R2RML: RDB to RDF Mapping Language

Editor's draft

This version:
$Id: Overview.html,v 1.17 2010/10/19 15:57:11 ssundara Exp $
Latest version:
http://www.w3.org/2001/sw/rdb2rdf/r2rml/Overview.html
Editors:
Souripriya Das, Oracle
Seema Sundara, Oracle
Richard Cyganiak, DERI, National University of Ireland, Galway
<richard@cyganiak.de>

Abstract

This specification defines the syntax and semantics of R2RML, a language for expressing customized mappings from relational databases to RDF datasets. Such mappings provide the ability to view existing relational data in the RDF data model, expressed in a structure and target vocabulary of the mapping author's choice. R2RML mappings are themselves RDF graphs and written down in Turtle syntax. R2RML enables different types of mapping implementations: delivering relational DB content as LinkedData, RDF dump or via a virtual SPARQL endpoint.

Comment [soa1]: I would omit this. First this is not important tob e mentioned in the abstract, secondly if ist RDF it can be serialized in any RDF serialization?!

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

The documents produced by this Working Group are:

- R2RML: RDB2RDF Mapping Language (this document)

This publication includes the RDF Schema that can be used to specify a mapping of relational data to RDF. The structure of this document will change based upon future decisions taken by the W3C RDB2RDF Working Group.

Comments on this document should be sent to public-rdb2rdf-comments@w3.org, a mailing list with a public archive.

This document was produced by the RDB2RDF Working Group, which is part of the W3C Semantic Web Activity.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

Appendices

---

1 Introduction

This specification defines the syntax and semantics of R2RML, a language for expressing customized mappings from relational databases to RDF datasets. Such mappings provide the ability to view existing relational data in the RDF data model, expressed in a structure and target vocabulary of the mapping author's choice.

A related specification is *A Direct Mapping of Relational Data to RDF* [*DIRECT*]. It defines a fixed “default mapping”. In the default mapping of a database, The structure of the resulting RDF graph directly reflects the structure of the database, the target RDF vocabulary directly reflects the names of database schema elements, and neither structure nor target vocabulary can be changed. With R2RML on the other hand, a mapping author can define highly customized views over the relational data.

The input to an R2RML mapping is a relational database. The output is an *RDF dataset* [*SPARQL*], as defined in SPARQL. The mapping is conceptual; implementations are free to materialize the output data, or to offer virtual access through an interface that queries the underlying database.

R2RML mappings are themselves expressed as RDF graphs and can be written down in *Turtle* [*TURTLE*] syntax.

The intended audience of this specification are implementors of software that generates or processes R2RML mapping documents, as well as mapping authors looking for a reference to the R2RML language constructs. The document uses concepts from *RDF Concepts and Abstract Syntax* [*RDF*] and from the *SQL language specifications* [*SQL1*][*SQL2*]. A reader's familiarity with the contents of these documents, as well as with the Turtle syntax, is assumed.

The R2RML language is designed to meet the use cases and requirements identified in *Use Cases and Requirements for Mapping Relational Databases to RDF* [*UCNR*].

Unless otherwise noted in the section heading, all sections and appendices in this document are normative.

@@Issue: The use of a "convention over configuration" approach has been proposed, where a mapping is not expressed completely, but rather as a delta from the default mapping. In other words, only those parts that one wants to be different from the default mapping have to be written down in the mapping file.

Comment [soa2]: I don't think we can/should force people to use a single RDF serialization syntax, which is furthermore not even standardized (yet).

Comment [soa3]: In this section we use default mapping in two different ways in the first paragraph a direct mapping oft he DB schema is meant, which fromm y POV can not be easily and incrementally refined into a default mapping, since it focusses rather on the technical representation than the conceptual model. I would suggest to distinguish again between direct mapping and default mapping.

1.1 Terminology

The following terms are defined in *RDF Concepts and Abstract Syntax* [*RDF*] and used in R2RML:

- RDF graph
- IRI (corresponds to the Concepts and Abstract Syntax term *RDF URI reference*)
- literal
- plain literal
- typed literal
- language tag
- lexical form
- datatype IRI (corresponds to the Concepts and Abstract Syntax term *datatype URI*)
- blank node

The following terms are defined in *SPARQL Query Language for RDF* [*SPARQL*] and used in R2RML:

- named graph
- RDF dataset

1.2 RDF-based Turtle Syntax; Media Type

An *R2RML mapping* defines a mapping from a relational database to an RDF dataset. This mapping itself is represented as an RDF graph. In other words, RDF is used not just as the target

data model of the mapping, but also as a formalism for representing the R2RML mapping itself. An RDF graph that represents an R2RML mapping is called a *mapping graph*.

An *R2RML mapping document* is any document written in ~~the Turtle [*TURTLE*]~~some RDF serialization~~syntax~~ that encodes an R2RML mapping graph. All examples throughout this specification use the Turtle syntax.

For R2RML mapping files, the same media type as for Turtle should be used: `text/turtle`. The preferred file extension is `.ttl`.

@@Issue: The working group is considering alternative syntaxes, such as XML or a custom SPARQL-inspired syntax, in place of or in addition to Turtle, as the snytax for R2RML mapping documents, and seeks feedback on this issue.

1.3 SQL Conformance

R2RML mappings contain various database objects, such as table names, column names, and SQL queries. These references form an important part of the mapping. These database concepts are defined in *ISO/IEC 9075-1:2008* [SQL1], and their syntax in *ISO/IEC 9075-2:2008* [*SQL2*].

In places where SQL expressions are allowed, these *MUST* conform to Core SQL 2008.

@@Issue: Implementors should be free to offer support for additional SQL dialects. How to address this?

Comment [soa4]: Replace MUST by SHOULD in the sentence above?!

1.4 Execution Environment and Database Connection

@@ Explain that an R2RML processor provides an execution environment, which provides the database that is the input to the mapping.

@@Issue: For convenience reasons, connection details could be provided in the mapping file as well, rather than configuring the connection externally.

@@Issue: The execution environment should probably provide other (optional) information, such as a base URI and default graph name.

1.5 Document Conventions

In this document, examples assume the following namespace prefix bindings unless otherwise stated:

Prefix	IRI
`rr:`	`http://www.w3.org/ns/r2rml#`
`rdf:`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
`rdfs:`	`http://www.w3.org/2000/01/rdf-schema#`

```
xsd:  http://www.w3.org/2001/XMLSchema#
xyz:  http://example.com/ns#
```

@@ Use `ex:` instead of `xyz:` throughout?

2 Overview

The overall flow for mapping relational data to RDF may be summarized as follows:

Given read-only access to a set of tables and/or views in a database, one can write a mapping of the content of some or all of these tables or views or any logical tables formed as SQL queries involving some of these tables and/or views to generate corresponding RDF triples. Besides generating triples for instance data items, the mapping may also create an RDF~~S~~ schema consisting of ~~RDFS~~ classes ~~and~~, ~~RDF~~ properties and any other schema elements, which can be expressed in RDF. ~~and~~ RDF named graphs may be created for holding subsets of the generated triples.

Note that the RDF triples ~~may~~ do not ~~actually~~ have to be ~~physically generated~~materialized and ~~instead~~ the mapping may only allow the relational data to be viewed as RDF triples.

The RDB2RDF Mapping Language, R2RML, described in this document, allows one to write a mapping to (virtually) generate a custom collection RDF graphs, named or unnamed, containing RDF triples, from a set of relational tables ~~and/or views~~ to which the user has ~~(at least)~~ read ~~only~~ access.

@@ Proposed text from the call: “A row in a relational table has <a row identifier> and a number of <column-name, value> pairs. Such a row may be translated to RDF as follows: subject := MAP(<row identifier>) and <predicate, object> := <MAP(column-name), MAP(value)>, for each <column-name, value> pair. An RDF triple is generated by combining the above as follows: <subject, predicate, object>.”

@@ More proposed text: “The domain of the mapping is XXX; the range is triples”

@@Issue: The sections up to here probably would benefit from some re-arranging. Ideas?

3 RDF Schema for R2RML using Turtle Syntax

@@ Requests from chair and team contact:

- Provide smaller/simpler, inline examples
- Make the document easier to digest (overall structure, grouping of functionality)
- Note which parts are under discussion (use 'Editorial note' such as found in the UC document [2]) - this fosters feedback from a wider community

Comment [soa5]: I would just refer to tables and add a small note, that this includes materialized/logical views

Comment [soa6]: I think we should not limit R2RML to RDF schema – some people might want to generate OWL, SKOS vocabularies, rules or whatever.

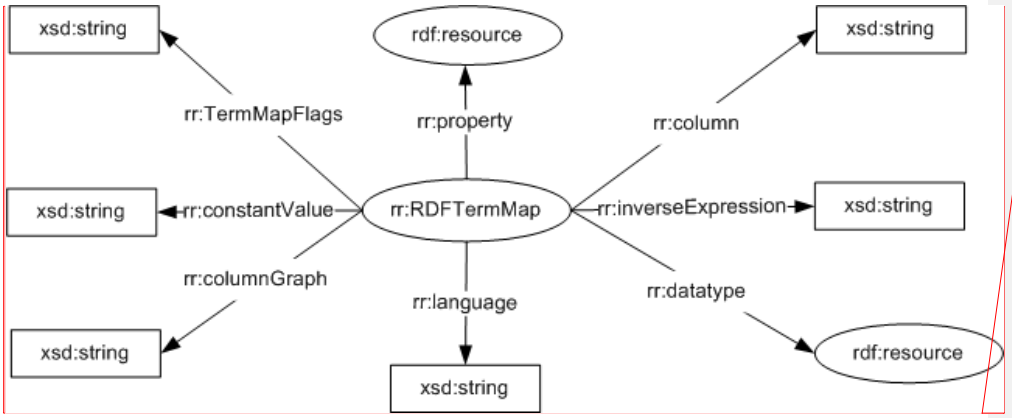
- If possible, explain what string literals have "special" meaning (i.e. SQL column, column name in database) etc. outside of just being normal strings. Explain that if those are not accepted, an error will result

This section describes the RDF Schema that can be used for specifying a mapping from a relational schema to RDF.

3.1 RDFTermMap Class

The RDFTermMap class represents the description of mapping from the values in a column of a relational table to a collection of RDF (property, object) pairs, where the property name is specified by the user and the objects are obtained from the values in the specified column. The object can be any RDF term namely IRIs, blank nodes, or literals. This mapping has two main components: a user-specified RDF property and the mapping of a value to an object (to be associated with the property).

Comment [soa7]: Describes the



Comment [soa8]: This is a very confusing illustration – it looks like the visualization of an RDF graph, but it is not. I would suggest to either depict an RDF graph here or use a table to list the properties (and their ranges), which have RDFTermMap as their domain

Figure 1: An RDF Graph describing the RDFTermMap class and its properties

3.1.1 Properties of the RDFTermMap Class

This section defines the RDF properties that have the RDFTermMap class as their domain.

3.1.1.1 rr:property

```
rr:property rdfs:range rdf:Resource .
```

This property specifies the RDF property component of the RDFTermMap instance. The following example shows the property component of an RDFTermMap instance for the DEPT table.

```
[] rr:property dept:name .
```

3.1.1.2 rr:column

```
rr:column rdfs:range xsd:string .
```

This property specifies the object value component of the RDFTermMap instance. Specifically, the column name of the ~~logical~~ table that will be used for obtaining the object value. In the following example, the dname column of the logical table defined using the DEPT table is specified as the source for the object component of an RDFTermMap instance and using rr:property, these objects are associated with the propery dept:name.

Comment [soa9]: This just confuses – just use table and explain once that all kinds of views are included

```
[] rr:property dept:name;
   rr:column "dname" .
```

Using the sample data for the DEPT table, the value "APPSERVER" in the dname column of the DEPT table, is mapped to the RDF (property, object) pair (dept:name, "APPSERVER").

3.1.1.3 rr:inverseExpression

```
rr:inverseExpression rdfs:range xsd:string .
```

This optional property specifies an expression that allows, at query processing time, use of indexes on any (underlying) relational table when accessing based on a value of a column (defined as an expression) in the logical table.

For example, for the deptId column in the logical table shown in Section 3.3.1.1, the inverse expression could be defined as follows:

```
"{alias.}deptno = substr({alias.}deptId,length('Department')+1)"
```

~~to~~ This facilitates the use of an index on the deptno column of the dept table. Note, that the actual alias (say d) used for the dept table is used to expand {alias.} (to d.).

3.1.1.4 rr:datatype

```
rr:datatype rdfs:range rdf:Resource .
```

This optional property specifies the datatype for the object value component of the RDFTermMap instance. If not specified, the datatype for the object value will be derived from the column definition of the logical table. In the following example, the datatype of the object value component of an RDFTermMap instance is specified to be a positive integer.

Comment [soa10]: We should add information how this is done.

```
[] rr:datatype xsd:positiveInteger .
```

3.1.1.5 rr:language

```
rr:language rdfs:range xsd:string .
```

This optional property specifies the language for the object value component of the RDFTermMap instance. This property is applicable only if the datatype of the object value is RDF plain literal (i.e. rr:datatype is omitted and the column datatype is mapped to `rdf:PlainLiteral` or `rr:datatype` explicitely has the value `rdf:PlainLiteral`). In the following example, the language tag for the object value component is specified to be US English.

```
[] rr:language "en-us" .
```

Comment [soa11]: When is this the case?

3.1.1.6 rr:columnGraph

```
rr:columnGraph rdfs:range xsd:string .
```

This property specifies the RDF named graph for the triples that are constructed with the (property, object) pair of the RDFTermMap. The value for this property is a string. If the flag ColumnGraphIRI is set (as part of rr:TermMapFlags value) then the string value represents an IRI, otherwise it represents the name of a column in the logical table. In the following example, the RDF named graph to be used for storing the triple generated using the (property, object) pair (emp:name, "ename") corresponding to an RDFTermMap is specified by the IRI "emp:empNameGraph".

Comment [soa12]: Why can't the value be a resource i.e. the GraphIRI itself? Than the cofiguration via flag would not be required.

```
[] rr:property emp:name;
   rr:column "ename";
   rr:columnGraph "emp:empNameGraph";
   rr:TermMapFlags "ColumnGraphIRI"
.
```

Given the above mapping and the sample data in the EMP table, the emp:empNameGraph holds the triple containing the following (predicate, object) pair

```
      (emp:name, "SMITH")
```

Comment [soa13]: I think for a reader ist very confusing that no triples seem tob e generated, but object value pairs. I think we should only refer to triples.

3.1.1.7 rr:constantValue

```
rr:constantValue rdfs:range ~~xsd:string~~rdfs:resource .
```

This property specifies the optional constant value for the object component of the RDFTermMap instance. The value can be any RDF term (i.e., IRI, blank node or a literal). In the following example, for the dept:COMPANY property of an RDFTermMap instance, the object value is defined as a constant "XYZ Corporation".

Comment [soa14]: Thatswhy range should be resource

```
[] rr:property dept:COMPANY;
   rr:constantValue "XYZ Corporation"
.
```

3.1.1.8 rr:TermMapFlags

```
rr:TermMapFlags rdfs:range xsd:string .
```

This property specifies settings for one or more flags (separated by whitespace) to indicate additional information about the values of some of the properties of an RDFTermMap instance.

The possible values for the flags are shown below:

- RDFTypeProperty

  This value indicates that the object value component of the RDFTermMap instance is also a value for the rdf:type property.

  ```
  [] rr:property emp:emptype;
     rr:column "empTypeURI";
     rr:TermMapFlags "RDFTypeProperty"
  .
  ```

- ComputedProperty

  This value indicates that the property component of the RDFTermMap instance is obtained from the values of a column in the logical table. Note that this requires that the TriplesMap instance has a computedPropertyMap with the same property name.

- ColumnGraphIRI

  This value indicates that string specified for the columnGraph property represents an IRI. See columnGraph for an example of its use.

Comment [soa15]: I think this is not required and just confuses, you can simply use [] rr:column rdf:type

Comment [soa16]: Why not allowing to give either an IRI or an string containing an SQL expression as range for rr:property? The we could omit this flag.

Comment [soa17]: This could be omited as per my comment above, overall the whole TermMapFlags could be omited thus streamlining and simplifying the standard.

3.2 Subclasses of the RDFTermMap Class

This section ~~enumerates~~ describes the subclasses of the RDFTermMap class. These classes ~~describe~~ comprise ~~the various types of~~specialized RDF term mappings~~s~~ namely to IRIs, blank nodes, and literals.

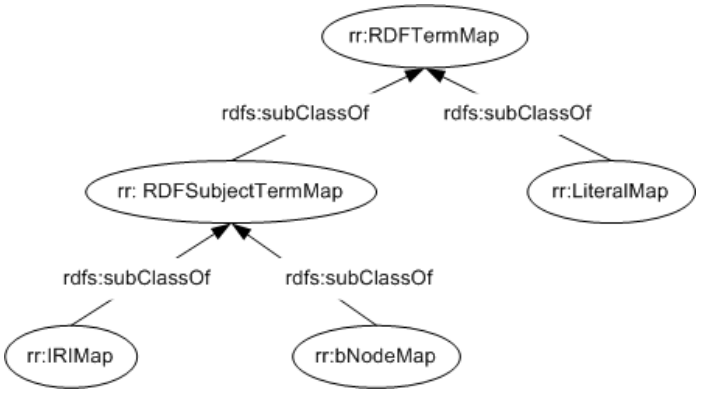


Figure 2: An RDF Graph describing the subclasses of the RDFTermMap class

3.2.1 RDFSubjectTermMap

The RDFSubjectTermMap class represents the mapping to any RDF term that can appear as subject of an RDF triple: IRIs or blank nodes.

```
rr:RDFSubjectTermMap rdfs:subClassOf rr:RDFTermMap .
```

3.2.2 IRIMap

The IRIMap class represents the mapping to a RDF Resource.

```
rr:IRIMap rdfs:subClassOf rr:RDFSubjectTermMap .
```

3.2.3 bNodeMap

The bNodeMap class represents the mapping to a RDF blank node.

```
rr:bNodeMap rdfs:subClassOf rr:RDFSubjectTermMap .
```

3.2.4 LiteralMap

The LiteralMap class represents the mapping to a RDF literal.

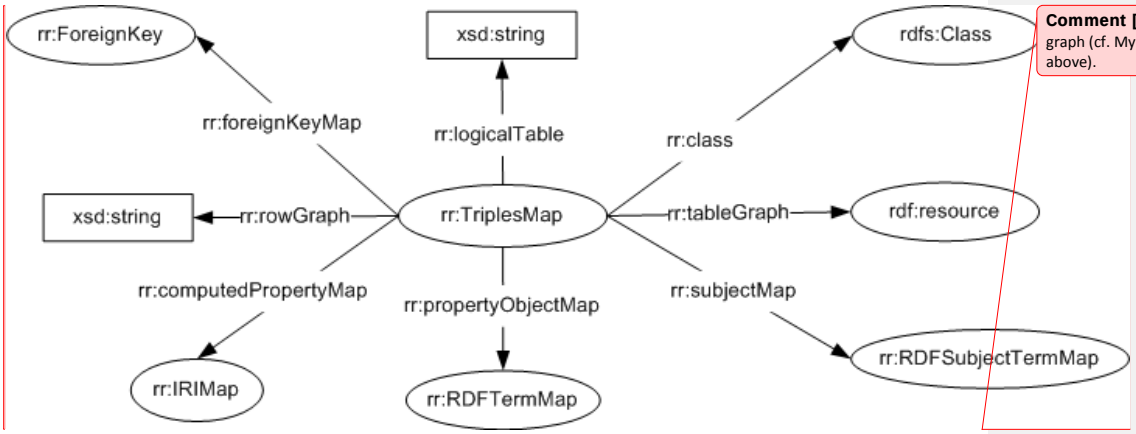
```
rr:LiteralMap rdfs:subClassOf rr:RDFTermMap .
```

3.3 The TriplesMap Class

TriplesMap is an RDFS class that allows the specification of a mapping of the rows in a ~~logical~~ table~~, represented by a SQL query, or the name of a table or view and its owner,~~ into RDF triples. Properties of a TriplesMap instance allow~~s~~ the specification of various aspects of the

Comment [soa18]: This is a very confusing name, since it has almost nothing to do with an RDF subject. I would rather call it RDFIRIorBnodeTermMap. Is this really required, are there cases, where we have to mapp to Bnodes and IRIs at the same time?

mapping such as, mapping of (some of) the columns in the logical table to RDF (property, object) pairs using RDFTermMap, name of the RDF graph(s) that would store the RDF triples, and so on.



Comment [soa19]: This is not an RDF graph (cf. My comment tot he figure above).

Figure 3: An RDF Graph describing the TriplesMap class and its properties

3.3.1 Properties of the TriplesMap Class

This section defines the RDF properties that have the TriplesMap class as their domain.

3.3.1.1 rr:logicalTable

```
rr:logicalTable rdfs:range xsd:String .
```

This property specifies the logical table (i.e., a SQL query or a table or view name plus its owner name) whose rows are mapped to RDF triples by this TriplesMap instance. Note that each row in the logical table is ~~identified as a~~mapped to a resource~~ and~~, which is used as the subject for the generated RDF triples describing the resource.

The following example ~~specifies~~specifies that the TriplesMap instance TriplesMap1 maps the rows in the logical table defined by the SQL ~~string~~ query to RDF triples.

```
<#TriplesMap1> rr:logicalTable
                                """Select 'Department' || deptno AS deptId
                                       , deptno
                                       , dname
                                       , loc
                                  from dept""" .
```

The cardinality of this property, for a TriplesMap instance, must be exactly 1. That is, a TriplesMap instance must have exactly one value for this property.

3.3.1.2 rr:subjectMap

```
rr:subjectMap  rdfs:range rr:RDFSubjectTermMap .
```

This property specifies the mapping to obtain the IRI or blank node that is used as the subject of *all* the RDF triples generated from *one* row of the table. In the following example, an IRIMap instance has been defined as the value for the subjectMap property of a TriplesMap instance for the logical table defined using the EMP table.

```
<#TriplesMap2> rr:subjectMap [ a rr:IRIMap; rr:column "empURI" ]
```

Using the above mapping definition, the obtained IRI for the single row in the sample data in the EMP table is

```
<xyz.com/emp/7369>
```

3.3.1.3 rr:propertyObjectMap

```
rr:propertyObjectMap rdfs:range rr:RDFTermMap .
```

This property specifies the mapping to obtain the (property, object) pair for each RDF triple corresponding to a column and its values in the table. The following example shows a propertyObjectMap of a TriplesMap instance for the logical table defined using the DEPT table.

```
<#TriplesMap1> rr:propertyObjectMap [ rr:property dept:name; rr:column
"dname" ] .
```

Using the above mapping and the sample data in the DEPT table, the value "APPSERVER" in the dname column of the logical table will be mapped to the following (predicate, object) pair.

```
(dept:name, "APPSERVER")
```

3.3.1.4 rr:class

Comment [soa20]: We should note, that this is just syntactic sugar for an IRIMap on property rdf:type with a constantValue containing the class.

```
rr:class rdfs:range rdfs:Class .
```

This property specifies the RDFS class associated with a certain TriplesMap ~~instance~~. In the generated RDF data, the resource corresponding to ~~each~~ a row in the logical table ~~is~~ will become an instance of this RDFS class.

In the following example, TriplesMap1 is associated with the the RDFS class xyz:dept using the rr:class property. This leads to the creation of xyz:dept as an RDFS class in the RDFS schema generated from the mapping specification and in the generated RDF data, RDF resources ~~corresponding to~~being created for each row in the logical table associated with TriplesMap1 ~~is an~~become instances of the xyz:dept class.

```
<#TriplesMap1> rr:class xyz:dept .
```

Given the above mapping and the sample data in the DEPT table, the following triple will be generated

```
_:Department10 rdf:type xyz:dept .
```

3.3.1.5 rr:tableGraph

Comment [soa21]: This name is confusing, since it has not much to do with a table. Maybe a better name is belongsToGraph

```
rr:tableGraph rdfs:range rdf:Resource .
```

This property specifies the graph IRI that would contain all the RDF triples in a TriplesMap instance.

Given the following example mapping,

```
<#TriplesMap1> rr:tableGraph xyz:DeptGraph .
```

all the RDF triples in TriplesMap1 would be stored in the RDF named graph

```
xyz:DeptGraph
```

3.3.1.6 rr:rowGraph

Comment [soa22]: Better name would be generateGraphURIfromColumn

```
rr:rowGraph rdfs:range xsd:string .
```

This optional property specifies the name of the column in the logical table that contains the RDF named graph for all the RDF triples generated from one row in the table. The minimum cardinality for this property is zero and no restriction on maximum cardinality. If for a row, the value in the specified column is NULL, then the triples generated from that row go to an unnamed graph. In the following example, all the triples generated from a row of the logical table defined using the EMP table are stored in the RDF named graph identified by the IRI from the "graphURI" column of the logical table.

```
<#TriplesMap2> rr:rowGraph "graphURI" .
```

Given the above mapping and the sample data in the EMP table, the rows generated are stored in the RDF named graph

```
<xyz.com/graph/CLERK/PART_TIME>
```

3.3.1.7 rr:computedPropertyMap

Comment [soa23]: Shouldn't the domain of this property be rather RDFTermMap? I think this would simplify things substantially!

```
rr:computedPropertyMap rdfs:range rr:IRIMap .
```

This optional property specifies ~~the mapping~~ to obtain the property IRIs from a column in the table. This mapping also contains a property name that identifies this computedPropertyMap and this property name is used in a rr:propertyObjectMap to form the association. The property IRI generated (from the column identified in the rr:computedPropertyMap) for each row, along with the value of the column identified in the rr:propertyObjectMap, are used to create the (property,

object) pairs. Consider the following example based on the logical table defined using the LIKES table.

```
<#TriplesMap3>
    rr:computedPropertyMap [ rr:property likes:likeType; rr:column "empLikes"
];
    rr:propertyObjectMap [ rr:property likes:likeType; rr:column "likedObj";
rr:TermMapFlags "ComputedProperty" ];
```

Here the property IRIs and the corresponding object values are obtained from the "empLikes" and "likedObj" columns, respectively, of the same row of the logical table. Note that the likes:likeType IRI has been used to associate the "likeType" column with the "likedObj" column.

Given the above mapping and the sample data in the LIKES tables, the generated (property, object) pairs are

```
      (<xyz.com/emp/likes/Playing>, "Soccer")
      (<xyz.com/emp/likes/Watching>, "basketball")
```

3.3.1.8 rr:foreignKeyMap

```
rr:foreignKeyMap rdfs:range rr:ForeignKey .
```

This property specifies the mapping to obtain the property, and a join condition that can be used to retrieve the object (from the parent Triples instance), for the generated RDF triple. This property would typically correspond to a foreign key definition in the table. The following example shows a foreignKeyMap for the logical table defined using the EMP table. See ForeignKey class for detailed explanation.

```
<#TriplesMap2> rr:foreignKeyMap [
     rr:key emp:c_ref_deptno;
     rr:parentTriplesMap xyz:dept;
     rr:joinCondition "{child.}deptno = {parent.}deptno";
   ];
```

3.4 The ForeignKey Class

ForeignKey is an RDFS class that allows specification of the mapping that describes a foreign key relationship.

```
rr:ForeignKey a rdfs:Class .
```

This class has three components: a key that specifies the column constraint in the child table, the TriplesMap corresponding to the parent table, and the join condition corresponding to the foreign key constraint. Conceptually, a ForeignKey instance is similar to a propertyObjectMap where the property name is identified by the key and a join condition is specified that can be used to obtain the object value from the TriplesMap corresponding to the parent (logical) table.

Comment [soa24]: What is meant by child and parent table – the refering and referenced tables?

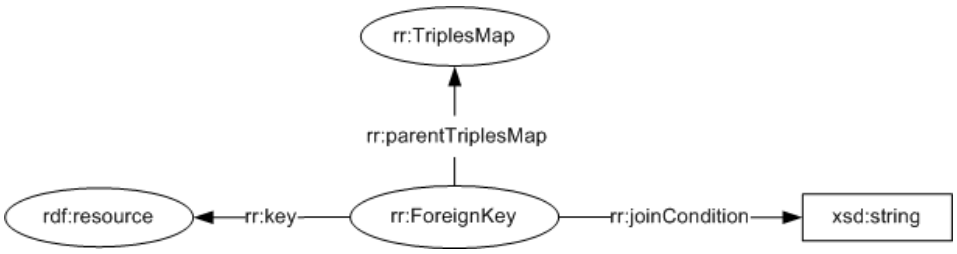


Figure 4: An RDF Graph describing the ForeignKey class and its properties

3.4.1 Properties of the ForeignKey Class

This section defines the RDF properties that have the ForeignKey class as their domain.

3.4.1.1 rr:key

```
rr:key rdfs:range rdf:Resource .
```

This property specifies the constraint component of the ForeignKey instance. The following example identifies the IRI emp:c_ref_deptno as a property that represents a foreign key constraint.

```
[] rr:key emp:c_ref_deptno .
```

3.4.1.2 rr:parentTriplesMap

```
rr:parentTriplesMap rdfs:range rr:TriplesMap .
```

This property specifies the TriplesMap corresponding to the parent table component, of the ForeignKey instance. The following example identifies the TriplesMap corresponding to the parent logical table.

```
[] rr:parentTriplesMap xyz:dept .
```

3.4.1.3 rr:joinCondition

```
rr:joinCondition rdfs:range xsd:string .
```

This property specifies the join condition of the ForeignKey instance. The following example shows a join condition to be used to obtain the object values from the parent logical table.

```
[] rr:joinCondition "{child.}deptno = {parent.}deptno" .
```

A. Example of SQL based RDB2RDF Mapping using the Turtle Syntax (Informative)

In this section we show an example relational schema and then map the example schema to RDF using the Turtle syntax.

A.1 Sample Relational Tables

We illustrate our example with the use of the following three relational tables, along with sample data.

A.1.1 DEPT Table

Column Name	Column Datatype	Column Key Constraints
deptno	NUMBER	UNIQUE
dname	VARCHAR2(30)	
loc	VARCHAR2(100)	

deptno	dname	loc
10	APPSERVER	NEW YORK

A.1.2 EMP Table

Column Name	Column Datatype	Column Key Constraint
empno	NUMBER	PRIMARY KEY
ename	VARCHAR2(100)	
job	VARCHAR2(30)	
deptno	NUMBER	REFERENCES DEPT(deptno)
etype	VARCHAR2(30)	

empno	ename	job	deptno	etype
7369	SMITH	CLERK	10	PART_TIME

A.1.3 LIKES Table

Column Name	Column Datatype	Column Key Constraint
id	VARCHAR2(4000)	
likeType	VARCHAR2(30)	
likedObj	VARCHAR2(100)	

id	likeType	likedObj
7369	Playing	Soccer
7369	Watching	Basketball

A.2 Mapping Specification for the Tables

The tables specified in the above section are mapped to the RDF using the mapping specificied in this document. The table below the TriplesMap shows the insatnce-level RDF triples generated from the sample data in the tables. In addition to the prefix conventions specified in Section 1.4 Document Conventions
, we also use the following prefixes in our example:

Prefix	IRI
emp:	http://www.example.com/emp#
dept:	http://www.example.com/dept#
likes:	http://www.example.com/likes#

A.2.1 Mapping Specification for the DEPT Table

```
<#TriplesMap1>
    a rr:TriplesMap;
    rr:logicalTable """
       Select ('Department' || deptno) AS deptId
            , deptno
            , dname
            , loc
         from dept
      """;
    rr:class xyz:dept;
    rr:tableGraph xyz:DeptGraph;
    rr:subjectMap [ a rr:bNodeMap; rr:column "deptId";
                    rr:InverseExpression "(alias.)deptno =
substr({alias.}deptId,length('Department')+1)"];
    rr:propertyObjectMap [ rr:property dept:deptno; rr:column "deptno";
rr:datatype xsd:positiveInteger ];
    rr:propertyObjectMap [ rr:property dept:name; rr:column "dname" ];
    rr:propertyObjectMap [ rr:property dept:location; rr:column "loc" ];
    rr:propertyObjectMap [ rr:property dept:COMPANY; rr:constantValue "XYZ
Corporation" ];
.
```

In the table below, the graph name is generated using the rr:tableGraph property.

Graph	Subject	Predicate	Object
xyz:DeptGraph	_:Department10	rdf:type	xyz:dept

xyz:DeptGraph	_:Department10	dept:deptno	10^^xsd:positiveInteger
xyz:DeptGraph	_:Department10	dept:name	"APPSERVER"
xyz:DeptGraph	_:Department10	dept:location	"NEW YORK"
xyz:DeptGraph	_:Department10	dept:COMPANY	"XYZ Corporation"

A.2.2 Mapping Specification for the EMP Table

```
<#TriplesMap2>
    a rr:TriplesMap;
    rr:logicalTable """
       Select ('xyz.com/emp/' || empno) AS empURI
            , empno
            , ename
            , ('xyz.com/emp/job/'|| job) AS jobTypeURI
            , job
            , deptno
            , ('xyz.com/emp/etype/'|| etype) AS empTypeURI
            , etype
            , ('xyz.com/graph/'|| job || '/' || etype) AS graphURI
       from emp
       """;
    rr:class xyz:emp;
    rr:subjectMap [ a rr:IRIMap; rr:column "empURI" ];

    rr:rowGraph "graphURI";

    rr:propertyObjectMap [ rr:property emp:jobtype; rr:column "jobTypeURI";
rr:TermMapFlags "RDFTypeProperty" ];
    rr:propertyObjectMap [ rr:property emp:emptype; rr:column "empTypeURI";
rr:TermMapFlags "RDFTypeProperty" ];

    rr:propertyObjectMap [ rr:property emp:empno; rr:column "empno" ];
    rr:propertyObjectMap [ rr:property emp:name; rr:column "ename";
rr:columnGraph "emp:empNameGraph"; rr:TermMapFlags "ColumnGraphIRI" ];
    rr:propertyObjectMap [ rr:property emp:job; rr:column "job" ];
    rr:propertyObjectMap [ rr:property emp:deptNum; rr:column "deptno" ];
    rr:propertyObjectMap [ rr:property emp:etype; rr:column "etype" ];
    rr:foreignKeyMap [
      rr:key emp:c_ref_deptno;
      rr:parentTriplesMap xyz:dept;
      rr:joinCondition "{child.}deptno = {parent.}deptno";
    ];
    .
```

In the table below, the graph name is generated using the rr:rowGraph property.

Graph	Subject	Predicate	Object
xyz.com/graph/CLERK/PART_TIME	xyz.com/emp/7369	rdf:type	xyz:emp
xyz.com/graph/CLERK/PART_	xyz.com/emp/73	emp:jobty	xyz.com/emp/job/CLERK

TIME	69	pe	
xyz.com/graph/CLERK/PART_TIME	xyz.com/emp/7369	emp:emptype	xyz.com/emp/etype/PART_TIME
xyz.com/graph/CLERK/PART_TIME	xyz.com/emp/7369	emp:empno	"7369"^^xsd:decimal
xyz.com/graph/CLERK/PART_TIME	xyz.com/emp/7369	emp:name	"SMITH"
xyz.com/graph/CLERK/PART_TIME	xyz.com/emp/7369	emp:job	"CLERK"
xyz.com/graph/CLERK/PART_TIME	xyz.com/emp/7369	emp:deptNum	"10"^^xsd:decimal
xyz.com/graph/CLERK/PART_TIME	xyz.com/emp/7369	emp:etype	"PART_TIME"

In the table below, the graph name is generated using the rr:columnGraph property.

Graph	Subject	Predicate	Object
emp:empNameGraph	xyz.com/emp/7369	emp:name	"SMITH"

A.2.3 Mapping Specification for the LIKES Table

```
<#TriplesMap3>
    a rr:TriplesMap;
    rr:logicalTable """
      Select ('xyz.com/emp/' || id) AS empId
           , ('xyz.com/emp/likes/' || likeType) AS empLikes
           , likedObj
        from likes
      """;
    rr:tableGraph xyz:LikesGraph;
    rr:subjectMap [ a rr:IRIMap; rr:column "empId"];
    rr:computedPropertyMap [ rr:property likes:likeType; rr:column "empLikes"
];
    rr:propertyObjectMap [ rr:property likes:likeType; rr:column "likedObj";
rr:TermMapFlags "ComputedProperty" ];
    .
```

In the table below, the graph name is generated using the rr:tableGraph property.

Graph	Subject	Predicate	Object
xyz:LikesGraph	xyz.com/emp/7369	xyz.com/emp/likes/Playing	"Soccer"
xyz:LikesGraph	xyz.com/emp/7369	xyz.com/emp/likes/Watching	"Basketball"

B. References

B.1 Normative References

[RDF]
Resource Description Framework (RDF): Concepts and Abstract Syntax, Graham Klyne, Jermey J. Carroll, Editors. World Wide Web Consortium, 10 February 2004. This version is http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. The latest version is http://www.w3.org/TR/rdf-concepts/.

[SPARQL]
SPARQL Query Language for RDF, Eric Prud'hommeaux, Andy Seaborne, Editors. World Wide Web Consortium, 15 January 2008. This version is http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/. The latest version is http://www.w3.org/TR/rdf-sparql-query/.

[SQL1]
ISO/IEC 9075-1:2008 SQL – Part 1: Framework (SQL/Framework). International Organization for Standardization, 27 January 2009.

[SQL2]
ISO/IEC 9075-2:2008 SQL – Part 2: Foundation (SQL/Foundation). International Organization for Standardization, 27 January 2009.

[TURTLE]
Turtle - Terse RDF Triple Language, Dave Beckett, Tim Berners-Lee. World Wide Web Consortium, 14 January 2008. This version is http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/. The latest version is http://www.w3.org/TeamSubmission/turtle/.

B.2 Other References

[DIRECT]
A Direct Mapping of Relational Data to RDF, Eric Prud'hommeaux, Editor. World Wide Web Consortium, 2 June 2010. The latest version is http://www.w3.org/2001/sw/rdb2rdf/directGraph/. This document is work in progress.

[UCNR]
Use Cases and Requirements for Mapping Relational Databases to RDF, Eric Prud'hommeaux, Michael Hausenblas, Editors. World Wide Web Consortium, 8 June 2010. This version is http://www.w3.org/TR/2010/WD-rdb2rdf-ucr-20100608/. The latest version is http://www.w3.org/TR/rdb2rdf-ucr/. This document is work in progress.

C. Acknowledgements (Informative)

@@

D. CVS History (Informative)

```
$Log: Overview.html,v $
```

Revision 1.17 2010/10/19 15:57:11 ssundara
added more inline examples using the sample data -- Seema/Souri

Revision 1.16 2010/10/18 20:39:15 sdas2
explained and added examples for RDFTermMap -- Souri/Seema

Revision 1.15 2010/10/18 16:18:39 sdas2
added graph names to the examples in Appendix A.2 -- Souri/Seema

Revision 1.14 2010/10/18 15:24:47 sdas2
removed CDATA because its use made sections unreadable by browsers --
Souri/Seema

Revision 1.13 2010/10/12 22:35:29 rcygania2
Added section on Execution Environment and many @@todos

Revision 1.12 2010/10/12 21:52:11 rcygania2
Make it validate as XHTML 1.0

Revision 1.11 2010/10/12 21:24:41 rcygania2
Organized references, added References section, made some section informative

Revision 1.10 2010/10/12 20:02:04 rcygania2
extended intro; work on references; lots of layout/markup work

Revision 1.9 2010/10/12 16:05:08 rcygania2
typo

Revision 1.8 2010/10/12 16:04:33 rcygania2
add note on intended audience

Revision 1.7 2010/10/12 16:01:04 rcygania2
re-did section 1

Revision 1.6 2010/10/12 15:57:15 sdas2
added examples with data for the sample tables -- Souri/Seema

Revision 1.5 2010/10/12 14:47:03 sdas2
incorporating the figures, submitted by Boris Villazon Terrazas, illustrating
classes and properties of the mapping language -- Souri/Seema

Revision 1.4 2010/10/11 21:21:21 sdas2
added inline examples for all the properties -- Souri/Seema

Revision 1.3 2010/10/11 19:00:12 sdas2
simplified destination graph related properties and added TermMapFlags to
replace isRDFTypeProperty and isComputedProperty and allow generalized flags
-- Seema and Souri

Revision 1.2 2010/10/06 13:59:19 sdas2
renamed Table2TriplesMap to TriplesMap

Revision 1.1 2010/10/05 17:19:24 rcygania2
Add Seema's version of R2RML draft to CVS