D2RQ Platform—Treating Non-RDF Databases as Virtual RDF Graphs

Tutorial at the SWAT4LS Workshop
Berlin, December 9th, 2010

Radoslaw Oldakowski
FU Berlin
Agenda

- Introduction

- D2RQ: Overview

- Using D2RQ-Platform
  - Mapping Definition
  - RDF dump
  - Accessing Data using Jena or Sesame API
  - Turning RDB into an SPARQL-Endpoint
  - Publishing RDB content as linked Data
Introduction

- A lot of valuable data resides in relational databases
- As Semantic Web technologies are getting mature, there is a growing need for RDF applications to access the content of non-RDF, legacy databases
- Integrating various databases is often a motivation for adopting RDF
Introduction: Relational Data vs RDF

| People | | | |
|---|---|---|
| ID | name | email |
| 12 | Chris | chris@bizer.de |

| Papers | | | |
|---|---|---|
| ID | title | confID |
| 312 | D2R Server | 132 |

| Rel_People_Papers | | | |
|---|---|---|
| PersonID | PaperID |
| 12 | 312 |
Introduction: Relational Data vs RDF

People

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Chris</td>
<td><a href="mailto:chris@bizer.de">chris@bizer.de</a></td>
</tr>
</tbody>
</table>

Papers

<table>
<thead>
<tr>
<th>ID</th>
<th>title</th>
<th>confID</th>
</tr>
</thead>
<tbody>
<tr>
<td>312</td>
<td>D2R Server</td>
<td>132</td>
</tr>
</tbody>
</table>

Rel_People_Papers

<table>
<thead>
<tr>
<th>PersonID</th>
<th>PaperID</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>312</td>
</tr>
</tbody>
</table>

ex:Person12 rdf:type foaf:Person
ex:Person12 faof:name Chris
ex:Person12 faof:mbox mailto:chris@bizer.de

ex:Papers312 dc:author ex:Person12
ex:Papers312 swc:conference ex:Conf312
ex:Papers312 dc:title D2R Server
Introduction: Integrating relational DB into Semantic Web

**What is needed**

- A way to map between the relational data model and RDF
  - Data Model heterogeneity
    - relations vs. Graph
    - Primary keys vs. URIs
  - Schema heterogeneity
    - Name vs. Given name, Surname
  - Syntactic heterogeneity
    - Timestamp vs. “2010-12-09”^^xsd:data
- Software to make the resulting RDF data accessible

**D2RQ platform offers**

- D2RQ Mapping Language
- D2R Server
D2RQ Platform: What is it offer?

- Allows applications to access the content of non-RDF legacy databases
- By treating relational databases as virtual RDF-Graphs
- Without the need to replicate the whole database into RDF
D2RQ: Project Background

- **Developed at the Feie Universität Berlin since 2004**
  - latest release (10/08/09): D2RQ Engine 0.7 and D2RQ Server 0.7
  - (29/11/10): SPARQL/Update Extension

- **Developers:**
  - Chris Bizer, Richard Cyganiak, Christian Becker, Andreas Langegger, Herwig Leimer, …

- **Free Software License:** Apache 2.0
- **13,300 downloads**
- **Project URL:**
  - [http://www4.wiwiss.fu-berlin.de/bizer/d2rq/](http://www4.wiwiss.fu-berlin.de/bizer/d2rq/)
D2RQ: Architecture and Interfaces
Database Compatibility

- **Tested/works well with**
  - Oracle, MySQL, PostgreSQL, Microsoft SQL Server

- **SQL-92 compatible databases**
  - D2RQ will interact with the database using SQL-92 standard by default

- **ODBC data sources (e.g. Microsoft Access)**
  - Connection over the ODBC-JDBC bridge
  - Limitations:
    - Mapping generator does not work
    - Automatic detection of column types does not work → manual specification needed
  - By using a dedicated JDBC driver the problems can be avoided

- **Other databases**
  - not tested
Acceptable performance on databases with hundred thousand to a few million records

Performance depends on access method
- RDF dumps, access through RDF API, and Linked Data and HTML interfaces scale well

Performance of SPARQL->SQL rewriting depends on query complexity
- queries only including basic triple patterns show performance similar to hand-written SQL-queries
- Additional SPARQL features (e.g. OPTIONAL, FILTER, LIMIT) may result in worse performance
D2RQ Limitations

- Integration of multiple databases or other data sources
- CREATE/DELETE/UPDATE operation
  - SPARQL/Update extension released
- Inference
- Support for Named Graphs
D2RQ: Architecture and Interfaces

- SPARQL Clients
- Linked Data Clients
- HTML Browsers
- Local Java Application
- Triple Store
  - Jena/Sesame
  - RDF dump

- D2RQ Server
  - SPARQL
  - RDF
  - HTML
  - D2RQ Mapping File

- D2RQ Engine
  - Non-RDF Database
D2RQ Mapping Language Specification

- Declarative language for describing the relations of a relational database schema and RDFS vocabularies or OWL ontologies

- Each mapping is represented in an RDF document

- Formally defined by the D2RQ RDFS Schema
  - D2RQ namespace:
    http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#
Example of a D2RQ Map
This tutorial will cover in detail the following central constructs of the mapping language:

- `d2rq:Database`
- `d2rq:ClassMap`
- `d2rq:PropertyBridge`
- `d2rq:TranslationTable`
- `d2rq:Configuration`
Example of a D2RQ Map
D2RQ Mapping Language: d2rq:Database

- Defines the connection to a local database
- Specifies the type of database columns used by D2RQ
- A D2RQ map can contain several d2rq:Database for accessing different local databases
D2RQ Mapping Language: Properties of d2rq:Database

- **d2rq:jdbcDSN**
  - JDBC database URL
  - The value of this property is a string of the form: `jdbc:subprotocol:subname`
  - Example for MySQL: `jdbc:mysql://hostname:port/DBname`

- **d2rq:jdbcDriver**
  - JDBC driver java class for the used database
  - Example for MySQL: `com.mysql.jdbc.Driver`

- **d2rq:odbcDSN**
  - ODBC data source name of the database

- **d2rq:username**

- **d2rq:password**
D2RQ Mapping Language: Properties of d2rq:Database

- **d2rq:resultSizeLimit**
  - Integer value which will be added to all SQL queries
  - Sets an upper bound for the number of results retrieved from large databases
  - WARNING: effectively cripples the server and may cause unpredictable results

- **d2rq:fetchSize**
  - number of rows retrieved with every database request
  - important for controlling memory consumption of D2RQ and database server
  - default value set to 500, or to `Integer.MIN_VALUE` for MySQL → enables streaming mode
D2RQ Mapping Language: Properties of d2rq:Database

- **d2rq:allowDistinct**
  - Specifies the database ability to handle SQL DISTINCT
    - for example MSAccess cuts fields longer than 256 characters
  - Values: true or false

- **d2rq:textColumn, d2rq:numericColumn, d2rq:dateColumn, d2rq:timestampColumn**
  - used to declare the type of database columns
  - property values: column names in the form of `Table_name.Column_name`
  - only need to be specified if the D2RQ Engine is not able to determine the column type automatically
  - d2rq:timestampColumn: columns combining date & time
Example

map:ExampleDatabase 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".
• Specifying additional JDBC connection properties
  • Many JDBC drivers offer a wide range of connection properties (see JDBC driver’s documentation)
  • Those can be specified in a D2RQ map using the jdbc: namespace (URI: http://d2rq.org/terms/jdbc/)

@prefix jdbc: <http://d2rq.org/terms/jdbc/> .

map:database a d2rq:Database;
# ... other database configuration ...
jdbc:autoReconnect "true";
jdbc:zeroDateTimeBehavior "convertToNull";
jdbc:keepAlive "3600"; # value in seconds
jdbc:keepAliveQuery "SELECT 1"; # (optionally to override default noop query).
Example of a D2RQ Map

map:Paper_ClassMap
  :dataStorage
  :belongsToClassMap

map:Database
  :belongsToClassMap

map:Author_ClassMap
  :belongsToClassMap

map:title_PropertyBridge
  :column "Paper.title"

map:abstract_PropertyBridge
  :column "Paper.abstract"

map:weblink_PropertyBridge
  :column "Paper.weblink"

map:author_PropertyBridge
  :join "Paper.author=Author.ID"

map:name_PropertyBridge
  :pattern "%%Author.first%% %%Author.last%%"

map:email_PropertyBridge
  :urlPattern "mailto:%%Author.email%%"

map:article_PropertyBridge
  :urlColumn "Paper.article"

map:doi_PropertyBridge
  :urlPattern "doi:%%Paper.doi%%"

map:reference_PropertyBridge
  :urlPattern "%%Paper.reference%%"

map:agent_PropertyBridge
  :urlPattern "%%Paper.agent%%"

map:paper_PropertyBridge
  :urlPattern "%%Paper.paper%%"

map:location_PropertyBridge
  :urlPattern "%%Paper.location%%"

map:series_PropertyBridge
  :urlPattern "%%Paper.series%%"

map:publisher_PropertyBridge
  :urlPattern "%%Paper.publisher%%"

map:isPartOf_PropertyBridge
  :urlPattern "%%Paper.isPartOf%%"

map:hasPart_PropertyBridge
  :urlPattern "%%Paper.hasPart%%"

map:hasContent_PropertyBridge
  :urlPattern "%%Paper.hasContent%%"

map:hasResource_PropertyBridge
  :urlPattern "%%Paper.hasResource%%"

map:hasVersion_PropertyBridge
  :urlPattern "%%Paper.hasVersion%%"

map:hasFormat_PropertyBridge
  :urlPattern "%%Paper.hasFormat%%"

map:hasLanguage_PropertyBridge
  :urlPattern "%%Paper.hasLanguage%%"

map:hasExtent_PropertyBridge
  :urlPattern "%%Paper.hasExtent%%"

map:hasContentType_PropertyBridge
  :urlPattern "%%Paper.hasContentType%%"

map:hasDuration_PropertyBridge
  :urlPattern "%%Paper.hasDuration%%"

map:hasPublicationPropertyBridge
  :urlPattern "%%Paper.hasPublicationPropertyBridge%%"
D2RQ Mapping Language: d2rq:ClassMap

- Represents a class or a group of similar classes of an OWL ontology or RDFS Schema
- Specifies how instances of a class are identified
- It connects to a d2rq:Database
- Has a set of d2rq:PropertyBridges which attach properties to the instances
Identifying instances of a class

- **URI patterns**

  - Resource URIs are generated by inserting certain column values into a pattern, e.g.:

  - If database columns contain characters not allowed in URIs, their values have to be encoded first before put into a URI pattern:
    - `$@@Table.Column|urlencode@@` → URL encoding will be applied
    - `$@@Table.Column|urlify@@` → URL encoding + converts spaces to underscore `_`
Identifying instances of a class

- Relative URI patterns
  - e.g. \texttt{persons/@@Persons.ID@@}
  - generate relative URIs, which combined with a \textit{baseURI} will form a full URI
    - e.g. given the \texttt{baseURI} \url{http://localhost/vocabulary/}
      and \texttt{Person.ID=1234}
      the corresponding full URI will be \url{http://localhost/vocabulary/persons/1234}
  - Useful in creation of portable mappings used for multiple instances of the same database schema

- URI columns
  - using column values already contains URIs, e.g. web page or document URL
• **Identifying instances of a class**
  • **Blank Nodes** (RDF Resources without a global identifier)
    - In D2RQ Blank Nodes may be generated from one or more columns
  • **Singleton classmaps**
    - In some cases it might be desirable to only produce a single resource with fixed, static identity.
D2RQ Mapping Language: Properties of d2rq:ClassMap

- **d2rq:dataStorage**
  - Reference to a d2rq:Database

- **d2rq:class**
  - RDFS or OWL class
  - All Resources generated by this ClassMap will be instances of this class

- **d2rq:uriPattern**
  - URI pattern used to generate relative or full URI identifying instances generated by this ClassMap

- **d2rq:uriColumn**
  - Database column containing URIs
  - Format: TableName.ColumnName
D2RQ Mapping Language: Properties of d2rq:ClassMap

- **d2rq:bNodedColumns**
  - comma-separated list of columns (TableName.ColumnName)
  - one distinct blank node per distinct tuple of these columns

- **d2rq:constantValue**
  - generates a single instance for this class map, named by the value of this property (i.e. URI or BlankNode)

- **d2rq:translateWith**
  - assigns a translation table to this ClassMap

- **d2rq:containsDuplicates**
  - Must be set to "true" if a database table is not fully normalized
  - D2RQ adds DISTINCT clause to all queries using this ClassMap
D2RQ Mapping Language: Properties of d2rq:ClassMap

- **d2rq:condition**
  - specifies an SQL WHERE condition
  - is used for example to hide some parts of the database contents (e.g. old records) from D2RQ

- **d2rq:classDefinitionLabel**
  - Label which will be served as rdfs:label

- **d2rq:classDefinitionComment**
  - Comment that will be served as rdfs:comment

- **d2rq:additionalClassDefinitionProperty**
  - Used to add fixed statements to all class definitions of a ClassMap
ClassMap where instances are identified using an URI pattern

Papers

<table>
<thead>
<tr>
<th>PaperID</th>
<th>Title</th>
<th>Abstract</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Trusting Information Sources One Citizen at a Time</td>
<td>This paper describes an approach to derive assessments abx</td>
</tr>
<tr>
<td>2</td>
<td>Automatic Generation of Java/SOL based Inference Engines from RDF Schema and RuleML</td>
<td>This paper describes two approaches for automatically conv</td>
</tr>
<tr>
<td>3</td>
<td>Three Implementations of SquishQL, a Simple RDF Query Language</td>
<td>RDF provides a basic way to represent data for the Semantic</td>
</tr>
<tr>
<td>4</td>
<td>A Data Integration Framework for E-commerce</td>
<td>A marketplace is the place in which the demand and su</td>
</tr>
</tbody>
</table>

map:PaperClassMap a d2rq:ClassMap;
d2rq:dataStorage map:Database1;
d2rq:uriPattern
d2rq:class :Paper;
d2rq:classDefinitionLabel "paper"@en;
d2rq:classDefinitionComment "A conference paper."@en.
D2RQ Mapping Language: d2rq:ClassMap Examples

- **ClassMap where instances are identified using Blank Nodes**

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

- For recognizing Blank Nodes within an RDF API, D2RQ generates a BlanNode label (ClassMap name + primary key)
  - e.g. [http://www.example.org/dbserver01/db01#Topic@@6](http://www.example.org/dbserver01/db01#Topic@@6)
### Conditional mapping

```plaintext
map:Paper a d2rq:ClassMap;
    d2rq:dataStorage map:Database1
d2rq:class :Paper;
d2rq:uriPattern
    d2rq:condition "Papers.Publish = 1"; .
```
- Additional property for a class definition

```turtle
map:PersonsClassMap a d2rq:ClassMap;
   d2rq:class :Person;
   d2rq:additionalClassDefinitionProperty
      map:PersonEquivalence.

map:PersonEquivalence a d2rq:AdditionalProperty;
   d2rq:propertyName owl:equivalentClass;
   d2rq:propertyValue foaf:Person.
```
Example of a D2RQ Map
D2RQ Mapping Language: d2rq:PropertyBridge

- Maps database table columns to RDF properties
- Attached to a resources created by a ClassMap
- Property values are either Literals or other Resources
- If the column value is `NULL` for some table records, no property will be created for the resource
D2RQ Mapping Language: Properties of d2rq:PropertyBridge

- **d2rq:belongsToClassMap**
  - maps a property Bridge to a ClassMap
  - must be specified for every PropertyBridge

- **d2rq:property**
  - defines an RDF property which connects instances created by a ClassMap with the object or Literal created by the bridge
  - must be specified for every PropertyBridge
  - multiple d2rq:properties are allowed

- **d2rq:dproperty**
  - URI pattern used to generate the Property URI at runtime
  - Multiple d2rq:dproperty are allowed
**D2RQ Mapping Language: Properties of d2rq:PropertyBridge**

- **d2rq:column, d2rq:uriColumn**
  - Indicates database columns (TableName.ColumnName) that contain literal values / URIs

- **d2rq:pattern, d2rq:uriPattern**
  - Used to extend and combine (several) column values to generate Literals / URIs

- **d2rq:sqlExpression, d2rq:uriSqlExpression**
  - The value of the property is created by evaluation this SQL expression
  - NOTE: the output of d2rq:uriSqlExpression must produce valid URI
  - May put heavy load on the database
D2RQ Mapping Language: Properties of d2rq:PropertyBridge

- **d2rq:datatype**
  - Specifies the datatype of literals

- **d2rq:lang**
  - Specifies the language tag of literals

- **d2rq:refersToClassMap**
  - For properties that correspond to a foreign key
  - The instances of the referenced ClassMap will be used as values of the bridge
  - d2rq:join have to be specified for selecting the correct instances
- **d2rq:constantValue**
  - Properties that have the same value (Literal, URI, BlankNode) for all instances of a ClassMap

- **d2rq:join**
  - Used if the property values are created using (additional) columns which are not in ClassMap‘s columns

- **d2rq:alias**
  - Form: `Table As Alias`
  - Used when a table needs to be joined to itself
  - Aliases can be used to refer to a Table within a PropertyBridge
**D2RQ Mapping Language: Properties of d2rq:PropertyBridge**

- **d2rq:condition**
  - SQL WHERE condition
  - D2RQ will only generate properties if this condition holds
  - E.g. suppress properties with empty literal values
    
    \[ \text{d2rq:condition} \mid \text{Table.Column <> ' '}. \]

- **d2rq:translateWith**
  - Assigns a translation table
  - Values from d2rq:column or d2rq:pattern will be translated

- **d2rq:valueContains, d2rq:valueMaxLength, d2rq:valueRegex**
  - assert that all values of this bridge contain a given string / are no longer than a given length / match a given regular expression
  - Used to optimise performance
D2RQ Mapping Language: Properties of d2rq:PropertyBridge

- **d2rq:propertyDefinitionLabel, d2rq:propertyDefinitionComment**
  - Label / comment which will be served as rdfs:label / rdfs:comment

- **d2rq:additionalPropertyDefinitionProperty**
  - Used to add fixed statements to all property definitions of a PropertyBridge

- **d2rq:limit, d2rq:limitInverse**
  - Maximum number of results to retrieve form database by this property bridge

- **d2rq:orderAsc, d2rq:orderDesc**
  - The column after which results for this property bridge are sorted
A simple property bridge

```xml
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;
```
D2RQ Mapping Language: d2rq:ClassMap Examples

- ClassMap for a group of classes with the same property

```turtle
map:PersonsClassMap a d2rq:ClassMap ;
  d2rq:uriColumn "Persons.URI" ;
  d2rq:dataStorage map:Database1 .

map:PersonsType a d2rq:PropertyBridge ;
  d2rq:belongsToClassMap map:PersonsClassMap ;
  d2rq:property rdf:type ;
```
D2RQ Mapping Language: d2rq:PropertyBridge Examples

- using information from another DB table

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

Radoslaw Oldakowski

Freie Universität Berlin

CSW Corporate Services for Web

STI • BERLIN

INNOREGION

D2RQ Mapping Language: d2rq:PropertyBridge Examples

**mailto: URIs**

```sparql
map:PersonsClassEmail a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:PersonsClassMap;
  d2rq:property :email;
  d2rq:uriPattern "mailto:@@Persons.Email@@";
```

**computing mailbox hashes**

```sparql
map:UserEmailSHA1 a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:User;
  d2rq:property foaf:mbox_sha1sum;
  d2rq:sqlExpression "SHA1(CONCAT('mailto:',user.email))";
```
D2RQ Mapping Language: d2rq:PropertyBridge Examples

- **URIs generated by an SQL expression**

```sparql
map:HomepageURL a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:PersonsClassMap;
  d2rq:property foaf:homepage;
  d2rq:uriSqlExpression
  "CONCAT('http://www.company.com/homepages/', user.username)";
```

- **Joining a table to itself using d2rq:alias**

```sparql
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";
```
### Linking instances from two database tables

```
map:PaperConference a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Paper;
  d2rq:property :conference;
  d2rq:refersToClassMap map:Conference;
  d2rq:join "Papers.Conference => Conferences.ConfID";

map:Conference a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "conferences/@@conferences.ConfID@@";
  d2rq:class iswc:Conference;.
```

- **the value of the property**: \texttt{conference} will be a resource generated by the ClassMap Conference
- Adding a constant property to each instance of a ClassMap

```r
map:PersonsClassMap a d2rq:ClassMap;
   d2rq:class :Person; .

map:seeAlsoBridge a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:PersonsClassMap;
   d2rq:property rdfs:seeAlso;
   d2rq:constantValue
```
### Conditional mapping

```plaintext
map:primaryTopic a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Paper;
  d2rq:property :primaryTopic;
  d2rq:refersToClassMap map:Topic;
  d2rq:join "Rel_Paper.TopicID => Topics.TopicID";
  d2rq:condition "Rel_Paper.TopicID.RelationType = 1".

map:secondaryTopic a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Paper;
  d2rq:property :secondaryTopic;
  d2rq:refersToClassMap map:Topic;
  d2rq:join "Rel_Paper.TopicID => Topics.TopicID";
  d2rq:condition "Rel_Paper.TopicID.RelationType = 2".
```
D2RQ Mapping Language: d2rq:TranslationTable

- Additional layer between database and RDF world
- Can be attached to a d2rq:ClassMap or d2rq:PropertyBridge using d2rq:translateWith property
- Can only be used for 1:1 mappings
- The translation must be provided for each possible value, if a translation is missing no property will be generated
- Example: translation of Color codes R → red
D2RQ Mapping Language: Properties of d2rq:TranslationTable

- **d2rq:translation**
  - Adds a d2rq:Translation to the d2rq:TranslationTable
  - Properties of d2rq:Translation
    - d2rq:databaseValue
    - d2rq:rdfValue

- **d2rq:href**
  - Link to a CSV file containing translations
  - Each line is a translation in the form: databaseValue, rdfValue

- **d2rq:javaClass**
  - Qualified name of the Java class performing the mapping
  - Must implement the Translator interface: `de.fuberlin.wiwiss.d2rq.values.Translator`
## Translating color codes

map:ColorBridge a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:ShinyObjectMap;
d2rq:property :color;
d2rq:uriColumn "ShinyObject.Color";
d2rq:translateWith map:ColorTable.

map:ColorTable a d2rq:TranslationTable;
d2rq:translation [ d2rq:databaseValue "R"; d2rq:rdfValue :red; ];
# ... more translations omitted ...
d2rq:translation [ d2rq:databaseValue "B"; d2rq:rdfValue :blue; ].
D2RQ Mapping Language: d2rq:Configuration

- **Controls global behaviour of D2RQ**
- **d2rq:serveVocabularies**
  - `true` (default) if inferred and user-supplied vocabularies should be served
- **d2rq:useAllOptimizations**
  - Specifies if bleeding edge optimization should be used
  - `false` by default
  - Activating optimizations:

```r
map:Configuration a d2rq:Configuration;
d2rq:useAllOptimizations true.
```
D2RQ: Architecture and Interfaces
Command Line Tools

- **Mapping generator**
  - Creates a default mapping file by analysing the existing database schema
    - Uses table names as class names and column names as property names

- **RDF dump**
  - Writes the entire database content into a single RDF file

- **Both work on Windows and Unix systems**

- **d driverclass**
  - fully qualified Java class name of the database driver
  - drivers for MySQL and PostgreSQL are included in D2RQ release
  - the jar file has to be in D2RQ's `/lib/db-drivers/` directory
  - Examples:
    - MySQL: `com.mysql.jdbc.Driver`
    - PostgreSQL: `org.postgresql.Driver`
    - Oracle: `oracle.jdbc.OracleDriver`

- **o output.n3**
  - Generated mapping will be stored in this file in N3 Syntax
  - If parameter not specified the output will be written to standard out
Command Line Tools: Mapping Generator

```bash
generate-mapping [-u username] [-p password] [-d driverclass] 
    [-o outfile.n3] [-b baseURI] jdbcURL
```

- **b baseURI**
  - Used to construct vocabulary namespace which is automatically served as Linked Data by D2R-Server, following the convention `http://baseURI/vocab/resource/`
  - Should match the baseURI used to invoke the server
  - Default: `http://localhost:2020/`

- **jdbcURL**
  - Refer to the JDBC driver documentation for the format of the used database engine
  - Examples:
    - MySQL: `jdbc:mysql://servername/databasename`
    - Oracle: `jdbc:oracle:thin:@servername:1521:databasename`
Command Line Tools: Mapping Generator

**Example invocation for a local MySQL database „iswc“**

```
generate-mapping -u root -p test -d com.mysql.jdbc.Driver
-o mapping.n3 jdbc:mysql://127.0.0.1/iswc
```

**Generated output:**

```n3
... 
@prefix vocab: <http://localhost:2020/vocab/resource/> . 
@prefix d2rq: <http://www.wiwiss.fu-erlin.de/suhl/bizer/D2RQ/0.1#> . 
@prefix jdbc: <http://d2rq.org/terms/jdbc/> . 

map:database a d2rq:Database;
   d2rq:jdbcDriver "com.mysql.jdbc.Driver";
   d2rq:jdbcDSN "jdbc:mysql://127.0.0.1/iswc";
   d2rq:username "root";
   d2rq:password "test";
   jdbc:autoReconnect "true";
   jdbc:zeroDateTimeBehavior "convertToNull"; .

# Table conferences
map:conferences a d2rq:ClassMap;
   d2rq:dataStorage map:database;
   d2rq:uriPattern "conferences/@conferences.ConfID@";
   d2rq:class vocab:conferences;
   d2rq:classDefinitionLabel "conferences"; .
```
Customizing the auto-generated mapping

Your data will be more understandable to Semantic Web clients if you replace the auto-generated terms with terms from well-known RDF vocabularies.

```sparql
@prefix vocab:<http://localhost:2020/vocab/resource/> .
...
map:persons_Email a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:persons;
d2rq:property vocab:persons_Email;
d2rq:column "persons.Email";
d2rq:propertyDefinitionLabel "persons Email"; .
```

```
map:persons_Email a d2rq:PropertyBridge;
d2rq:belongsToClassMap map:Persons;
d2rq:property foaf:mbox;
d2rq:uriPattern mailto:@@persons.Email@@"; .
```
Command Line Tools: RDF Dump

```
dump-rdf [-m mapping.n3] [-u username] [-p password]  
          [-d driverclass] [-b baseURI] [-o outfile] [-f format]  
          [-s fetchsize] jdbcURL
```

-m mapping.n3
- mapping used to translate database contents into RDF
- if not specified:
  - generate-mapping will be invoked first and default mapping used instead
  - database connection must be specified on the command line

-f format
- RDF serialization syntax
- Supported formats: "RDF/XML" (default), "RDF/XML-ABBREV", "N3", "N-Triple"  
  (works best for large databases)

-s fetchSize
- number of rows retrieved with every database request
- important for controlling memory consumption of D2RQ and database server
- default value set to 500, or to `Integer.MIN_VALUE` for MySQL \(\rightarrow\) enables streaming mode
Command Line Tools: RDF Dump

dump-rdf [-m mapping.n3] [-u username] [-p password]
[-d driverclass] [-b baseURI] [-o outfile] [-f format]
[-s fetchsize] jdbcURL

- Example invocation for a local MySQL database „iswc“

dump-rdf -m mapping-iswc.n3 -f RDF/XML -b http://localhost:2020/
-o iswc.rdf

- Generated output:

```xml
<?xml version="1.0"?>
<rdf:RDF
    xmlns:iswc="http://annotation.semanticweb.org/iswc/iswc.daml#"
    <rdf:Description rdf:about="persons/10">
        <iswc:research_interests rdf:resource="topics/10"/>
        <iswc:research_interests rdf:resource="topics/15"/>
        <iswc:has_affiliation rdf:resource="organizations/9"/>
        <foaf:mbox rdf:resource="mailto:bergamaschi.sonia@unimo.it"/>
        <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
        <iswc:phone>+39 059 2056132</iswc:phone>
        <iswc:address>DII- Universita di Modena e Reggio Emilia via Vignolese 905 41100 Modena</iswc:address>
        <iswc:phone>+39 059 2056132</iswc:phone>
    </rdf:Description>
```

Radoslaw Oldakowski
D2RQ: Architecture and Interfaces

![Diagram of D2RQ architecture and interfaces]

- **SPARQL Clients**
- **Linked Data Clients**
- **HTML Browsers**
- **Local Java Application**
- **Triple Store**
- **D2RQ Server**
  - SPARQL
  - RDF
  - HTML
- **D2RQ Engine**
- **Non-RDF Database**
  - D2RQ Mapping File
  - Jena/Sesame
  - RDF dump
Using D2RQ with Jena

- **Jena 2 Semantic Web Toolkit:**
  - RDF API
  - Reading and writing RDF (RDF/XML, N3, N-Triple)
  - OWL API
  - In-memory and persistent storage
  - SPARQL query engine

- **D2RQ requires Jena 2.5 or higher**
  - All required jar files are included in the D2RQ distribution
GraphD2RQ/CachingGraphD2RQ provide a read-only statement centric view on a Jena Graph backed by a non-RDF database

```java
// Load mapping file
Model mapping = FileManager.get().loadModel("doc/example/mapping-iswc.n3");

// Set up the GraphD2RQ
GraphD2RQ g = new GraphD2RQ(mapping, "http://localhost:2020/");

// Create a find(spo) pattern
Node subject = Node.ANY;
Node predicate = DC.date.asNode();
Node object = Node.createLiteral("2003", null, XSDDatatype.XSDgYear);
Triple pattern = new Triple(subject, predicate, object);

// Query the graph Iterator
Iterator it = g.find(pattern);

// Output query results
while (it.hasNext()) {
    Triple t = (Triple) it.next();
    System.out.println("Published in 2003: " + t.getSubject());
}
```
Using D2RQ with Jena: Model API

- ModelD2RQ provides a read-only resource centric view on a Jena model backed by a D2RQ-mapped non-RDF database.

```java
// Set up the ModelD2RQ using a mapping file
Model m = new ModelD2RQ("file:doc/example/mapping-iswc.n3");

// Find anything with an rdf:type of iswc:InProceedings
StmtIterator paperIt = m.listStatements(null, RDF.type, ISWC.InProceedings);

// List found papers and print their titles
while (paperIt.hasNext()) {
    Resource paper = paperIt.nextStatement().getSubject();
    System.out.println("Paper: " + paper.getProperty(DC.title).getString());
}

// List authors of the paper and print their names
StmtIterator authorIt = paper.listProperties(DC.creator);
while (authorIt.hasNext()) {
    Resource author = authorIt.nextStatement().getResource();
    System.out.println("Author: " + author.getProperty(FOAF.name).getString());
}
System.out.println();
```
Using D2RQ with Jena: SPARQL queries

- SPARQL queries against a D2RQ model are processed by Jena’s ARQ query engine

```java
ModelD2RQ m = new ModelD2RQ("file:doc/example/mapping-iswc.n3");
String sparql =
  "PREFIX dc: <http://purl.org/dc/elements/1.1/>" +
  "PREFIX foaf: <http://xmlns.com/foaf/0.1/>" +
  "SELECT ?paperTitle ?authorName WHERE {
    " +
  " ?author foaf:name ?authorName ." +
  "})";

Query q = QueryFactory.create(sparql);
ResultSet rs = QueryExecutionFactory.create(q, m).execSelect();
while (rs.hasNext()) {
  QuerySolution row = rs.nextSolution();
  System.out.println("Title: " + row.getLiteral("paperTitle").getString());
  System.out.println("Author: " + row.getLiteral("authorName").getString());
}
```
D2RQ: Architecture and Interfaces
D2R Server

- Tool for publishing the content of relational databases on Semantic Web

- Access methods:
  - Linked Data interface
  - SPARQL interface
  - HTML interface
**D2R Server: Easy setup**

- **You need:**
  - Java 1.4 or newer
  - Modern browser: Firefox, Opera, Safari

- **Installation**
  - Download D2R Server: [http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/#download](http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/#download)
  - Download appropriate JDBC Driver
  - Create a D2RQ mapping file: generate-mapping
  - Start the Server: `d2r-server mapping.n3`

- **Test / Explore**
D2R Server: Test and Explore

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.

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 Disco. To start browsing, open this entry point URL in your Semantic Web browser:
   http://localhost:2020/all

3. SPARQL Endpoint
SPARQL clients can query the database at this SPARQL endpoint:
   http://localhost:2020/sparql

The database can also be explored using this AJAX-based SPARQL Explorer.
D2R Server: SPARQL interface

- Search and query the database using SPARQL query language over the SPARQL protocol
  
  **SPARQL query:**
  
  ```
  PREFIX dc: <http://purl.org/dc/elements/1.1/>
  SELECT ?book ?who
  WHERE { ?book dc:creator ?who }
  ```

  **HTTP request sent to the SPARQL-Endpoint:** http://localhost:2020/sparql/
  
  ```
  GET /sparql/?query=EncodedQuery HTTP/1.1
  Host: localhost:2020
  User-agent: my-sparql-client/0.1
  ```
D2R Server: SPARQL interface

- Search and query the database using SPARQL query language over the SPARQL protocol
  
  - HTTP response

```xml
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="book"/>
    <variable name="who"/>
  </head>
  <results distinct="false" ordered="false">
    <result>
      <binding name="who">r29392923r2922</binding>
    </result>
    ...
  </results>
</sparql>
```
D2R Server: SPARQL Explorer


SPARQL:

PREFIX do: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vocab: <http://www.w3.org/2001/vcard-rdf/3.0#>
PREFIX dotems: <http://purl.org/do/terms/>
PREFIX lwo: <http://annotation.semanticweb.org/lwo/lwo.daml#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#

SELECT DISTINCT ?personName ?topicName WHERE {
  ?person a:foaf:Person.
  ?person a:foaf:name ?personName.
}

Results

<table>
<thead>
<tr>
<th>personName</th>
<th>topicName</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Yolanda GIL&quot;</td>
<td>&quot;Knowledge Representation Languages&quot;</td>
</tr>
<tr>
<td>&quot;Yolanda GIL&quot;</td>
<td>&quot;Knowledge Systems&quot;</td>
</tr>
<tr>
<td>&quot;Yolanda GIL&quot;</td>
<td>&quot;Artificial Intelligence&quot;</td>
</tr>
<tr>
<td>&quot;Varun Rathnakar&quot;</td>
<td>&quot;Semantic Web&quot;</td>
</tr>
<tr>
<td>&quot;Varun Rathnakar&quot;</td>
<td>&quot;Semantic Web Languages&quot;</td>
</tr>
</tbody>
</table>

Powered by D2R Server
D2R Server: Lined Data interface

- Dereferencing URIs
- Content negotiation

D2R Example:
- Information resource (RDF/XML representation) http://localhost:2020/data/persons/12
D2R Server: Lined Data interface

- **RDF/XML representation of**
  
  `http://localhost:2020/resource/persons/12`

```xml
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
...
<rdf:Description rdf:about="http://localhost:2020/data/persons/12">
    <rdfs:label>RDF Description of Christian Bizer</rdfs:label>
    <foaf:primaryTopic>
            <foaf:name>Christian Bizer</foaf:name>
            <rdfs:label>Christian Bizer</rdfs:label>
            <iswc:address>Freie Universität Berlin</iswc:address>
        </iswc:Researcher>
    </foaf:primaryTopic>
...</rdf:RDF>
```
D2R: HTML interface

- HTML representation of

  http://localhost:2020/resource/persons/12
D2R: HTML interface

- The HTML output can be customized by editing the velocity templates
  - found in D2R’s directory `\webapp\WEB-INF\templates`

```xml
<?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>All $classmap | $server_name</title>
    <link rel="stylesheet" type="text/css" href="${home_link}snorql/style.css"/>
    <link rel="alternate" type="application/rdf+xml" href="$rdf_link?output=rdfxml" title="This page in RDF (XML)"/>
    <link rel="alternate" type="text/rdf+n3" href="$rdf_link?output=n3" title="This page in RDF (N3)"/>
</head>
<body class="browser">
    <div id="rdficon"><a href="$rdf_link" title="RDF data"><img src="${home_link}snorql/rdf_flyer.24.gif" alt="[RDF data]" /></a></div>
    <div id="header">
        ...
```
Related Work

- [http://esw.w3.org/topic/RdfAndSql](http://esw.w3.org/topic/RdfAndSql)

  While SQL databases are sometimes used for *StoringRDF* (a *Large Triple Store*), it’s also quite interesting to take *SQL stores that were not designed with RDF in mind and export* them as RDF. With SPARQL, the connection becomes even more interesting. Some implementations map the database contents to an RDF vocabulary that is created automatically from the database schema. Others require a manual mapping of tables and columns to RDF properties and classes, but support use of existing vocabularies without an external rules engine.

  **SPARQL-based**

  - Datar Server: SPARQL and [LinkedData](http://www.verse7.com) over HTTP, automatic, highly customizable mapping. Uses DJRQ (see below).
  - SparqlRDF - SPARQL over API or HTTP; automatic or manual mapping. SquirrelRDF exports LDAP as well as SQL via SPARQL. See [Adapting Architecture](http://esw.w3.org/topic/RdfAndSql) for more along those lines.
  - SQLRDF - SPARQL over MySQL, automatic mapping.
  - RDQL over SQL and SPARQL (it) over GQL, automatic mapping (same as RelationalOwl).
  - OrisGrid - a semantic Grid system, includes a SPARQL rewriting component using Datalog-like rules (more details in a [PPT presentation](http://esw.w3.org/topic/RdfAndSql) and a [Conference paper](http://esw.w3.org/topic/RdfAndSql).)
  - Virtuoso Universal Server - SPARQL-based *Declarative Metaschema Languages* for exposing any Virtuoso-housed data, as well as HTTP, ODBC, JDBC, and otherwise-accessible SQL and XML data sources as URI dereferencable *Linked RDF Instance Data*. RDF data may be retrieved as RDF/XML, JSON, ...etc.
  - J2R - provides dynamic access to Prolog knowledge bases, which may wrap SQL queries, calls to web services, XML parsing, etc.
  - Asso Data Collaboration Server - from Cambridge Semantics, uses SPARQL to do integrated queries of RDBMSes in addition RDF stores, LDAP directories, etc. Also pushes RDF updates to SQL inserts/updates when possible.

  **SPARQL Across Federated Sources**

  The [Semantic Discovery System](http://esw.w3.org/topic/RdfAndSql) provides the functionality to rapidly build solutions for non-technical Users to create and execute Ad Hoc queries using the network Graph User Interface (SPARQL to SQL is auto generated). Integrates and interconnects ALL data silo types: providing a virtual Semantic Web interface to all RDBMS’ Web Services, Excel Spreadsheets, and any Hybrid File Systems.

  - Semantic Discovery Systems (Main Site)
  - Semantic Discovery Systems ("Oracle"/"Summary Web-site")

  **RDF/XML-based**

  - d3view - Browsable RDF/XML, [LinkedData](http://www.verse7.com) over HTTP, manual mapping (it)
  - RelationalOwl - RDF/XML dump, automatic mapping
  - DJRQ - RDF/XML dump, manual mapping
  - METAMorphoses - Produces RDF/XML through a template language that selects a subgraph. Includes server for publishing the RDF/XML as [LinkedData](http://www.verse7.com). Manual mapping
  - TripleX - is a small plug-in for Web applications, which reveals the semantic structures encoded in relational databases by making database content available as RDF, JSON or Linked Data.
  - Virtuoso Universal Server (in SPARQL-based section above) facilitates retrieval of RDF/XML-hosted data in RDF Linked Data form, including RDF/XML serialization.

  **Proprietary APIs and query languages**

  - DJRQ - Java library, provides access to relational data through the Java API, Sesame API, SPARQL, RDQL. Automatic, highly customizable mapping.
  - InDeXable - Algol query language, manual mapping (it)
  - B2O - proprietary query language, manual mapping
  - jena property tables - Jena API, manual mapping
  - Virtuoso Universal Server - (in SPARQL-based section above) uses SPARQL to implement its own *Declarative Metaschema Language* for exposing ODBC, or JDBC-accessible SQL data sources as URI Dereferencable *Linked RDF Instance Data*. RDF data may be retrieved as RDF/XML, JSON, &c.

  **Other Related Dynamic RDF->SQL Mappings**

  - FOPLRelationalModel - Models a target relational model as a set of objects which dynamically generate optimized SQL queries (intersections, unions, &c.) from Basic Triple Patterns

  **Benchmarking RDF-to-RDF Tools**

  - Linked data is found in [RdfsStarBench](http://esw.w3.org/topic/RdfAndSql)
  - Martin Svetlik, Ivan Jelinek: [Benchmarking RDF Production Tools](http://esw.w3.org/topic/RdfAndSql) Paper comparing the performance of relational database to RDF mapping tools (METAMorphoses, DJRQ, SquirrelRDF) with native RDF stores (Jena, Sesame)
Thank you!
Several Examples in this tutorial come from [http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/](http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/).


Slide 77: [http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/](http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/)


Example of a D2RQ map [http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/](http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/)