Query translation between RDF and XML: A case study in the educational domain

Zoltán Miklós
Department of Information Systems and New Media
Vienna University of Economics and Business Administration
Augasse 6-9
Vienna, Austria
zmiklos@wu-wien.ac.at

Stefan Sobernig
Department of Information Systems and New Media
Vienna University of Economics and Business Administration
Augasse 6-9
Vienna, Austria
stefan.sobernig@wu-wien.ac.at

General Terms

RDF, XML, Semantic Web, eLearning

ABSTRACT

RDF is used as a central building block for the Semantic Web. Considering providers of learning resources, it is commonplace to store and exchange meta-information in XML rather than RDF. Instead of transforming meta-data artefacts into RDF, we integrate these meta-data by translating users' queries issued against RDF into queries over XML meta-data. We demonstrate the applicability of our query translation method in a concrete application scenario taken from the educational domain.

1. INTRODUCTION

The World Wide Web becomes more and more a marketplace for various informational goods and services. Providers of educational materials and services intensively use the Web as distribution channel and search engines help learners to find and access appropriate learning resources. The Semantic Web vision [1] depicts a possible alternative to this situation: Learning resources are described with a set of meta-data linked to some ontology and human searchers can issue queries against these ontologies to bring to light the learning resources needed. There are still many efforts needed to turn this vision into reality, even for a limited domain.

In this paper we concentrate on one particular issue we faced when trying to realize an Educational Semantic Web. The W3C has established several working groups to create standards for meta-data formats and ontology representation (RDF [2], OWL[3]), but currently there is a considerable amount of meta-data about learning resources available which is exported into and exchanged in XML rather than RDF. Therefore, we sought for a technique to integrate these meta-data with meta-data already represented in RDF. In particular, we analyzed how to transform queries expressed in a specific query language designed for RDF, QEL (Query Exchange Language, [4]) into corresponding XQuery queries [5] that can be evaluated over XML repositories.

The paper is organized as follows: Section 2 describes the context and motivation of our work. In Section 3 we analyze mapping strategies originally proposed for integrating XML sources and their applicability to our problem. Section 4 then introduces our transformation method. Section 5 presents related work and Section 6 concludes the paper.

WWW2005, May 10–14, 2005, Chiba, Japan.

2. ELENA – INTEGRATION IN AN EDUCATIONAL CONTEXT

In the scope of ELENA[1] we are developing a mediation infrastructure for learning services including web-based, but also traditional courses and learning materials. A common use scenario is the following: Providers offer meta-data about learning materials and services at their web portals. The web allows users to search for, visit and scan these pages for relevant courses or learning materials. Our goal is to improve this situation and provide an infrastructure that enables users to search for relevant information and create their own marketplace for learning resources. This joint effort can be seen as a step towards an Educational Semantic Web, as envisioned for example in [6].

The mediation infrastructure, called Smart Spaces for Learning [7, 8], integrates meta-data provided by a number of learning service providers. Though it is not intended to build a web-scale application, the integrated information opens way for personalized services and intelligent applications. An operative prototype designed for a human resource development scenario using the mediation infrastructure is available at http://www.hcd-online.com/ubp.

The currently connected providers cover a large spectrum of heterogeneous sources, ranging from *EducaNext* [9], a web-based knowledge brokerage platform, and *ULI*[2], a German academic network for sharing learning resources, to *Amazon*'s media store[3] or the Edutella P2P network [10].

2.1 Integration Architecture

Learning object repositories hold information on learning objects (meta-data). Our research was motivated by the need to integrate the meta-data available at distributed and heterogeneous learning repositories.

We apply the Learning Object Resource Interoperability

[1]See http://www.elena-project.org
[2]See http://www.uli-campus.de
[3]See http://www.amazon.com

(LORI) framework [11] which has lately become subject of an international standardization process. LORI is a layered integration architecture, which defines services to achieve interoperability among learning repositories. The framework includes core services, for example authentication, session management and application services like query management or provision facilities.

In most of the systems we aim at integrating meta-data already stored in RDF (*real* RDF repositories) or being bridged to RDF (*virtual* RDF repositories). The latter case we experienced with various systems storing, providing or exchanging valuable meta-data in XML. The providers of these learning repositories cooperate during the integration process in different manners, either by publishing their locally used meta-data schemes or by providing mappings to ELENA's common schema.

3. MAPPING STRATEGIES

In this section we review some generic mapping strategies originally outlined for the integration of XML only. We examine these methods briefly and evaluate in how far they are applicable to our integration scenario.

The automated or assisted process of creating mappings of two different representations of data is called *representation matching*. In our paper we assume that mappings are created manually unlike discussed in [12, 13]. The conceptualization and representation of *matchings* or *mappings* differ in various domains of application, e.g. schema integration, data warehousing and data mining, knowledge base construction and finally information integration systems [12, 13]. We restrict our work to the domain of information integration aiming at incorporating heterogeneous XML sources into a RDF environment. Therefore, we consider conflicting or at least varied structured representations of meta-data characterizing learning resources as subject to our integration method. These meta-data representations might be either encoded in XML described by Document Type Descriptions (DTDs) or XML Schemata (XML/S) or in RDF described by RDF Schema (RDF/S). This concrete run-time scenario has also been coined *semantic query processing* [13] as opposed to the practice of establishing mappings when designing federated information systems.

The term *mapping* denotes a set of *mapping statements* consisting of *mapping elements* and *mapping expressions* with the former declaring correspondences between representation or syntactic elements of the source representations. The latter describe the very nature of these relationships. Mappings might be either organized as *mapping relations* or *mapping tables*. The latter denote relations allowing for the usage of variables [14].

3.1 Mapping between different XML representations

Aguliera et al. [15] compare several approaches for mapping XML data. Here we outline their cases and analyze whether they can be applied to our RDF scenario. The mapping strategies discussed consider different levels of structural information and distinguish three types of mappings between XML trees.

Nevertheless, the mapping strategies are derived from general tree-structured data so that they can equally be applied to other data models than XML. As the following sections will show, various follow-up approaches, in particular borrowing from conceptual modelling, have applied these mapping strategies in their specific integration scenarios. The terms *node* and *path* are considered as general as possible for the purpose of the following evaluation. It will turn out that the actual conceptualization of nodes and paths for the respective mediating representations differ considerably from approach to approach depending on the underlying data models.

3.1.1 node-to-node (tag-to-tag)

When establishing correspondences between individual nodes – referring to XML elements in this case – mappings do not consider the document structure. Therefore, they do not take into account semantic information expressed in structural configurations. Mapping relations resulting from node-to-node mappings consist of tuples of mediating nodes and multiple corresponding nodes in the local information representations. The query translation algorithm simply replaces the label of a mediating node with its local correspondences in the respective query body issued against the mediating schema.

Figure 1 illustrates this mapping style for two mediating nodes, a learning resource and its title. A *node-to-node* mapping relation would thus be defined as MR1 whereas the resource's title would be identified by the local sources' elements in MR2:

```
MR1(LearningResource, Publication)
MR2(Title, Title)
```

If we consider both the mediating schema and its local sources being expressed in XML, a query over the mediating schema might be expressed in XPath as e.g. */LearningResource/Title*. An algorithm based on these mapping relations would, for instance, generate a corresponding XPath statement over local source A of the form *//Activity/ActivityLabel*. As for local source B, the translation would result in a corresponding query *//Publication/Title* with the latter being invalid considering the structure of this repositorium. Therefore, node-to-node mappings will certainly result in erroneous query transformations as the mappings and mapping relations do not reflect the structural configuration of the underlying information sources.

Node-to-node mappings or *atomic element-level mappings* [13] are thus easily perceivable but considerably limited in terms of mapping precision and mapping complexity. The increased complexity is due to the multiplicity of node-to-node mapping resulting from the entire set of possible correspondences.

3.1.2 tree-to-tree

This mapping strategy resembles characteristics of defining and creating *views* in a relational setting. Views as notion and concrete technique have been discussed both as derived and encapsulating constructs, for example rules or classes, and as *subschemas* of any underlying schema-like representation. In either case, views are essentially query definitions or unions of individual query bodies over a targeted schema. Tree-to-tree mappings are comparable to defining views inasmuch as any node in the mediating representation points to a set of concrete, isolated queries over the various source representations. The union of these individual query definitions bound to individual mediating nodes describe the entire mediating representation and thus

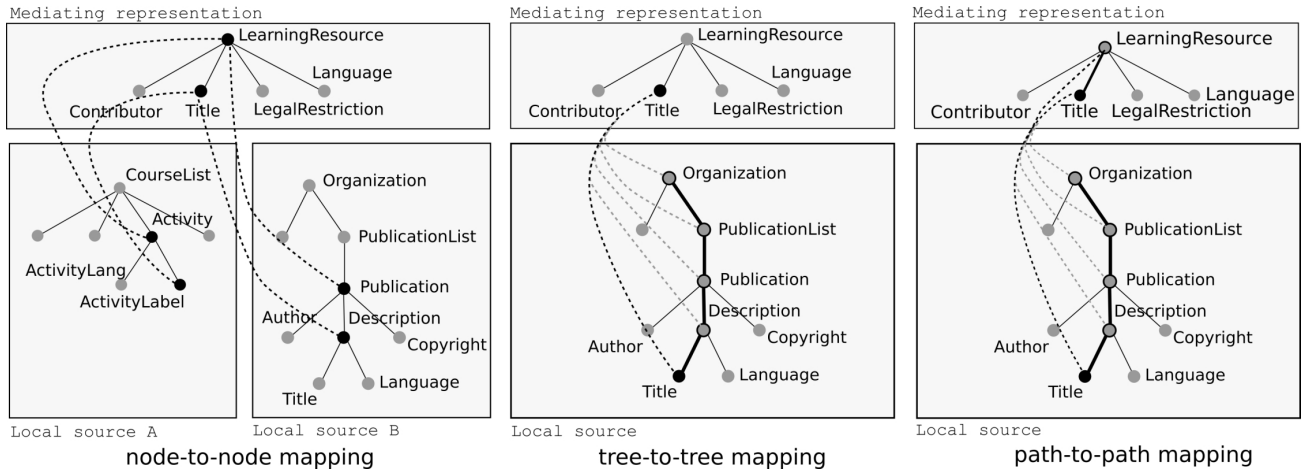

Figure 1: Mapping strategies

its structure. Tree-to-tree mappings may therefore be conceived as an ordered set of node-to-path mappings from the mediating representation's perspective in a global- as-view setting. In Figure 1 only such a single node-to-path mapping stands representatively for additional ones needed for each node of the mediating representation. To sum it up, tree-to-tree mappings are stored queries or unions of subqueries over local information sources, XPath expressions in this case. Figure 1 depicts such a scenario. The mapping relations outlined above for local source B transform into:

```
MR1(LearningResource, /Organization/
    PublicationList/Publication)
MR2(Title, /Organization/PublicationList/
    Publication/Description/Title)
```

The query */LearningResource/Title* would therefore be translated into a single view-generating query. Unless the established mapping relations are extended to mapping tables, this basic practice of tree-to-tree-mappings would create a certain redundancy in terms of irrelevant mapping elements i.e. path constructs. This becomes evident when considering that *MR1* is implicitly reflected in any mapping pointing to subtrees below the entry node. Introducing an appropriate variable syntax and therefore extending mapping relations to mapping tables would allow for a certain reduced mapping complexity. These more flexible mapping tables could take the following form:

```
MT1(LearningResource, /Organization/
    PublicationList/Publication)
MT2(Title, MT1/ActivityLabel, MT1/Title)
```

This mapping strategy, similarly described as *higher-level* or *non-atomic element-level mappings* by [13], considers a high level of structural information. These view-like mappings do not cause redundant or irrelevant mapping statements for a single local source. Nonetheless, they do not allow for any factorization between mappings of different local sources although there might occur considerable structural similarities within a common domain, as the educational one for instance.

3.1.3 path-to-path

The next available mapping strategy aside from node-to-node and tree-to-tree mappings is called *path-to-path*. In this case correspondences between paths in the mediating representation and paths in the various local source representations are created.

The conceptualization of path constructs varies considerable in various approaches adopting the path-to-path strategy depending on the data models used for the mediating representation and local information sources. Early contributions defined them as conventional XPath location paths or derivates thereof considering only XML element types, so called *tree paths* [16, 17]. XPath location paths as mapping elements are not limited to XML-like mediating representations. A recent approach uses location paths as mapping elements to create correspondences with a mediating representation based on the datalog model (see [18]). Follow-up approaches (see e.g. [19, 20, 21]) employed conceptual models (Entity-Relationship model, ORA-SS and others) as mediating representation and offer a different path concept. Amann et al. [21] identify for instance two types of so called *conceptual paths*: On the one hand *role paths*, on the other hand *concept paths*. The former are either constituted by single roles, i.e. ER relationship types, or a conjunction of single roles, also referred to as *derived* roles as they link two distanced concepts directly. Concept paths consist either of a single ER entity, i.e. concept, or a chain of concepts and roles. Considering paths in RDF we adopted the concept of *triple paths* as mapping element for ELENA's mediating representation (see Section 4.1 for details).

Mapping between paths resembles and combines major characteristics of the previously sketched techniques: On the one hand it inherits the property of node-to-node mappings allowing multiple occurrences of a mapping element, for example nodes and paths respectively, within the same set of mapping relations. On the other hand they increase the degree of how much structural information is considered, though to a lesser extent than tree-to-tree mappings. They incorporate rather substructures than the entire structural configuration of mediating and local representations into the mapping.

An example for path-to-path mappings can be constructed within the scope of Figure 1. The path /LearningResource/Title can be mapped to the local path //Organization/PublicationList/Publication/Description/Title for local source B.

As compared to tree-to-tree mappings, this last strategy

does not preserve the structure of entire (sub-)trees but rather the structural context of single nodes. At the same time it allows for the factorization of mapping elements, i.e. path constructs, as path structure might reveal considerable similarities for a single and even different local sources. Therefore the path-to-path approach represents an option that inherits advantages of both previous strategies and allows for the design of a mapping language convenient for the human integration engineer.

4. TRANSFORMING QEL QUERIES INTO XQUERY

4.1 Mapping language

In the following sections we outline a language for setting up mappings between elements of ELENA's mediating representation expressed in RDF and local educational information sources casted in XML. The mapping information is then used by the query translation algorithm in order to transform QEL into corresponding XQuery queries.

A single mapping is an XML application which comprises three logical sections: a header section and two body sections.

4.1.1 Defining the target

We distinguish between logical and physical information sources. A logical source may contain several physical ones, a logical information source does not necessarily correspond to a single XML document. A logical entity might be either casted in a single XML document or scattered across a collection of XML documents. The header section of the entire mapping requires to state a single, logical information source.

The header section may take the following form when applied to information source B in Figure 1, assuming first that we are dealing with a both logically and physically sole information source:

```
<q2xq:source id="pub" document="sourceB.xml"
    contextNode="/Organization/PublicationList/
    Publication" />
```

The *q2xq:source* element contains three attributes. First, name allows for defining a name for a physical – and under the current assumption also logical – information source . The second attribute provides information on the storage name of the XML information source, i.e. a XML Document either physically stored in the SQI target's filesystem or in a native XML data base system such as *eXist* [22]. The last attribute, *contextNode*, marks the node of entry or rather the *absolute context node* for the resulting XQuery query statement. It is absolute insofar as it serves as absolute point of reference for the subsequent mapping statements in the second body section.

Take the example of local source B in Figure 1. The source's structure embeds the actual learning resource, i.e. the Publication element, into a superordinate element, i.e. PublicationList, that can be structurally and semantically neglected in the subsequent correspondences between identified mapping elements. In this respect, the header section provides an optional selection of relevant subtrees of the source's document structure and thus facilitates establishing the actual mapping statements. In terms of the resulting XQuery expression, the header element will be casted in a FLWOR expression. In other words, the values of *document* and *contextNode* will constitute the required input expression of such a FLWOR statement.

We now drop the initial assumption the logical information source under consideration consists only of a single physical XML Document. We continue our considerations assuming a logical source that is splitted up into several physical carriers. Just imagine the aforementioned example featuring a publication list and publications being a generic XML dump of a relational storage system. In that case, the original relations publication list and publications will constitute two separate XML documents to be joined.

```
<q2xq:source id="publist" document="sourceB_1.
    xml" />
<q2xq:source id="pub" document="sourceB_1.xml"
    />
```

In order to combine these two physical data sources into a single logical one, the proposed mapping language provides another bridging facility to XQuery's FLWOR expression and its WHERE clause [23]. By expanding the header section with *q2xq:source* elements for each physical information source, the query translation algorithm is instructed to create a *join* between them.

4.1.2 Setting-up mapping statements

Referring to the methodological taxonomy presented in Section 3, we opted for a path-to-path mapping strategy for the reasons already discussed. Therefore, the mapping elements to be related to one another are *path constructs*.

Paths in RDF: In a general sense, a RDF path is considered a sequence of the form *node - predicate - node* as depicted in [2]. To put it differently, paths are equated with the concept of triples in RDF and corresponding serialization formats such as N-Triples [24] for instance. Therefore, RDF paths can be conceived as directed sequences of the form *subject node - predicate or property edge - object node* and might be coined triple paths.

Triple paths especially fulfil the requirement of unambiguity in order to be used as mapping elements. The RDF model can be seen as a directed, vertex- and edge-labelled graph. The labels attached to graph edges correspond to RDF predicates or properties. This raises the issue of unambiguity of edge labels or predicates as the multiple occurrence of predicate titles is not restricted or prohibited in the RDF syntax.

Consider first that path constructs in RDF are only identified by their predicates' labels. This assumption can be represented in a N-Triples-like style with <*> denoting an arbitrary, unspecified node element:

```
<*> <predicate> <*> .
```

Consider the two examples given in Figure 2. Both cases refer to the case of a publication list linking to publications but in two different settings. Setting 1 shows a publication list (n_1) that contains a single publication (n_2) both having either one or more detailed descriptions (n_3, n_4, n_5) attached. Setting 2 exhibits a publication list (n_1) containing two publications (n_2, n_3) with both of them being described in further detail (n_4, n_5) but having the same title (n_6). The fact that these two settings point to the multiple occurrence of semantically identical predicates, both at the same or different structural levels, might be considered an artifical construction. Whereas it is not appropriate with respect to

designing mediating representations for meta-data, it is not restricted or prohibited in the RDF model as such. Therefore, this assumption is satisfactory to outline the problem of unambiguity. When identifying paths in RDF only by edge

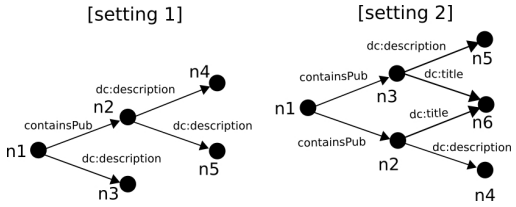

Figure 2: Unambuigity of paths

labels, only two distinct paths can be identified in Setting 1, i.e. containsPub and dc:description. Establishing a single correspondence between dc:description and a corresponding XML element at the local information source containing description information about a learning resource would not be unambiguous as it would be equally applied to describe the entire publication list. Even establishing multiple correspondences would not resolve the problem as they could not be assigned uniquely.

In order to resolve the problem of unambiguity, a first step could be the further identification of paths by considering further subject or object nodes:

```
<subject> <predicate> <*> .

<*> <predicate> <object> .
```

In Setting 1 the predicate path dc:description could only be located twice provided that the source node, i.e. the subject, is considered. When identifying a path by its target node, i.e. the object, all three occurrences could be uniquely determined. The latter is not correct when applied to Setting 2 as the predicate path dc:title could not be characterized clearly by its target node. Therefore, considering only a single identifier, either subject or object node, does not allow to construct unambiguous correspondences with mapping elements at the local information sources that hold in both settings.

Concluding from that, resolving the issue related to identifying paths unambiguously can only be achieved by pinpointing paths both by their source and object nodes, i.e. the subject and object connected. We therefore consider in accordance with the RDF specification *triple paths* the appropriate path construct and mapping element in the scope of our mapping language. Triple paths are thus RDF predicates extended by two *node identifiers*.

```
<subject> <predicate> <object> .
```

The first body section of a mapping thus contains an unambiguous set of triple paths, expressed in a straight forward XML format. Referring to Figure 1, the triple paths that are derived from the simplified mediating representation describing learning resources are shown in Example 2 at lines 6-9.

The triple notation is entirely adopted with the attribute *id* of each triplepath element serving as the binding variable to the right-hand or XML side of the mapping statements.

Paths in XML: Considering XML and the query model targeted, i.e. XQuery, path constructs are gathered from XQuery 1.0 and XPath 2.0 data model. XQuery path expressions are considered proper *location paths* as in XPath 1.0 [25] and are therefore identical to XPath 2.0. Nonetheless, the common data model of XQuery 1.0 and XPath 2.0 introduces various modifications compared to XPath 1.0. These include ordered sequences of nodes as return type of path expressions instead of unordered node-sets and both limitations and extensions in terms of location steps available, e.g. a reduced set of axis, generalized predicate statements and minor syntactic deviations in terms of comparison operators etc [23]. The proposed mapping language thus allows for the usage of XQuery path expressions or XPath 2.0 location paths as mapping elements with respect to local XML sources.

We refer to the mapping statement depicted in the path-to-path scenario in Figure 1 for the following remarks. The establishment of a correspondence between the RDF triple path <LearningResource><dc:title><Title> and the semantically analogous XML location Path /Organisation/PublicationList /Publication/Description/Title is realized in the second body section of a mapping. This section binds the afore-created descriptions of RDF triple paths to the corresponding location path and completes the mapping statements as such. The right-hand-side mapping element and the necessary binding to the first body section are achieved by the q2xq:mapping elements shown in Example 2 at lines 11-27.

The mapping language thus describes mapping statements as a binding between two XML elements. The left-hand or RDF side is represented by a *q2xq:triplepath* element in the first body section, the right-hand or XML side is casted into a *q2xq:mapping* element in the second body section. The latter allows for declaring a XPath location path in a specific XML document by referring to the header section and its target definitions by passing the respective value to the attribute *source*.

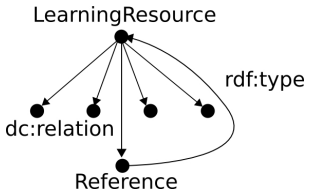

Figure 3: A cyclic mediating structure

Mapping of graph cycles: Concluding this section on the expressivity of the mapping language proposed, we would like to drop the simplifying assumption about directed acyclic graphs and consider the case of simple cycles in the RDF mediating representation. The description of the mediating RDF structure in terms of triple paths allows for the representation of graph cycles with cycles forming predicate paths such that the first node of the path corresponds to the last. Consider an extended example of the mediating representation given in Figure 1. Meta-data on learning resources are likely to comprise description information about related or even recommended learning resources, e.g. references. In that case the mediating representation in Figure 1 could be enriched by an additional node *Reference* representing a learning resource in the scope of ELENA's common schema: This cycle can be easily represented by two triple path statements. Cycles having a predicate path length greater than 1 may be represented as well. Cycles can be recognized by

looking for a node occurring at least once as subject and once as object in two distinct triple paths. A mapping for the cycle in Figure 3 takes the following form:

```
<q2xq:triplepath id="A" subject="
    LearningResource" predicate="dc:relation"
    object="Reference" />
<q2xq:triplepath id="B" subject="Reference"
    predicate="rdf:type" object="
    LearningResource" />
```

Note that in a global-as-view setting such a self-referential structure has to occur and find its correspondence within a single logical local source. The actual mapping element at the local XML source depends on the realization of the conceptual self-reference in terms of document structure. In XML this might be achieved for instance vertically by nesting elements of the same type in a recursive way or horizontally by making use of ID-IDREF relations. Both can be reflected in XQuery syntax, either using recursive functions in the former or ID-IDREF-based navigational functions built in XQuery such as *xf:id* [5] in the latter case. The representability of cyclic structures is nevertheless limited to simple cycles. On the one hand this is due to the manual generation of mappings and thus to the human perception of complex cyclic structures. On the other hand the processing of complex cycles in RDF has to be considered non-trivial as documented for instance in [26, 27].

4.2 Query translation algorithm

Our query translation algorithm transforms the input QEL query into a XQuery query by making use of the mapping rules. Regarding the global-as-view approach adopted in ELENA, query transformation basically consists of a translation of an input into an output query to be evaluated over a local XML source. Query transformations in local-as-view settings are usually referred to as *query rewriting* and involve an incomparably more complex transformation.

4.2.1 A primer for QEL

Query Exchange Language (QEL) is a query language specially designed for RDF, and is based on datalog. The QEL specification provides two different encoding styles, on the one hand datalog-QEL, on the other hand RDF/XML-QEL [4]. For reasons of clarity, the following section is in accordance with QEL's datalog notation. Consider a simple QEL query over the mediating representation described in Figure 1:

Example 1: A sample input QEL query

```
@prefix qel: <http://www.edutella.org/qel#>.
@prefix lom-rights: <http://ltsc.ieee.org
    /2002/09/lom-rights#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
?- qel:s(LearningResource,dc:title,Title),
qel:like(Title,'%education%'),
qel:s(LearningResource,dc:contributor,
    Contributor),
qel:s(LearningResource,dc:language,Language),
qel:s(LearningResource,lom-rights:
    copyright_and_other_restrictions,
    LegalRestriction).
```

The given datalog-QEL query might be intuitively interpreted as the following request: *Give me the title, the contributor or provider, the language and possible legal restrictions of all learning resources containing the term "education" in its title.*

The key concept borrowed from datalog are predicate expressions with QEL distinguishing between *matching* and *constraint* predicates. The most important pre-defined **matching predicate** in QEL is *qel:s* denoting a so called *statement literal*. The underlying common data model considers RDF data being organized in triple structures of the form *subject - predicate - object*. The range of allowed value types for subjects, predicates and objects are in accordance with the RDF specification [2]. The QEL matching predicate (*qel:s*) resembles this structure of RDF triples and serves as matching or binding facility to be used in queries. Corresponding to the range of value types in a RDF triple, each argument in a *qel:s* construct might be filled with an appropriate value type. Literals are thus proper values in datalog predicate expressions, URI references correspond to constant names. In addition, arguments can represent variables identified by capitalized names. Predicate expressions that contain variables as arguments are also referred to as *query literals*.

When examining the first matching predicate in Example 1 `qel:s(LearningResource,dc:title,Title)` both subject and object are variables whereas the predicate corresponds to a proper URI reference. Variables in qel:s constructs are bound to the entire spectrum of possible subject and object values stored in a RDF triple repository. Therefore, the qel:s construct taken from Example 1 selects all triples that contain the RDF predicate "dc:title" without any further restrictions.

Apart from matching predicates, the QEL syntax comprises another category of pre-defined predicates. This set of predicates helps constraining further the selection of matched triples based upon comparison operations on the RDF triples' values. They are referred to as **constraint predicates** and provide conventional value-based comparisons such as equals-, like-, greater-than- and less-than operators and verifications for node types and language encodings [4]. The construct `qel:like(Title,'%education%')` in Listing 1 shows such a value constraint, a like-operator more precisely, on all matching triples pre-selected by the *qel:s* construct mentioned before.

4.2.2 Translating a simple QEL query

In the following, we outline an algorithm to transform a QEL query as depicted in Example 1 into a corresponding XQuery query according to the mapping example given in Section 4.1. The entire mapping can be found in Example 2 attached to this paper. The translation algorithm is guided by the syntactic structure of XQuery's FLWOR expression (see [5]). QEL queries require to iterate through instances of learning resource elements. The analogous iteration can be achieved by a FLWOR expression in XQuery.

The translation algorithm uses only FOR, WHERE and RETURN blocks. The algorithm is therefore organized in three block declaring steps with the latter two distinguishing between a phase of *query parsing* and a phase of *mapping correspondences*. The parsing of the input QEL query aims at identifying relevant query elements, particularly *constraining* and *matching* constructs. In addition, all mapping statements relevant to this specific query elements are identified. The binding phase refers to the construction of an output XQuery query based on the previously identified QEL constructs and mapping statements.

Declaring the FOR clause: In a first step, the algorithm parses the mapping information of the header section of a given mapping file (see Section 4.1.1). Each *q2xq:source* element is considered and based on its attributes' value a FOR block is created. A XQuery FOR construct binds custom variables to some sort of input expression, e.g. the input function *doc* in our case. This input function returns the document node or some sub-level node of the physical XML document identified by both the attribute *document* and *contextNode*, pointing to a specific subtree as entry point for the iteration. This node of entry is bound to the variable defined by the attribute *id*. The heading mapping element given in Example 4.1.1 is thus transformed into the following partial XQuery expression:

```
for $pub
doc("sourceB.xml")/Organization/PublicationList
    /Publication
```

Provided that several q2xq:source elements are recognized they are attached to this initiating FOR block in terms of an additional variable-node binding and can be used to create joins between multiple XML documents at a later stage.

Declaring the WHERE clause: In a next step, the algorithm aims at extracting relevant constraint predicates in order to build a WHERE clause. This WHERE statement eliminates XML elements which do not match certain conditions. First, the algorithm identifies required mapping statements to map the constraining RDF elements. Then, the XPath location paths expressed in the identified mapping statements are attached to the previously defined path variables being context nodes. The XML element identified thereby serves as basis for the conditional operation. The extracted constraint predicate constructs specify the nature of these filtering conditions with conventional comparison operators (e.g. qel:equals, qel:greaterThan) being transformed into their XQuery equivalents (e.g. "=", ">"). More complex operators such as *qel:like* are equated with specific built-in functions of XQuery, *contains()* for instance. The constraint predicate statement `qel:like(Title,'%education%')` in Listing 1 would therefore be transformed in to the following WHERE clause:

```
where fn:contains($pub/Description/Title,"
    education")
```

The WHERE block is equally relevant when considering the transformation of conjunctions, disjunctions and negations expressed in the input QEL query.

Declaring the RETURN clause: The closing block, the RETURN clause, builds the result of the previously defined for-where expression. In other words, it exclusively returns tuples that match the constraints and allows for casting them in an user-defined XML output format. The latter is determined by QEL which requires a specific result format serialized in RDF/XML. The entire QEL result block comprises two interrelated sections, on the one hand the actual QEL *ResultSet* in terms of a RDF sequence containing result values, on the other hand another RDF sequence carrying the QEL result variables [4]. The two collections are related insofar as the sequential ordering determines the binding of result variables in the latter to the result values in the former. In order to populate these two result sections, the algorithm needs to identify all *matching predicates* of the input QEL query. Unlike datalog, QEL determines the order of result tuples. The *qel:s* constructs contain the information needed, particularly the result variables.

The two result sections are produced by another parsing and binding procedure. At first, the algorithm examines the input QEL query for all *matching predicates* and extracts their respective RDF *objects*, the result variables in QEL's terminology. By finding all mapping statements and thus location path correspondences to the triple paths represented by the matching predicates the first section is constructed. Each matching predicate is transformed into a RDF list item (*rdf:li*) whose value is determined by a corresponding XML element. This is identified by an absolute XPath location path composed of the context node variable and the location path from the respective mapping statement. Finally the object element of the respective matching predicate is added to the sequential list of result variables and thus bound to the previously rendered value. The entire output XQuery query resulting from the QEL query in Example 1 and the underlying mapping in Example 2 are attached to this paper. The two result section described above are shown at lines 14-29.

4.2.3 Issues

At this stage we would like to discuss important aspects concerning more complex transformations. They include brief accounts on the correspondence of negation operators as well as transforming conjunctive and disjunctive QEL queries into their XQuery representations.

Negation: As there is no negation of matching predicates available in QEL, negations in a limited sense may exclusively be applied to constraint predicates. In the course of parsing the input QEL query when declaring the WHERE clause the identified constraint predicates are checked for QEL's negation operator ("-"). Following this, they are transformed similar to non-negated constraint predicates where the negation operator ("not") is added. The negated constraint predicate `-qel:like(Title,'%education%')` would therefore be transformed into `not(fn:contains($pub/Description/Title,"education"))`

Conjunction: Conjunctions in QEL are represented by comma-separated sequences of predicate expressions [4]. Once again matching and constraint predicates have to be distinguished: Conjunctive sets of the former as given in Example 1 pre-select a set of triples subject to further restrictions. Conjunctions between constraint predicates are directly translated into a logical AND operator in the WHERE clause of the corresponding XQuery.

Disjunction: Disjunctions are expressed as in datalog: several rules with the same *rule head*. The rule head itself is a query literal on the left-hand side of a rule definition while at the right-hand side an arbitrary order of query literals, both matching and constraint predicates, can be specified.

A disjunction comprising constraint predicates is transformed into conditional elements of a WHERE clause, connected by a logical OR operator.

5. RELATED WORK

Several fields of research emerged as relevant and related to our efforts. In the educational domain, in particular in the scope of Edutella, Qu and Nejdl [28] staged a comparable approach to integrate a SCORM meta-data repository stored in XML with a RDF-based P2P infrastructure by means of – though not exclusively – query translation. The entire in-

tegration involves first a replication and modification of the targeted XML repository into a generic RDF-graph-based meta-data view which is represented in a XML serialization of RDF triples. Second, they offer a complementary wrapper implementation that translates between users' QEL and XQuery queries over the replicated, normalized and XML-encoded RDF meta-data repository. Their contribution differs at least in two aspects: On the one hand they provide a technique to integrate arbitrary common RDF representations and QEL queries with a specific and complex local meta-data representation (SCORM) whereas we provide a facility to target arbitrary and less complex local XML meta-data storages through a specific and pre-defined RDF mediating representation. On the other hand their approach reflects an integration scenario which allows replicating entire repositories with our translation technique being applicable to more restricted scenarios where only pre-selected meta-data are exposed by integration partners.

In the more general discussion on integrating heterogeneous XML sources some key approaches can be distinguished. One group applies XML itself as mediating representation. Their relevance to our efforts results from their analysis of mapping strategies, already discussed in Section 3. Important contributions in this group include Xyleme [17, 29, 15] and Lee et al. [30]. A second group of authors [20, 21] criticize the use of XML as a mediating schema and proposed conceptual models for the integration of XML sources instead. Although they use self-defined conceptual models or ER derivates, they adopt the mapping strategies developed for XML. Relevant projects include STyX [20, 21] and ORA-SS [19].

Finally, we identified several approaches using the RDF graph model, i.e. either RDF or RDF/S, as mediating vehicle for integrating XML. PEPSINT [31] is a Peer-to-Peer system based upon a super-peer infrastructure and a global RDF ontology against which RDQL queries are evaluated. Depending on the target's meta-data model the original RDQL query is either simply reformulated according to mapping rules or syntacticly translated into a XQuery query over a XML repository. In a global-as-view setting the mapping is realized - in contrast to our solution - semi-automatically both at the global and local stage. First a local RDF/S ontology is generated for each RDF and XML repository with the local ontology preserving structural or nesting information of XML trees. At the local level PEPSINT applies a tree-to-tree mapping strategy as each concept in the local RDF/S ontology is mapped to a XML location path. In a second step, a node-to-node mapping is established between single concepts of the global and local RDF/S meta-data representations. Based upon this *combined* mapping strategy the query translation algorithm provides for a translation back and forth between XQuery and RDQL. Contrasting to PEPSINT, query translation in ELENA is performed only in a single direction (from RDF to XML).

Another research project focusing on integration of RDF and XML is SWIM [18]. The SWIM server hosts the mediating and query transformation facilities, which integrate not only XML meta-data but also relational databases. SWIM does not apply the mediator-wrapper architecture but it relies on a single wrapper solution. This implies that SWIM is based on a centralized mapping methodology whereas ELENA and PEPSINT operate in a decentralized manner with respect to mappings and query translation. This requires the employment of a single mapping methodology which provides the expressivity to represent all data models subject to integration. SWIM achieves this by using a datalog-based mapping language which incorporates XPath location paths as datalog atoms. As for query transformation, the proposed algorithm translates between RQL over the virtual mediating RDF/S representation and XQuery queries over local XML repositories.

Piazza [32] is also a mediating infrastructure that enables the integration of XML data into a RDF-based environment. Zachary et al. adopt a local-as-view integration technique and apply an extenstion to XQuery as mapping language to expose XML meta-data as virtual RDF repositories. Due to the usage of an extended XQuery syntax they propose a necessary query evaluation algorithm. In contrast to Piazza our approach can be applied to any standard XQuery processor.

Other aspects relevant to our work are handling RDF cycles and designing mapping languages. Barton [26], for instance, applies indexing to RDF structures and resolve cycles in this context. Various mapping languages are proposed in the context of integrating heterogeneous XML sources. A rule-based XML syntax called LMX (Language for Mapping XML) has been applied by [33] underlining its applicability for the tool-assisted mapping generation by human integration engineers. Other approaches on integrating XML use datalog syntax [18], RDF/XML [34] and XQuery [32] as mapping languages.

6. CONCLUSION AND FUTURE WORK

We realized a query translation method, and successfully integrated XML data with a RDF-based application based on a manually created mapping language. These efforts involved an analysis of existing XML mapping techniques that we adapted for our RDF-XML scenario, resulting in a XML-encoded mapping language and a corresponding query translation algorithm. We were able to translate all user queries in our application and integrate entire XML meta-data repositories into ELENA's RDF environment. The mapping language enables engineers to design complex mappings, however in case of big structures it can become impracticably complex. Translating QEL queries and evaluating the resulted XQuery performed equally well as processing the same QEL query on a RDF meta-data set of the same size.

We did not formally analyze the soundness of our method. Therefore, we are currently pursuing a number of research directions to identify limitations to our prototype implememtation. If we also formalize the application specific constraints, such a formal proof is thinkable. In the context of meta-data integration we consider this kind of translation technique complementary to the use of emerging versatile query languages [35] applicable both to RDF and XML.

Acknowledgements

The authors would like to express their gratitude to Michael Kamleitner for his contributions.

7. REFERENCES

[1] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
[2] RDF. (http://www.w3.org/RDF/)

[3] OWL. (http://www.w3.org/TR/owl-features/)
[4] QEL. (http://edutella.jxta.org/spec/qel.html)
[5] XQuery. (http://www.w3.org/TR/xquery/)
[6] Anderson, T., Whitelock, D.: The Educational Semantic Web: Visioning and Practicing the Future of Education. Volume 1 of Journal of Interactive Media In Education. (2004)
[7] Simon, B., Miklós, Z., Nejdl, W., Sintek, M., Salvachua, J.: Smart Space for Learning: A Mediation Infrastructure for Learning Services. In: Proceedings of the Twelfth International World Wide Web Conference (WWW2003), Budapest (2003)
[8] Simon, B., Dolog, P., Miklós, Z., Olmedilla, D., Sintek, M.: Conceptualising smart spaces of learning. Journal of Interactive Media in Education **9** (2004) Special Issue on the Educational Semantic Web.
[9] Law, E., Maillet, K., Quemada, J., Simon, B.: Educanext: A service for knowledge sharing. In: Proceedings of the 3rd Annual Ariadne Conference. (2003) Katholieke Univeriteit Leuven.
[10] Edutella. (http://edutella.jxta.org/)
[11] Simple Query Interface (SQI) for Learning Repositories. http://www.prolearn-project.org/lori (2004)
[12] Doan, A.: Learning to Map between Structured Representations of Data. PhD thesis, University of Washington (2002)
[13] Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal **10** (2001) 334–350
[14] Kementsietsidis, A., Arenas, M., Miller, R.J.: Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In: ACM SIGMOD International Conference on Mangement of Data. (2003) 325–336
[15] Aguilera, V., Cluet, S., Milo, T., Veltri, P., Vodislav, D.: Views in a Large Scale XML Repository. VLDB Journal **11** (2002) 238–255
[16] Reynaud, C., Sirot, J.P., Vodislav, D.: Semantic Integration of XML Heterogeneous Data Sources. In: Proceedings of the International Database Engineering & Applications Symposium, IEEE Computer Society (2001) 199–208
[17] Delobel, C., Reynaud, C., Rousset, M.C., Sirot, J.P., Vodislav, D.: Semantic Integration in Xyleme: a uniform tree-based approach. Data & Knowledge Engineering (2003) 267–298
[18] Christophides, V., Karvounarakis, G., Koffina, I., Kokkinidis, G., Magkanaraki, A., Plexousakis, D., Serfiotis, G., Tannen, V.: The ics-forth swim: A powerful semantic web integration middleware. In: First International Workshop on Semantic Web and Databases - VLDB 2003, Berlin, Humboldt-Universität (2003)
[19] Yang, X., Lee, M.L., Ling, T.W.: Resolving Structural Conflicts in the Integration of XML Schemas: A Semantic Approach. In: Conceptual Modeling - ER 2003. Volume 2813 of Lecture Notes in Computer Science. (2003) 520–533
[20] Fundulaki, I., Marx, M.: Mediation of XML Data through Entity Relationsip Models. In: First International Workshop on Semantic Web and Databases (SWDB) 2003. (2003) 349–357
[21] Amann, B., Beeri, C., Fundulaki, I., Scholl, M.: Ontology-Based Integration of XML Web Resources. In: Proceedings of the 1st International Semantic Web Conference (ISWC 2002). (2002) 117–131
[22] eXist. (http://exist.sourceforge.net/)
[23] Chamberlin, D., Draper, D., Fernández, M., Kay, M., Robie, J., Rys, M., Soméon, J., Tivy, J., Wadler, P.: XQuery from the Experts - A Guide to the W3C XML Query Language. Addison-Wesley, Boston (2004)
[24] N-Triples. (http://www.w3.org/2001/sw/RDFCore/ntriples/)
[25] XPath. (http://www.w3.org/TR/xpath)
[26] Barton, S.: Designing Indexing Structure for Discovering Relationsships in RDF Graphs. In: Databáze, Texty, Specifikace a Objekty (DATESO) 2004. (2004) 7–17
[27] Matono, A., Amagasa, T., Yoshikawa, M., Uemura, S.: An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays. In: First International Workshop on Semantic Web and Databases (SWDB) 2003. (2003) 151–168
[28] Nejdl, W., Qu, C.: Integrating XQuery-enabled SCORM XML Metadata Repositories into a RDF-based E-Learning P2P Network. Educational Technology & Society **7** (2004) 51–60
[29] Rousset, M.C., Reynaud, C.: Knowledge representation for information integration. Information Systems (2004) 3–22
[30] Lee, M.L., Yang, L.H., Hsu, W., Yang, X.: XClust: Clustering XML Schemas for Effective Integration. In: 11th ACM International Conference on Information and Knowledge Management (CIKM), McLean, Virginia (2002)
[31] Cruz, I.F., Xiao, H., Hsu, F.: An Ontology-based Framework for Semantic Interoperability between XML Sources. In: Eighth International Database Engineering & Applications Symposium (IDEAS 2004). (2004)
[32] Ives, Z.G., Halevy, A.Y., Mork, P., Tatarinov, I.: Piazza: mediation and integration infrastructure for Semantic Web data. Web Semantics: Science, Services and Agents on the World Wide Web **1** (2004) 155–175
[33] Vdovjak, R., Houben, G.: RDF Based Architecture for Semantic Integration of Heterogeneous Information Sources. In Simon, E., Tanaka, A., eds.: International Workshop on Information Integration on the Web - WIIW2001. (2001) 51–57
[34] Barrett, T., Jones, D., Yuan, J., Sawaya, J., Uschold, M., Adams, T., Folger, D.: RDF Representation of Metadata for Semantic Integration of Corporate Information Sources. In: WWW2002. (2002)
[35] Bry, F., Koch, C., Furche, T., Schaffert, S., Badea, L., Berger, S.: Querying the Web Reconsidered: Design Principles for Versatile Web Query Languages. International Journal on Semantic Web and Information Systems **1** (2005) 1–20

Example 2: A sample mapping

```
<?xml version="1.0" encoding="iso-8859-1"?>
<QEL2XQueryMapping xmlns:q2xq="http://www.elena-project.orf/ns#q2xq">

        <q2qx:source id="pub" document="sourceB.xml" contextNode="/Organization/PublicationList/
            Publication"/>

        <q2xq:triplepath id="A" subject="LearningResource" predicate="dc:contributor" object="
            Contributor" />
        <q2xq:triplepath id="B" subject="LearningResource" predicate="dc:title" object="Title" />
        <q2xq:triplepath id="C" subject="LearningResource" predicate="
            lom:copyright_and_other_restrictions" object="LegalRestriction" />
        <q2xq:triplepath id="D" subject="LearningResource" predicate="dc:language" object="Language
            " />

        <q2xq:mapping>
           <q2xq:lhs triplepath="A" />
           <q2xq:rhs source="pub" locationpath="/Author" />
        </q2xq:mapping>
        <q2xq:mapping>
           <q2xq:lhs triplepath="B" />
           <q2xq:rhs source="pub" locationpath="/Description/Title" />
        </q2xq:mapping>
        <q2xq:mapping>
           <q2xq:lhs triplepath="C" />
           <q2xq:rhs source="pub" locationpath="/Copyright" />
        </q2xq:mapping>
        <q2xq:mapping>
           <q2xq:lhs triplepath="D" />
           <q2xq:rhs source="pub" locationpath="/Language" />
        </q2xq:mapping>

</QEL2XQueryMapping>
```

Example 3: A sample output XQuery query

```
xquery version "1.0";
<rdf:RDF
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:edu="http://www.edutella.org/qel#"
         xmlns:RDF/S="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
 <rdf:Description>
    <rdf:type rdf:parseType="LearningResource"/>
    <rdf:type rdf:resource="http://www.edutella.org/qel#ResultSet"/>
    {
         for $pub doc("sourceB.xml")/Organization/PublicationList/Publication
         where fn:contains ($pub/Description/Title, "education")
           return
                <edu:result>
                        <rdf:Seq>
                                <rdf:li> { string($pub/Author) } </rdf:li>
                                <rdf:li> { string($pub/Description/Title) } </rdf:li>
                                <rdf:li> { string($pub/Copyright) } </rdf:li>
                                <rdf:li> { string($pub/Language) } </rdf:li>
                        </rdf:Seq>
                </edu:result>
                <edu:resultVariables>
                        <rdf:Seq>
                                <rdf:li rdf:resource="#Contributor"/>
                                <rdf:li rdf:resource="#Title"/>
                                <rdf:li rdf:resource="#LegalRestriction"/>
                                <rdf:li rdf:resource="#Language"/>
                        </rdf:Seq>
                </edu:resultVariables>    }
  </rdf:Description>
</rdf:RDF>
```