D2R MAP

Language Specification

Table of Content

Contact: Chris Bizer (chris@bizer.de)

Version of this document: V0.1.1 / 4.7.2003

1 Introduction

D2R is a declarative language to describe mappings between relational database schemas and OWL ontologies. The mappings can be used by a D2R processor to export data from a relational database into RDF.

Semantic Web is an extension of the current Web, in which data is given a well-defined meaning by representing it in RDF and linking it to commonly accepted ontologies. This semantic enrichment allows data to be shared, exchanged or integrated from different sources and enables applications to use data in different contexts.
Most formatted data today is stored in relational databases. To be able to use this data in a semantic context, it has to be mapped to RDF, the data format of the Semantic Web.
The data model behind RDF is a directed labelled graph, which consists of nodes and labelled directed arcs linking pairs of nodes. To export data from a RDBMS into RDF, the relational database model has to be mapped to the graph-based RDF data model.

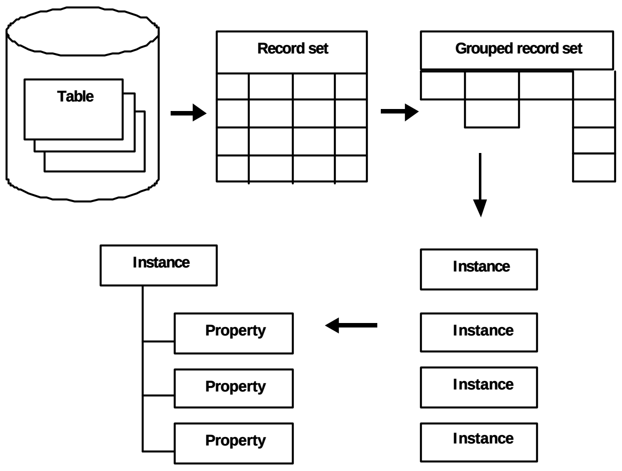
This document defines the syntax and semantics of the D2R MAP language.

Further information and usage examples of D2R MAP are found at http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rmap/D2Rmap.htm.

1.1 The D2R Mapping Process

The D2R mapping process executed by the D2R processor has four logical steps:

1. Selection of a record set from the database using SQL
2. Grouping of the record set by the d2r:groupBy columns.
3. Creation of class instances and identifier construction.
4. Mapping of the grouped record set data to instance properties.



For each class or group of similar classes a record set is selected from the database. Second, the record set is grouped according to the groupBy columns of the specific ClassMap. Then the class instances are created and assigned an URI or a blank node identifier. Finally, the instance properties are created using datatype and object property bridges.

The division between step three and four allows references to blank nodes within the model and to instances dynamically created in the mapping process.

1.2 The D2R MAP Processor

A D2R processor prototype is publicly available under GNU LGPL license. The processor is implemented in Java and is based on the Jena API. It exports data as RDF, N3, N-TRIPLES and Jena models. It is compliant with all relational databases offering JDBC or ODBC access. The processor can be used in a servlet environment to dynamically publish XHTML pages containing RDF, as a database connector in applications working with Jena models or as a command line tool.

2 Language Specification

A D2R Map is a well-formed XML document. D2R-defined elements are distinguished by belonging to the D2R namespace:
http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RMap/0.1#

2.1 Root Element: Map

Description:
Root element of the D2R-Map

Attributes:
xmlns, *optional*
D2R namespace declaration.
versionInfo, *optional*
Version Information.

Example:

```
<d2r:Map
   xmlns:d2r="http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RMap/0.1#"
   d2r:versionInfo=”$Id: TestMap.d2r, v 1.4 2002/07/31 19:44:09 Chris Exp
$”>

    <!— D2R Map -->

</d2r:Map>
```

2.2 Top-Level-Elements

2.2.1 DBConnection

Description:
Defines a ODBC or JDBC database connection.

Attributes:
odbcDSN, *optional*
    ODBC data source name.
jdbcDriver, *optional*
    JDBC driver.
jdbcDSN, *optional*
    JDBC data source name.
username, optional
    username, if required by the database.
password, *optional*
    password, if required by the database.

Example:

```
    <d2r:DBConnection d2r:odbcDSN=”E-ShopDB”
                      d2r:username=”admin”
                      d2r:password=”bla” />
```

2.2.2 Namespace

Description:
Maps a prefix to a specific namespace. The defined prefixes can be used in class and property mappings.

Attributes:
prefix, *required*
    Namespace Prefix
namespace, *required*
    Namespace URI

Example:

```
    <d2r:Namespace d2r:prefix=”eb”
         d2r:namespace=”http://www.example.org/eBusiness/1-2002#” />
```

2.2.3 Prepend / Postpend

Description:
Adds Text, XHTML, XML or RDF at the beginning or at the end of the output file.

Attributes:
fragment, *optional*
        Fragment of Text, XML or RDF.
href, *optional* *(Not implemented in V0.1)*
        URL of an external fragment to be imported.

Example:

```
    <d2r:Prepend>
       This text is added at the beginning of the document.
    </d2r:Prepend>
```

Hint: To prepend a XML declaration or other XML code, you have to use a CDATA section.

```
    <d2r:Prepend><![CDATA[
       <?xml version='1.0' encoding='ISO-8859-1'?>
    ]]></d2r:Prepend>
```

2.2.4 ProcessorMessage

Description:
Sends a message to the D2R processor. How the message is interpreted depends on the specific processor.

Attributes interpreted by D2R processor prototype V0.1:
saveAs, *optional*
        Path where to save the resulting output.
outputformat, *optional*
        Jena output format e.g. RDF/XML, RDF/XML-ABBREV, N-TRIPLE, N3

Example:

```
    <d2r:ProcessorMassage saveAs=”c:\test.rdf” outputformat=”N-TRIPLE”>
```

2.3 Mapping Elements

The content of the database is mapped to classes of the ontology using d2r:ClassMap, d2r:DatatypePropertyBridge and d2r:ObjectPropertyBridge elements.

Example:
The following ClassMap selects data about customers and their credit cards and maps the data to the eb:Customer class.

```
  <d2r:ClassMap
    d2r:type="eb:Customer"
```

```
     d2r:sql="SELECT Tab_K_Kunde.KundenNr, Tab_K_Kunde.FullName,
          Tab_K_Kunde.Anrede, Tab_K_Kunde_Kreditkarte.CCNo
          FROM Tab_K_Kunde_Kreditkarte INNER JOIN Tab_K_Kunde ON
          Tab_K_Kunde_Kreditkarte.KundenNr = Tab_K_Kunde.KundenNr;"
     d2r:groupBy="Tab_K_Kunde.KundenNr"
     d2r:uriPattern=
         “http://example.org/customers#@@Tab_K_Kunde.KundenNr@@“>

     <d2r:DatatypePropertyBridge
       d2r:property="per:Title" d2r:column="Tab_K_Kunde.Anrede"/>
     <d2r:DatatypePropertyBridge
       d2r:property="per:FullName" d2r:column="Tab_K_Kunde.FullName"/>

     <d2r:ObjectPropertyBridge d2r:property="per:CCard"
          d2r:referredClass="eb:CCard"
          d2r:referredGroupBy="Tab_K_Kunde_Kreditkarte.CCNo"/>
  </d2r:ClassMap>
```

2.3.1 ClassMap

ClassMaps are used to map the result of an SQL query to a class or to a group of similar classes. An example of a group of similar classes are different subclasses of a person class, all with similar properties, e.g. Student, Researcher, Professor, PhDStudent, Employee. ClassMaps are containers for property mappings.

Two attributes are required: An SQL statement to select data from the database and the groupBy attribute containing a columnlist to group the rows of the resultset.

Instances are identified by an URI or by a blank node identifier. URIs can be created with the d2r:uriColumn and d2r:uriPattern attributes. If no column or pattern is used, instances are identified as blank nodes.

Attributes:
id, *optional*
  ID of the ClassMap. Used to refer to the map with d2r:referredClass attributes. If a d2r:type is defined for the ClassMap, the type can also be used in references.
type, *optional*
  URI of an OWL Class.
sql, *required*
  SQL-Statement for selecting data from the database.
groupBy, *required*
  Column or column list used to group the rows of the result set. All rows with the same groupBy column values are used to create one instance.
uriColumn, *optional*
  Database column containing instance URIs.
uriPattern, *optional*
  Pattern to create instance URIs.

Example:

Table: Books

ISBN	Title	Keyword
1235647	Programming with C++	Programming
1235647	Programming with C++	C++

D2R Map Fragment:

```
  <d2r:ClassMap
    d2r:type="book:Book"
    d2r:sql="SELECT * from Books;"
    d2r:groupBy="book.ISBN"
    d2r:uriPattern="http://example.org/books#@@Books.ISBN@@">

    <d2r:DatatypePropertyBridge
      d2r:property="book:title" d2r:column="Books.Title"/>
    <d2r:DatatypePropertyBridge
      d2r:property="book:hasKeyword" d2r:column="Books.Keyword" />
  </d2r:ClassMap>
```

RDF Result:

```
<rdf:RDF>
  <book:Book rdf:about="http://example.org/books#1235647"
   book:title="Programming with C++">
          <book:hasKeyword>Programming</book:hasKeyword>
          <book:hasKeyword>C++</book:hasKeyword>
  </book:Book>
</rdf:RDF>
```

Example: Mapping to a group of simular OWL classes:

In order to map the result of an SQL query to a group of similar classes, use a d2r.ClassMap together with a d2r:ObjectPropertyBridge for creating the apropriate “rdf:type" property.

Table: Persons

PersonID	Name	Position
45	Chris Bizer	PhdStudent
46	Uwe Suhl	Full_Professor

D2R Map Fragment:

```
  <d2r:ClassMap
    d2r:sql="SELECT * from Persons;"
    d2r:groupBy="Persons.PersonID"
    d2r:uriPattern="http://www.fu-berlin.de#@@Persons.Name@@">

    <d2r:DatatypePropertyBridge
      d2r:property="iswc:name" d2r:column="Persons.Name"/>

    <d2r:ObjectPropertyBridge d2r:property="rdf:type"
       d2r:pattern="iswc:@@Persons.Position@@"/>

   </d2r:ClassMap>
```

RDF Result:

```
<rdf:RDF>

   <iswc:PhdStudent rdf:about="http://www.fu-berlin.de#Chris Bizer“
            <iswc:name>Chris Bizer</iswc:name>
   </iswc:PhdStudent>

   <iswc:Full_Professor rdf:about="http://www.fu-berlin.de#Uwe Suhl“
            <iswc:name>Uwe Suhl</iswc:name>
   </iswc:Full_Professor>

</rdf:RDF>
```

2.3.2 Property Mappings

Property mappings define bridges between columns of the result set and instance properties.

If a value in the result set is NULL, no property of the specific type is defined for that instance.

2.3.2.1 DatatypePropertyBridge

Description:
A DatatypePropertyBridge defines a bridge between a column of the result set and a literal property of the instances created.

Attributes:
property, *required*
  Qualified name of a datatype property.
column, *optional*
  Column name in the result set.
pattern, *optional*
  D2R pattern to create the property value.
value, *optional*
  d2r:value adds an additional property with a fixed value to all instances of the class.
translate, *optional*
  Id of a d2r:TranslationTable used to translate database to property values.
xml:lang, *optional*
  Language identifier, e.g. “de”, “en”.
datatype, *optional* *(not implemented in D2R processor VO.1)*
  XML datatype URI.
useCollection, *optional* *(not implemented in D2R processor VO.1)*
  Instructs the processor to use a collection for multiple values of a single property. Options are: rdf:Bag, rdf:Alt, rdf:Seq.

Example:

```
    <d2r:DatatypePropertyBridge d2r:property="per:FullName"
    d2r:column="Tab_K_Kunde.FullName" />
```

```
<d2r:DatatypePropertyBridge d2r:property="eb:Number"
d2r:pattern="Product no.: @@Tab_A-Artikel.ArtikelNummer@@"/>
```

2.3.2.2 ObjectPropertyBridge

Description:
ObjectPropertyBridge defines a bridge between a column of the result set and an object property of the instances created.

Attributes:
property, *required*
Qualified name of a datatype property.
column, *optional*
Column name in the result set.
pattern, *optional*
D2R pattern to create the property value.
value, *optional*
Adds an additional property with a fixed value to all instances of the class.
translate, *optional*
Id of a d2r:TranslationTable used to translate database to property values.
referredClass, *optional*
Reference to a d2r:ClassMap. The referredClass attribute is used to refer to dynamically created instances. The referred class is identified by it's URI used in a d2r:type attribute or it's d2r:id. A referredGroupBy attribute has to be used together with a referredClass attribute.
referredGroupBy, *optional*
Column or column list to identify the target instances of the referedClass. If more then one column is used to identify the referred instance, the order of the columns must match the column order in the referred class.
useCollection, *optional* *(not implemented in D2R processor VO.1)*
Instructs the processor to use a collection for multiple values of a single property. Options are: rdf:Bag, rdf:Alt, rdf:Seq.

Example:

```
<d2r:ObjectPropertyBridge d2r:property="per:Ccard"
d2r:referredClass="eb:CCard"
d2r:referredGroupBy="Tab_K_Kunde_Kreditkarte.CCNo"/>

<d2r:ObjectPropertyBridge d2r:property="eb:What"
d2r:pattern="http://www.mops.fu-berlin.de/shop/articles#@@Tab_A-
Artikel.ArtikelNummer@@"/>
```

2.4 Patterns

Patterns can be used to transform column values before using them as property values. Patterns can also be used to merge several column values to one property value.

A Pattern is a string including column names marked with @@. The column names are replaced with their values in the transformation process.

Example:

```
    d2r:pattern="http://www.mops.fu-berlin.de/shop/articles#@@Tab_A-
    Artikel.ArtikelNummer@@"

    d2r:pattern="http://www.mops.fu-berlin.de/shop/articles#@@Tab_A-
    Artikel.ArtikelNummer@@-@@Tab_A-Artikel.OfferDate@@"
```

2.5 Translation Tables

Translation tables can be used to substitute column values before they are used as property values.

TranslationTable
D2r:TranslationTable is a container for d2r:Translations.

Attributes:
id, *required*
ID of the translation table. The id is used by other elements to refer to the translation table.
href, *optional* *(not implemented in D2R processor VO.1)*
Location of a translation table, if it is stored in an external file.

Translation
A d2r:Translation defines the translation of key to a value.

Attributes:
key, *required*
Key to identify a translation.
value, *required*
Corresponding value.

Example:

```
      <d2r:TranslationTable d2r:id="Products2URI">
           <d2r:Translation d2r:key="12"
                   d2r:value="http://www.producer1.com/products/Prod1" />
           <d2r:Translation d2r: key ="1"
                   d2r: value ="http://www.company3.com/SuperProduct" />
                  <d2r:Translation d2r: key ="124"
                   d2r: value ="http://www.myCompany.de/products#124" />
      </d2r:TranslationTable>
```