Access via D2R server**

- Explore via HTML
- Via SPARQL endpoint



```

All people

Home | people



- people #AI Turing
      http://localhost:8080/resource/people/AI_Turing
- people #Chuck Babbage
      http://localhost:8080/resource/people/Chuck_Babbage
- people #Don Knuth
      http://localhost:8080/resource/people/Don_Knuth


Generated by D2R Server

```

**Access via D2R server**

- Explore via HTML
- Via SPARQL endpoint



Property	Value
rdfs:label	people #AI Turing
vocab:people_Age	32 (xsd:int)
vocab:people_Mobile	443-253-3863
vocab:people_Name	AI Turing
rdf:type	vocab:people

```

people #AI Turing

Resource URI: http://localhost:8080/resource/people/AI_Turing

Home | All people

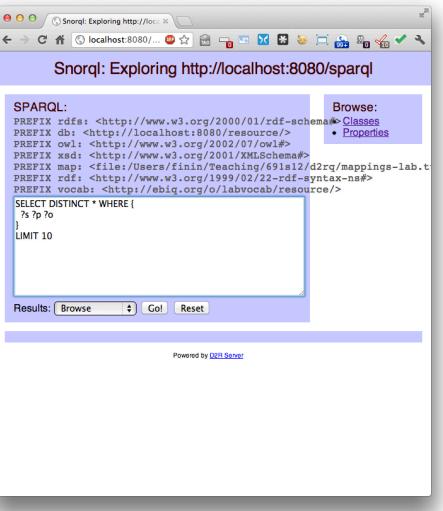
Property Value
rdfs:label people #AI Turing
vocab:people_Age 32 (xsd:int)
vocab:people_Mobile 443-253-3863
vocab:people_Name AI Turing
rdf:type vocab:people

Generated by D2R Server

```

**Access via D2R server**

Via SPARQL endpoint



```

Snorql: Exploring http://localhost:8080/sparql

SPARQL:
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://localhost:8080/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX map: <file:/Users/fimin/Teaching/691s12/d2rq/mappings-lab.t
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX vocab: <http://ebiq.org/o/labvocab/resource/>

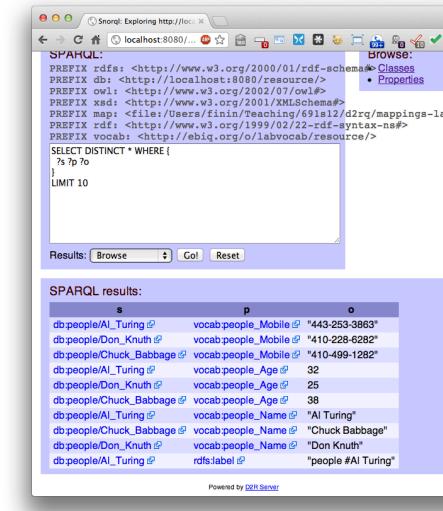
SELECT DISTINCT * WHERE {
    ?s ?p ?o
}
LIMIT 10

Results: Browse Go! Reset
Powered by D2R Server

```

**Access via D2R server**

Via SPARQL endpoint



SPARQL:

```

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://localhost:8080/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX map: <file:/Users/fimin/Teaching/691s12/d2rq/mappings-lab.t
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX vocab: <http://ebiq.org/o/labvocab/resource/>
SELECT DISTINCT * WHERE {
    ?s ?p ?o
}
LIMIT 10

```

BROWSE:

- Classes
- Properties

SPARQL results:

s	p	o
db:people/AI_Turing	vocab:people_Mobile	"443-253-3863"
db:people/Don_Knuth	vocab:people_Mobile	"410-228-6282"
db:people/Chuck_Babbage	vocab:people_Mobile	"410-499-1282"
db:people/AI_Turing	vocab:people_Age	32
db:people/Don_Knuth	vocab:people_Age	25
db:people/Chuck_Babbage	vocab:people_Age	38
db:people/AI_Turing	vocab:people_Name	"AI Turing"
db:people/Chuck_Babbage	vocab:people_Name	"Chuck Babbage"
db:people/Don_Knuth	vocab:people_Name	"Don Knuth"
db:people/AI_Turing	rdfs:label	"people #AI Turing"

## Access via D2R server

Via SPARQL endpoint

The screenshot shows the Snorql web interface. At the top, it says "Snorql: Exploring http://localhost:8080/sparql". Below that is a code editor containing a SPARQL query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db: <http://localhost:8080/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX map: <file:///Users/farin/Teaching/691e12/d2rq/mappings-lab.t
PREFIX rdfs: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX vocab: <http://ehbg.org/o/labvocab/resource/>

SELECT DISTINCT ?who ?phone WHERE {
  ?who vocab:Mobile ?phone
}
LIMIT 10
```

Below the code editor is a table titled "SPARQL results:":

who	phone
db:people/Al_Turing	"443-253-3863"
db:people/Don_Knuth	"410-228-6282"
db:people/Chuck_Babbage	"410-499-1282"

At the bottom of the interface, it says "Powered by D2RQ Server".

## Content Negotiation

- D2RQ automatically recognizes URIs for
  - Entities (e.g., an RDF object like a class or instance)  
http://localhost:8080/resource/people/Al\_Turing
  - RDF representations  
http://localhost:8080/data/people/Al\_Turing
  - HTML representations  
http://localhost:8080/page/people/Al\_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

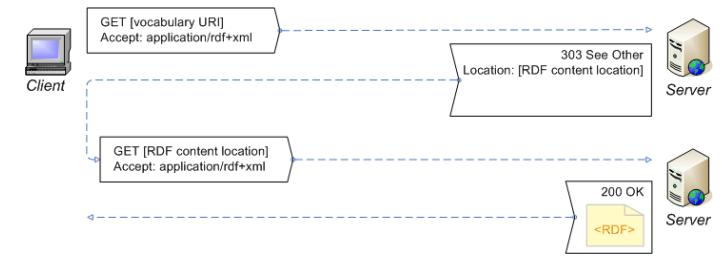
- Asking for a raw resource doesn't make sense – it's just an identifier
- But we can specify in the HTTP header what kind of content we want, e.g. HTML or RDF
- If client gets a 303 (redirect) it knows where to go
- For example:

```
% curl -H "Accept: text/html" http://localhost:8080/resource/people/Al_Turing
303 See Other: For a description of this item, see http://localhost:8080/page/people/Al\_Turing
```

```
% curl -H "Accept: application/rdf+xml" http://localhost:8080/resource/people/Al_Turing
303 See Other: For a description of this item, see http://localhost:8080/data/people/Al\_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

```
% curl http://localhost:8080/data/people/AI_Turing
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix vocab: <http://ebiq.org/o/labvocab/resource/> .

<http://localhost:8080/data/people/AI_Turing>
  rdfs:label "RDF Description of people #AI Turing" ;
  foaf:primaryTopic <http://localhost:8080/resource/people/AI_Turing> .

vocab:people
  rdfs:seeAlso <http://localhost:8080/sparql?query=DESCRIBE+%3Chttp%3A
%2F%2Febiq.org%2Fo%2Fvocab%2Fresource%2Fpeople%3E> .

<http://localhost:8080/resource/people/AI_Turing>
  a    vocab:people ;
  rdfs:label "people #AI Turing" ;
  vocab:people_Age "32"^^xsd:int ;
  vocab:people_Mobile "443-253-3863" ;
  vocab:people_Name "AI Turing" .
```

## Asking for HTML

```
% curl http://localhost:8080/page/people/AI_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 #AI Turing | D2R Server </title>
  <link rel="stylesheet" type="text/css" href="http://localhost:8080/snorql/
style.css" />
  <link rel="alternate" type="application/rdf+xml" href="http://localhost:8080/
data/people/AI_Turing?output=rdfxml" title="This page in RDF (XML)" />
  <link rel="alternate" type="text/rdf+n3" href="http://localhost:8080/data/people/
AI_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://localhost: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

```
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 generate 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/>
- 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