Functional Queries to Wrapped Educational Semantic Web Meta-Data *

Tore Risch
Dept. of Information Technology, Uppsala University, Sweden
email: Tore.Risch@it.uu.se

2004

Abstract

The aim of the Edutella project is to provide a peer-to-peer infrastructure for educational material retrieval using semantic web meta-data descriptions of educational resources. Edutella uses the semantic web meta-data description languages RDF and RDF-Schema for describing web resources. The aim of this work is to wrap the Edutella infrastructure with a functional mediator system. This makes it possible to define general functional queries and views over educational and other material described using RDF and RDF-Schema. It is based on the observation that RDF-Schema definitions are easily mapped into a functional data model. This allows any RDF-Schema file to be wrapped by a functional mediator system. The RDF-Schema definitions are translated into a corresponding functional meta-data schema consisting of a type hierarchy and a set of functions. Queries then can be expressed in terms of the translated semantic functional schema. Since meta-data descriptions can contain both RDF and RDF-Schema definitions the system allows both to co-exist with the difference that queries over basic RDF meta-data are expressed on a lower level in terms of a generic functional schema representing all RDF structures.

1 Introduction

In most query languages the queries are expressed against *schema* that provides a meta-data description of the data to be queried. Not only does the schema provide for efficient data management, but, very importantly, it provides also a 'map' for the user of the structure of the data to be queried. A well known example of queries without schemas are free text searches in web search engines, e.g., where most users have been frustrated by the lack of guidance how to

*Published in P.Gray, L.Kerschberg, P.King, and A.Poulovassilis (eds.): *Functional Approach to Computing with Data*, Springer, ISBN 3-540-00375-4, 2004.

find the information relevant for a subject. By contrast, SQL-databases always have a detailed database schema for efficient access, documentation, and user guidance. The goal of this paper is to show how web search can be enhanced by means of a schema.

RDF (Resource Description Framework) [9] is an XML based notation for associating arbitrary properties with any web resource (URI). However, there is no real schema description in RDF since there are no restrictions on the kinds of properties a given class of web resources can have. In contrast, RDF-Schema [3] extends RDF with schema definition capabilities. For example, for a particular class of web resources an RDF-Schema provides a description of allowed properties along with an inheritance mechanism.

The observation of this work is that the semantic data model of RDF-Schema can be regarded as a functional data model as well. Given that observation, we show how RDF-Schema definitions can be mapped to typed functions. We then develop a wrapper for RDF-Schema definitions allowing a functional mediator engine, Amos II [14], to import RDF-Schema definitions. The imported RDF-Schema definitions are translated to the functional data model used in Amos II. The user can then specify high-level functional queries in the query language AmosQL [14] of data described by the translated schema.

A particular problem in the semantic web environment is that RDF and RDF-Schema descriptions can be freely mixed. Thus, some web resources for a subject may be described only though RDF while others have RDF-Schema descriptions. Furthermore, a class of resources described through RDF-Schema may have additional RDF properties not included in the schema. Therefore, it is important to be able to transparently query both RDF and RDF-Schema meta-data.

Edutella [11] provides a peer-to-peer standardized interface and infrastructure for searching and accessing educational material known to Edutella. Web sources described by RDF or RDF-Schema are accessed through Edutella and can be queried through a family of RDF-based query languages, *QELi*, of increasing expressibility. In this way the data and knowledge in educational sources provided through Edutella can be queried.

The system presented here, RDFAmos, can directly wrap any RDF- or RDF-Schema-based data source on the web. Its mediator functionality allows reconciliation of differences and similarities between different kinds of wrapped sources, e.g. different RDF-Schema views, RDF-standards such as Dublin Core [5], relational databases, etc. General functional queries of mediated views over different sources are then allowed [14].

RDFAmos provides mediation services for Edutella. This means that RDFAmos peers are made accessible through Edutella's JXTA-based [13] peer-to-peer (P2P) infrastructure. Through the infrastructure RDFAmos receives QEL3 queries for execution. The query result is delivered back to Edutella through the same infrastructure.

Edutella-based sources can be wrapped as well. A wrapper including a QEL3 query generator is being developed. It can generate QEL3 query expressions executed through Edutella's distributed P2P search facilities. This allows

mediating functional views to be defined that combine heterogeneous learning material from Edutella with data from other sources.

The next section presents an overview of the RDFAmos system, followed by a description of how it is being integrated into the Edutella P2P infrastructure. Two subsequent sections show how RDF-Schema and basic RDF meta-data descriptions are translated.

2 Architecture

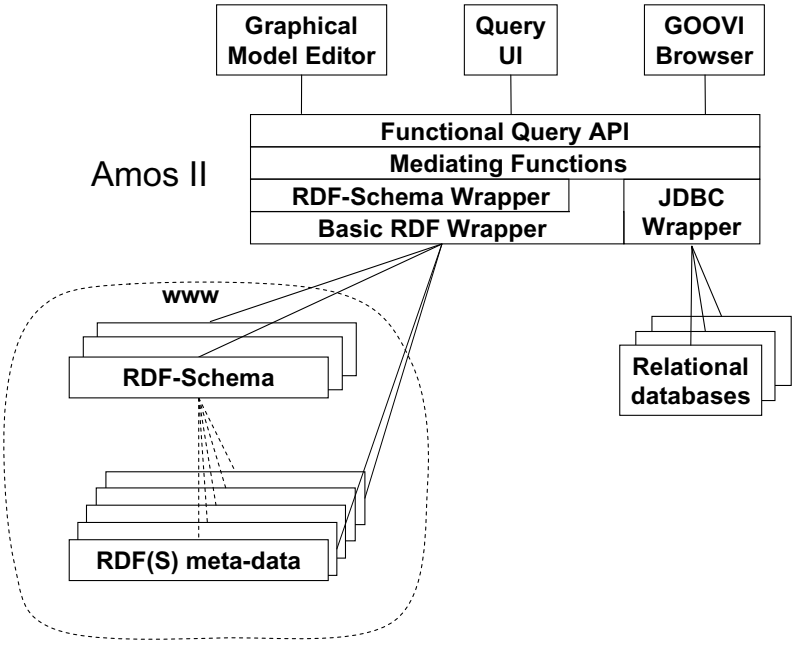

Figure 1: RDFAmos Architecture

Figure 1 illustrates the architecture of RDFAmos, and how it accesses various data sources, including RDF and RDF-Schema meta-data definitions describing web resources. The meta-data definitions of each source are imported to the mediator where they are represented using the Amos II functional data model. Users and applications can formulate queries over the imported meta-data in terms of a functional schema automatically created in the mediator from the imported meta-data definitions. In Fig. 1 there are examples of three kinds of applications: The *GOOVI Browser* is a general browser of functional Amos II databases [4], the *Query API* is a query executor for ad-hoc functional queries,

and the *Graphical Model Editor* is a system for editing meta-data structures and graphically specifying RDF-based queries [12].

The *Mediating Functions* of RDFAmos define functional views over combinations of data from different sources. Applications and users may specify arbitrary queries and views in terms of the mediated sources and combinations of them.

In Fig. 1 there are examples of three kinds of data sources: relational databases, RDF-Schema, and basic RDF web sources. For each kind of data source the mediator needs a *wrapper* which is a program module to import any schema definition for that kind of data source and translate it to functions and types in the functional data model. Once a particular schema is imported, the wrapper can translate queries in terms of the translated functional schema into accesses to the source. For example, relational databases can be accessed through JDBC and there is a *JDBC Wrapper* that imports relational schema descriptions through the standard JDBC relational database API and translates functional queries to SQL.

For RDF based sources the *Basic RDF Wrapper* allows import of any RDF meta-data description. All imported RDF objects are represented in the mediator using a simple functional schema, the *basic functional RDF representation*. It can represent any RDF-based meta-data. An RDF parser [2] translates RDF statements to corresponding binary relationships (triples) imported to the mediator. Once imported to the mediator, the RDF-structures can be analyzed through queries to the basic functional RDF representation. Such queries are relatively primitive with little user guidance on what properties are available for a resource, its classifications, etc.

On top of the basic RDF wrapper the *RDF-Schema* wrapper translates the RDF-Schema meta-data definitions expressed in RDF to a functional representation called *a functional RDFS view*. Since RDF-Schema provides schema information for RDF sources, unlike basic RDF, RDF-Schema definitions can be translated to data-source-specific function and type definitions in the mediator. Once a functional RDFS view for a web source has been imported by the wrapper it can be queried. Thus location-independent queries utilizing the full power of the semantic functional data model can be specified and executed. In particular web sources describing learning material can be imported and queried. The result of such a query is a set of web resources that can be browsed by the user using a web browser or GOOVI [4].

In contrast to basic RDF, RDF-Schema requires further processing to semantically enrich the basic functional RDF representation to include the data source specific functions (properties), types (classes), and inheritance. Therefore, after an RDF meta-data document is loaded using the basic RDF wrapper, the system goes through the loaded binary RDF relationships (triples) to find the RDF-Schema type, inheritance, and property definitions. From these definitions the corresponding meta-definitions in RDFAmos are automatically generated as a set of type and derived function definitions. The function definitions are defined in terms of the basic RDF binary relationships as views. In this way we maintain both the basic functional RDF representation of meta-data along side

with semantic views that access it, thus making it possible to query the data using different models with different semantic expressiveness.

One difference between relational database sources and web sources is that in RDF-Schema there is no clear distinction between schema and data definitions, and RDF-Schema and RDF definitions can be freely mixed. This requires a system that can dynamically extend the schema at run time.

The RDFAmos architecture allows functional queries and views over any web source described by RDF or RDFSchema alone, or combined with any other wrapped data source. Applications can be written in C, C++, Java, or Lisp. The applications do not have to know any details of the data sources, but need only see mediated data. For Java there is a 'bean' package [1] that generates transparent Java interface class definitions from the mediator schema. This allows navigating the mediated structures as regular Java classes, while at the same time being able to dynamically execute from Java functional queries over the mediated structures.

An RDFAmos peer can be also a data source for other RDFAmos peers [14]. There is thus a special wrapper for interfacing other RDFAmos peers. This mechanism allows modularization of mediators for scalable mediation [7, 8].

3 Mediating Edutella Peers

Figure 2 illustrates how RDAmos is interfaced with the Edutella infrastructure for P2P access to educational material. Edutella uses QEL3 as its query language. QEL3 is a Datalog-based query language for basic RDF data. RDFAmos is made available as a peer to Edutella by being able to execute queries in QEL3 (top of Figure). Our approach thus accepts basic RDF queries in QEL3 to RDFAmos which processed the query over mediated data sources.

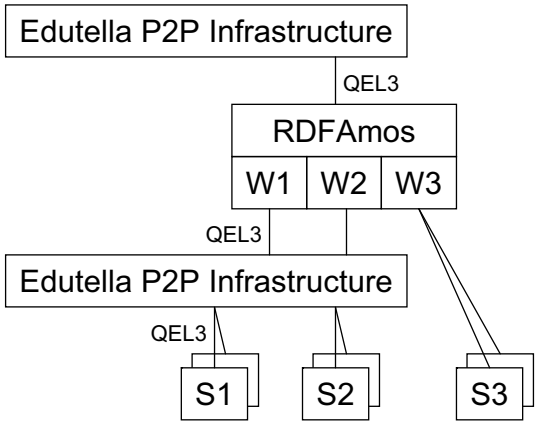

Figure 2: Edutella Interfaces

A special property of QELi is that the general structure of, e.g., QEL3 queries is described by a special RDF-Schema, the *query schema* for QEL3, which is imported to RDFAmos through the RDF-Schema wrapper as other schemas. Once the QEL3 schema is imported the system can import any QEL3 query as data producing a functional representation of the query itself in the mediator. Thus queries are regarded as yet another RDF-Schema data source. The functional representation of QEL3 query structures can be queried as other data.

In order to not only represent but also execute QEL3 queries, we have created a query building AmosQL function that produces an AmosQL query string from an imported QEL3 query description. The query generation itself is implemented as query functions concatenating query elements retrieved from the imported query structure[1]. The query string is then evaluated using a system function `eval` and the result is converted and delivered back to Edutella as a QEL3 result structure in RDF. This makes RDFAmos an Edutella peer capable of executing arbitrary QEL3 queries over its mediated data.

Some wrapped sources may be interfaced from mediators using the regular web-based wrapper interfaces of Fig. 1. In Fig. 2 wrapper *W3* illustrates this. In this way, e.g., relational databases and RDF-Schema web sources are mediated and made available to Edutella.

A wrapper is also being developed for treating Edutella as a QEL3 based source, as illustrated by *W1* in Fig. 2. This wrapper will allow mediation of data from Edutella based QEL3 compliant peers. Similar as is done for SQL [6] the wrapper will generate QEL3 query specifications in RDF to the wrapped Edutella source. After the query has been processed by some peers managed by Edutella, the result is delivered back to the mediator as RDF structures. Some of the processing peers might be other RDFAmos mediators.

One limitation of the current implementation is that it is based on using QEL3 as query language when interfacing with Edutella. QEL3 is equivalent to basic Datalog over RDF statement without recursion. Rules allow named and disjunctive queries. Even though QEL3 itself is described through RDF-Schema, the QEL3 queries range only over basic RDF statements. It supports object abstraction only through explicit predicates over the RDF-Schema type system. Thus the QEL3 queries must be specified over the basic functional RDF representation which has little semantics.

In order to fully utilize the functional data model, a next step is to define a QELf for functional queries and support these in RDFAmos. An Edutella based wrapper for RDFAmos data sources can then be defined, illustrated by *W2* in Fig. 2). In order to allow one mediator to access mediator definitions from other mediators through Edutella, the mediator definitions of RDFAmos should be exportable as RDF-Schema.

[1] AmosQL has string construction and concatenation functions.

4 Functional Wrapping of RDF-Schema Definitions

The RDF-Schema wrapper can import any RDF-Schema meta-data description document from the web. The RDF-Schema definitions are thereby translated into corresponding function and type definitions. Data described by the RDF-Schema definitions are translated into object instances.

Types in RDFAmos are equivalent to *classes* in RDF-Schema. Both RDFAmos and RDF-Schema use extent-subset semantics [14] for inheritance. Only stored functions [14] have correspondence in RDF-Schema and they correspond to *properties* there. Meta-objects (schema elements) in RDFAmos mediators, such as types and function, are first class and can be queried as any other objects, as in RDF-Schema.

For example, assume the following RDF-Schema definition of learning materials [11]:

```
<rdf:Description ID="Book">
 <rdf:type resource=
        "http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource=
        "http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<rdf:Description ID="Title">
 <rdf:type resource=
        "http://www.w3.org/2000/01/rdf-schema#Property"/>
 <rdfs:domain rdf:resource="#Book"/>
 <rdfs:range rdf:resource=
        "http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:Description>

<rdf:Description ID="AI_Book">
 <rdf:type
  resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource="#Book"/>
</rdf:Description>
```

The RDF-Schema definitions are imported into the mediator and automatically translated to the following type and function definitions, as illustrated graphically by Fig. 3.

```
create type Book;
create function title(Book b)->Literal as
       select getprop(b,"Title");
create type AI_Book under Book;
```

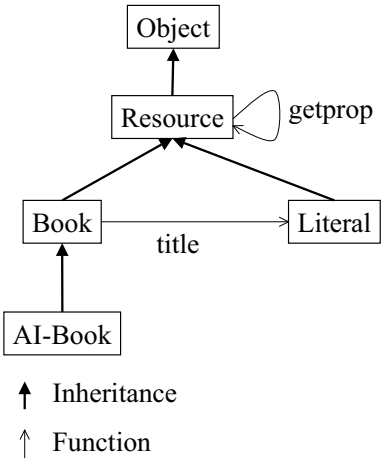

Figure 3: Functional Schema

Notice that short, readable type and function *nick names* are constructed from the URIs. For example, the nick name of `http://www.edutella.org/edutella#Title` is `Title`. The nick names allow readable query statements without having to specify full URIs as function and type names. The function `getprop` is a function accessing properties of objects stored using the basic functional RDF representation.

The following database instances satisfy the above RDF-schema definitions:

```
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:sch="http://user.it.uu.se/~torer/rdf/schema#"
  >

<sch:Book about="http://www.xyz.com/sw.html">
  <sch:Title>Software Engineering</sch:Title>
</sch:Book>

<sch:Book about="http://www.xyz.com/ai.html">
  <sch:Title>Artificial Intelligence</sch:Title>
</sch:Book>

<sch:AI_Book about="http://www.xyz.com/pl.html">
  <sch:Title>Prolog</sch:Title>
</sch:AI_Book>
```

```
</rdf:RDF>
```

Once the instances are imported to the mediator they can be queried in terms of functions and types using the functional query language AmosQL of RDFAmos.

The general syntax for AmosQL queries is:

```
select <result>
   from <domain specifications>
   where <condition>
```

For example,

```
select distinct X
   from Book X, AI_Book Y
   where title(X) = 'Artificial Intelligence' or
         X = Y;
```

The query searches for AI-books by looking for books where either the title is 'Artificial Intelligence' or the book is an instance of type `AI-book`. It is an example of a query to a functional RDFS view where the types `Book` and `AI-Book` are defined as RDF-Schema classes and the function `title` is a property. Each *domain specification* associates a query variable with a type where the variable is universally quantified over the extent of the type.

5 Basic Functional RDF Representation

The schema used in basic RDF descriptions (`http://www.w3.org/TR/rdf-mt/`) is rudimentary and provides little guidance at all for the user. In general one may associate freely chosen properties about any web resource, e.g.:

```
<edu:Book about="http://www.xyz.com/sw.html">
  <edu:Title>Software Engineering</edu:Title>
</edu:Book>
```

The above states that the Edutella registered book in `http://www.xyz.com/sw.html` has the title `Software Engineering`. In basic RDF the user can choose any property for the annotation (here `Title`), while RDF-Schema allows to restrict the allowable properties.

The basic functional RDF representation is described by the following functional schema:

```
create type Resource;
create function uri(Resource) -> String as stored;
create function rl(String u)  -> Resource r
       as select r
          where uri(r)=u;
```

```
create function name(Resource) -> String as stored;
create type Statement under Resource;
create function triple(Statement s) -> <Resource subject,
                                        Resource predicate,
                                        Resource object>
       as stored;
create function stmt(Resource s, Resource p, Resource o)
                     -> Boolean
       as select true
          from Statement st
          where triple(st)= <s,p,o>;
create function getprop(Resource r, String propname)
                     -> Resource v
        as select v
           from Resource p
           where stmt(r,p,v) and
                 name(p)=propname;
```

When RDF data is imported, the wrapper creates `Resource` and `Statement` objects and adds instances to the functions `uri` (resources) and `triple` (statements). The RDF statements always have an OID in the wrapper. In RDF only *reified statements* have URIs. By treating all statements as objects we are able to easily reason about statements and treat reified statements as a subtype of type `Statement`.

The function `getprop` is a convenience function for accessing properties of objects without having to specify long and complex URIs. Examples of queries:

```
getprop(uri("http://www.xyz.com/sw.html"),"Title")

Answer: "Artificial Intelligence"

select r
from Resource r
where "Artificial Intelligence" = getprop(r,"Title");

Answer: uri("http://www.xyz.com/sw.html")
```

Further abstractions can be made manually by defining functions accessing properties such as `title` above. Such property function are automatically generated by the wrapper for RDF-Schema properties.

It would be possible to automatically define the property functions from basic RDF definitions as the triples were asserted. However, this might generate very many functions and would not really guide the user further. The RDF-Schema provides a better solution by explicitly specifying allowable properties.

6 Summary

RDFAmos extends Amos II [14] by providing transparent functional mediation over RDF- and RDF-Schema-based meta-data descriptions of web sources. Wrappers are defined for other sources too, e.g. relational databases [6] or engineering systems [10]. Applications can access mediated data using an API based on functional queries. The RDF and RDF-Schema wrappers can import any RDF(-Schema) definition and make it available for functional queries. The data model of RDF-Schema is very similar to a subset of the functional data model used in Amos II which makes it particularly straight-forward to mediate RDF-Schema described web sources.

The Edutella P2P infrastructure allows queries over distributed educational material. RDFAmos can run as an Edutella peer thus providing general mediation services. In Edutella both schema, data, and queries are represented using a mixture of RDF and RDF-Schema. Since the approach makes no clear distinction between data, schema, and query definitions, the mediator engine treats all three kinds of data in a uniform way as wrapped data. Edutella queries in the Datalog based query language QEL3 are transformed by RDFAmos queries to functional query strings for evaluation.

This work shows that the functional data model is very well suited for managing and mediating RDF and RDF-Schema based data.

References

[1] M.Bendtsen and M.Björknert: Transparent Java Access to Mediated Database Objects, *http://user.it.uu.se/∼udbl/publ/ace.pdf*, 2001

[2] J.Carroll: ARP: Another RDF Parser, *http://www-uk.hpl.hp.com/people/jjc/arp/*, 2001.

[3] D.Brickley, R.V.Guha: RDF Vocabulary Description Language 1.0: RDF-Schema, WC3 Working Draft, *http://www.w3.org/TR/rdf-schema*, 2003.

[4] K.Cassel and T.Risch: An Object-Oriented Multi-Mediator Browser. *2nd International Workshop on User Interfaces to Data Intensive Systems*, Zürich, Switzerland, May 31 - June 1, 2001

[5] Dublin Core Metadata Initiative, *http://dublincore.org/*, 2003.

[6] G. Fahl, T. Risch: Query Processing over Object Views of Relational Data. *The VLDB Journal*, Springer, 6(4), 261-281, 1997.

[7] V.Josifovski, T.Katchaounov, T.Risch: Optimizing Queries in Distributed and Composable Mediators. *4th Conference on Cooperative Information Systems*, CoopIS'99, 291-302, 1999.

[8] T.Katchaounov, V.Josifovski, and T.Risch: Scalable View Expansion in a Peer Mediator System, *Proc. 8th International Conference on Database Systems for Advanced Applications (DASFAA 2003)*, Kyoto, Japan, March 2003.

[9] G.Klyne and J.J.Carroll: Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Working Draft, *http://www.w3.org/TR/rdf-concepts/*, 2003

[10] M.Koparanova and T.Risch: Completing CAD Data Queries for Visualization, *International Database Engineering and Applications Symposium (IDEAS 2002)*, Edmonton, Alberta, Canada, July 17-19, 2002.

[11] W.Neidl, B.Wolf, C.Qu, S.Decker, M.Sinek, A.Naeve, M.Nilsson, M.Palmèr, and T.Risch: EDUTELLA: A P2P Networking Infrastructure Based on RDF. *11th International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, USA, 2002.

[12] M.Nilsson, M.Palmèr, A.Naeve: Semantic Web Meta-data for e-Learning - Some Architectural Guidelines, *11th World Wide Web Conference (WWW2002)*, Hawaii, USA, 2002.

[13] S.Oaks, B.Traversat, and L.Gong: *JXTA in a Nutshell*, ISBN 0-596-00236-X, O'Reilley, 2002.

[14] T.Risch, V.Josifovski, T.Katchaounov: Functional Data Integration in a Distributed Mediator System, in P.Gray, L.Kerschberg, P.King, and A.Poulovassilis (eds.): *Functional Approach to Computing with Data*, Springer, ISBN 3-540-00375-4, http://user.it.uu.se/~torer/publ/FuncMedPaper.pdf, 2004.