

# Searching SCORM Metadata in a RDF-based E-Learning P2P Network Using XQuery and Query by Example

Changtao Qu  
Learning Lab Lower Saxony  
University of Hannover  
Expo Plaza 1, D-30539, Hannover, Germany  
qu @learninglab.de

Wolfgang Nejdl  
Learning Lab Lower Saxony  
University of Hannover  
Expo Plaza 1, D-30539, Hannover, Germany  
nejdl @learninglab.de

## Abstract

*The SCORM (Sharable Content Object Reference Model) Metadata Information Model is a reference to the IMS Learning Resource Metadata Information Model, which itself is based on the IEEE 1484.12.1 LOM (Learning Object Metadata) standard. Since the LOM standard possesses a rather complex data model consisting of over 60 metadata entries, searching SCORM metadata has to overcome several challenges to both queries' run-time performance and efficient, user-friendly query GUI (Graphical User Interface) design. Furthermore, because we start from Edutella, a RDF-based E-Learning P2P (Peer-to-Peer) network, we also have to address the incompatibility between the SCORM Metadata data model and the RDF data model, as well as the syntax incompatibility between XML and RDF. In this paper we propose an approach for searching SCORM metadata in Edutella, which addresses aforementioned challenges through using an XQuery-enabled triple-like SCORM metadata data view and a QBE (Query by Example) based SCORM query GUI. While the triple-like data view efficiently bridges the SCORM Metadata data model and the RDF data model, and at the same time greatly improves queries' run-time performance, the QBE-based SCORM query GUI facilitates the query construction process.*

## 1. Introduction

The SCORM (Sharable Content Object Reference Model) is a model that references a set of interrelated technical specifications and guidelines with the purpose of achieving the reusability, accessibility, durability, and interoperability of learning resources [1]. The cornerstone of the SCORM is a Web-based learning "Content Aggregation Model" defining how learning resources can be identified and described, aggregated into a course or

portion of a course, and moved between different LMSs (Learning Management Systems) or content repositories. In the latest SCORM 1.2, the SCORM Content Aggregation Model is composed of three parts [1]:

- *Content Model*: Nomenclature defining the content components of a learning experience. Three types of defined components are Assets, SCO (Sharable Content Object), and Content Aggregation.
- *Metadata*: A mechanism for describing specific instances of the components of the Content Model.
- *Content Packaging*: Define how to represent the intended behavior of a learning experience (Content Structure) and how to package learning resources for movement between different LMSs (Content Packaging).

Among these three parts, the key to the learning resource searching and locating is the SCORM Metadata, which provides descriptive information for each of the SCORM Content Model components to enable the discovery of learning resources at different aggregation levels. From its basis, the SCORM Metadata Information Model is a reference to the IMS Learning Resource Metadata Information Model [6], which itself is based on the IEEE 1484.12.1 LOM (Learning Object Metadata) standard [5]. In addition, the SCORM Metadata also references the IMS Learning Resource Metadata XML Binding Specification [6] thus provides an XML representation for the SCORM Metadata data model. Throughout this paper the "SCORM Metadata" implies the SCORM Metadata XML binding, as currently the SCORM Metadata RDF binding is not yet available.

The SCORM Metadata data model is rather complex, consisting of over 60 metadata entries and even recursive data structure, e.g., in the category of "Classification". In order to store and manage SCORM metadata based on such a complex data schema, a common practice in E-Learning is to use native XML databases to construct SCORM metadata repositories [8]. This is partly because that currently some "XML-enabled" RDBs (Relational DataBases) or OODBs (Object-oriented DataBases)

cannot satisfactorily handle the storage and management of complex XML data in an efficient, user-friendly manner, whereas native XML databases provide a more natural and straightforward solution [3]. In native XML databases, SCORM metadata can be stored and managed in their original hierarchical XML format rather than in some other representations, e.g., decomposed relational tables in RDBs, or decomposed objects in OODBs. This implies that the database schema used to define how the XML is stored is virtually identical to the XML data schema. Therefore, based on the SCORM Metadata data schema, multiple SCORM metadata profiles can be contained in a single collection and thus be queried as a whole through using W3C XQuery [2]. Moreover, the stored SCORM metadata profiles can be easily updated through direct manipulation on XML fragments instead of on the whole profiles.

However, despite of aforementioned advantages, native XML databases also introduce several challenges to the searching of SCORM metadata.

First, since the SCORM Metadata data model is quite complicated, queries directly against the original SCORM metadata through using XQuery considerably harm queries' run-time performance. At this point, some existing practices, e.g., using a tailored SCORM Metadata data model, downwards metadata mapping [9], etc., will unavoidably lead to the loss of original SCORM metadata information.

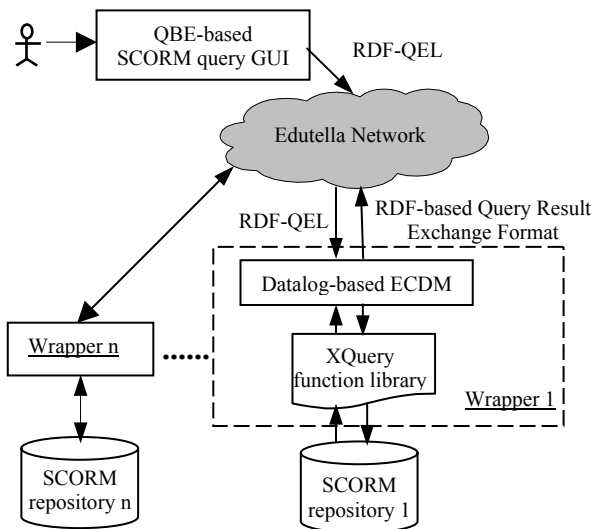
Second, since native XML databases do not support some structured query languages, the complexity of the SCORM Metadata data model also challenges the query construction process. For those more than 60 SCORM metadata entries, since the users' query interest are unforeseeable, we cannot expect which metadata entry would be queried and how the Boolean logic between the queries against multiple metadata entries would seem. In fact, including all SCORM metadata entries and all possible query Boolean logic in a "forms" based query GUI (Graphical User Interface) is a straightforward first idea, but can lead to some cumbersome and inefficient query experiences.

Finally, in the context of our project, the searching and locating of SCORM metadata goes beyond a simple learning resource sharing scenario such as a centralized learning resource portal, it occurs in a RDF-based E-Learning P2P (Peer-to-Peer) network Edutella, which aims at accommodating heterogeneous learning resource metadata repositories in a P2P manner and further facilitating the exchange of learning resource metadata between these repositories based on RDF [7]. Edutella makes aforementioned two challenges more crucial, as in Edutella the searching and locating of SCORM metadata also has to tackle the incompatibility between the SCORM Metadata data model and the RDF data model, as well as the syntax incompatibility between XML and RDF.

In this paper we propose an approach for searching SCORM metadata in Edutella, which addresses these challenges through using an XQuery-enabled triple-like SCORM metadata data view and a QBE (Query by Example)[10] based SCORM query GUI. While the triple-like data view efficiently bridges the SCORM Metadata data model and the RDF data model, and at the same time greatly improves queries' run-time performance, the QBE-based SCORM query GUI facilitates the query construction process.

## 2. Searching SCORM metadata in Edutella

Edutella employs a wrapper-like architecture to integrate heterogeneous metadata repositories. In figure 1 we illustrate the Edutella integration architecture for SCORM metadata repositories.



**Figure 1. Edutella integration architecture for SCORM metadata repositories**

The key to such an integration architecture is the Edutella Common Data Model (ECDM), which is shared by all metadata repositories and provides the common data view of the underlying metadata. At its basis, the ECDM is a binary relational data model, which is defined in full compliance with the RDF data model and uses Datalog [4] as its internal query language. Externally, Edutella defines a common query language: RDF Query Exchange Language (RDF-QEL) and a common result exchange format for the whole Edutella network using RDF syntax [7]. The two provide a uniform way to represent queries and query results in Edutella.

According to the Edutella integration architecture for SCORM metadata repositories, searching SCORM metadata in Edutella consists of three phases. First of all, we have to manipulate SCORM metadata stored in native XML databases to generate a SCORM metadata data view, whose underlying data model is compatible with the

ECDM/RDF data model. This is actually the prerequisite for SCORM metadata repositories to be integrated into Edutella and sequentially be queried via ECDM's internal query language Datalog. Here we also have to note that the SCORM metadata data view should be able to be easily manipulated through using XQuery in order to ensure queries' run-time performance. The second phase is to translate the Edutella common query language RDF-QEL into XQuery, the local query language of native XML databases, as well as to transform the XML-based local query results into the RDF-based Edutella common result exchange format. This phase can be referred to as the development of the wrapper program for SCORM metadata repositories. The third phase is to construct RDF-QEL queries against SCORM metadata repositories. Since RDF-QEL queries might be rather complicated for ordinary users, we here implement a QBE-based SCORM query GUI to facilitate the query construction process.

### 2.1. Phase 1: generate the triple-like SCORM metadata data view

In the first phase, we propose a triple-like SCORM metadata data view, which can be generated through using XQuery without the loss of any original SCORM metadata information. The triple-like data view has two features. First, its underlying data model is 100% compatible with the ECDM/RDF data model thus can be queried via ECDM's internal query language Datalog without any problem. Second, it adopts a very simple XML syntax, which can be easily manipulated through using XQuery thus can ensure queries' run-time performance. In figure 2 and figure 3 we take a SCORM metadata entry: lom.general.catalogentry as an example to demonstrate the generating process of the triple-like data view. Figure 2 shows the graphical data model of this metadata entry in the form of XML Schema. Figure 3 shows an example metadata instance.

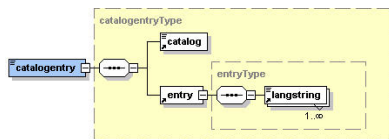


Figure 2. The graphical data model of the SCORM metadata entry: lom.general.catalogentry



Figure 3. An example instance of the SCORM metadata entry: lom.general.catalogentry

In order to generate the triple-like data view, the example metadata instance is represented through a RDF graph, which is then serialized using a simple XML syntax, as illustrated in figure 4 and figure 5. The serialization is realized through using a self-developed XQuery function library.

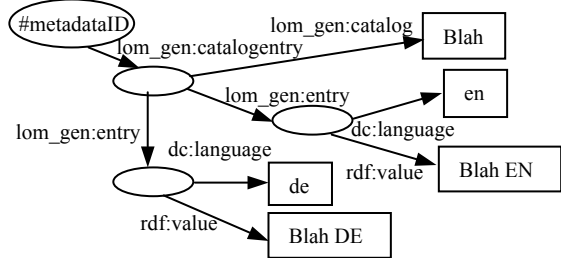


Figure 4. A RDF graph representing the example metadata instance



Figure 5. The XML serialization of the RDF graph illustrated in figure 4

From figure 5 we can see that taking advantage of a very simple XML syntax, the triple-like data view can be easily manipulated through using XQuery. Moreover, since currently most of native XML databases support the indexing on specific XML elements, the queries' run-time performance against the triple-like data view could be further improved through the indexing on "//subject", "//predicate", and "//object".

As the underlying data model of the triple-like data view is compatible with the ECDM/RDF data model, any SCORM Metadata instance could be represented as a RDF

graph in the way demonstrated in figure 4. However, the triple-like SCORM metadata data view is not 100% compatible with the IMS Learning Resource Metadata RDF binding [6], which is expected to become the potential SCORM Metadata RDF binding. Actually, the potential SCORM RDF binding is proposed without taking into account the compatibility with the SCORM XML binding [6]. It takes advantage of the semantic richness of RDF and goes beyond a simple syntactic level representation, which makes it quite hard to transform the SCORM XML binding into its potential RDF binding without the loss of original metadata information. Therefore, while handling SCORM XML binding metadata, we should not expect 100% compatibility with the potential SCORM RDF binding.

On the other hand, we should be also aware that it is unnecessary to achieve 100% compatibility between the SCORM metadata data view and the potential SCORM RDF binding. From the syntactic view, the potential SCORM RDF binding uses several RDF built-in properties, e.g., `rdf:type`, in a rather ambiguous way, which might increase the complexity and the run-time cost of XQuery functions. While handling self-contained SCORM XML binding metadata, it is unnecessary for us to bear such sort of additional query overhead.

Nevertheless, we still tried to achieve the maximal compatibility between the triple-like SCORM metadata data view and the potential SCORM RDF binding with the purpose of handling SCORM XML and RDF binding metadata in a uniform way in Edutella. Some efforts include, e.g., using mostly the same namespaces proposed in the potential SCORM RDF binding, remaining the compatibility with Dublin Core, Dublin Core Qualifiers, and vCard in the triple-like data view, etc. These efforts have actually covered most of principal design criteria for the potential SCORM RDF binding [6].

## 2.2. Phase 2: develop the wrapper program for SCORM metadata repositories

Developing the wrapper program for SCORM metadata repositories consists of two tasks: (1) translating RDF-QEL into XQuery; and (2) transforming XML-based query results into RDF-based Edutella common result exchange format. Since we adopt a triple-like SCORM metadata data view, whose underlying data model is quite close to the ECDM/RDF data model, and also because XQuery is capable of returning query results in any desirable XML format, completing the second task is relatively easy. In this section, our discussion will be focused on the first task.

According to the Edutella integration architecture illustrated in figure 1, the first task can be further divided into two sub-tasks. The first sub-task is to translate RDF-QEL into Datalog, which, as ECDM's internal query

language, serves as the unique query interface bridging RDF-QEL and various local query languages. In Edutella, this sub-task is completed through using a common parser program detailed in our previous publication [7]. The second sub-task is to translate Datalog into XQuery, more precisely, into sets of calls to the self-developed XQuery function library.

Datalog is a non-procedural, relationally complete query language based on Horn clauses without function symbols [4]. As its counterpart, XQuery is also relationally complete, being able to perform the operations of relational algebra on XML's hierarchical data model. Moreover, as a functional, strongly typed query language, XQuery can also conduct some complicated query operations such as recursive query. While handling SCORM metadata entry: `lom.classification.taxonpath.taxon`, we have taken advantage of XQuery's recursive query capability.

In order to demonstrate the translation process from Datalog to XQuery, we illustrate an example Datalog query in figure 6, which could be read as: *find a SCORM metadata record, whose lom.general.title entry contains English value "computer" and lom.general.keyword entry contains English value "TCP", or a SCORM metadata record, whose lom.general.description entry contains English value "network"*.

```

scorm(X) :- lom_gen:title(X,U), dc:language(U,"en"), rdf:value(U,"computer"),
           lom_gen:keyword(X,V), dc:language(V,"en"), rdf:value(V,"TCP")
scorm(X) :- lom_gen:description(X,W), dc:language(W,"en"), rdf:value(W,"Network")
? - scorm(X)

```

(Here X,U,V,W are Datalog variables)

Figure 6. An example Datalog query

One basic construct of Datalog is the Literal, which describes ground assertion and can be represented in a simplified form corresponding to the binary relational data model as:  $P(arg1, arg2)$ , where  $P$  is Predicate that might be a relation name or arithmetic predicates (e.g., " $<$ ", " $>$ ", etc.), and  $arg1, arg2$  are Arguments that might be variables or constants. A Datalog query can be expressed as a set of Datalog rules. Each Datalog rule has a general representation as *head :- literal1, literal2, ..., literaln*, where *head* is a single positive Literal, and *literal1 to literaln* are a set of Literals conjunctively called the body of the Datalog rule. The disjunction in Datalog is expressed as a set of rules with the identical head. As illustrated in figure 6, the example Datalog query consists of two rules covering conjunctive and disjunctive query. It can be translated into sets of calls to the self-developed XQuery function library, as illustrated in figure 7.

```

let $p := query_on_element_with_langstring("lom_gen_title","", "en", "computer")
let $q := query_on_element_with_langstring("lom_gen_keyword","", "en", "TCP")
let $r := query_on_element_with_langstring("lom_gen_description","", "en", "Network")
return handle_Boolean_OR( handle_Boolean_AND($p union $q) union $r)

```

Figure 7. XQuery translated from the example Datalog query

The self-developed XQuery function library contains sets of functions used to query different SCORM metadata entries. The returned query results are further handled by two specific XQuery functions: “handle\_Boolean\_OR” and “handle\_Boolean\_AND”, which are used to manage the Boolean logic between the queries against multiple metadata entries. These two functions are also responsible for eliminating duplicate result sets.

### 2.3. Phase 3: search SCORM metadata using QBE

QBE is a graphical language originally designed for querying RDBs. The idea behind QBE is that the user provides an example of outputs that he expects from the query and constructs the query by filling example tables [10]. The principal goal of applying QBE is to simplify the query construction process.

Whereas QBE fits quite well with RDBs in that QBE’s tabular query interface is quite analogous to the internal tabular structure of RDBs, it cannot be directly used to query a native XML database, which, as a document database by nature, adopts hierarchical tree-like data model to store XML data. In the third phase, we propose an improved QBE, which uses a visual template to represent the query against individual SCORM metadata entry, and further adopts a single table to represent the Boolean logic between multiple visual templates. In this way, while the visual template provides a quite analogous representation of the internal structure of individual SCORM metadata entry stored in native XML databases, the single tabular structure inherits QBE’s original advantage for representing the Boolean logic between queries against multiple metadata entries.

In general, the QBE-based SCORM query GUI has four features: (1) arbitrary SCORM metadata entry could be taken as the “example”; (2) user-friendly drag & drop manipulation based on the SCORM XML binding DOM (Document Object Model) tree; (3) automatic RDF-QEL output; and (4) integration of the graphical RDF query result presentation. As an example, in figure 8 we show a screen shot of the SCORM query GUI used to construct the example query illustrated in figure 6/figure 7.

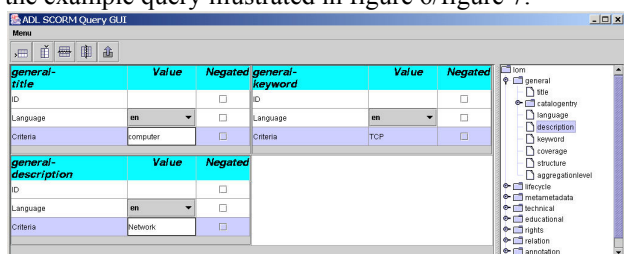


Figure 8. QBE-based SCORM query GUI

In the QBE-based SCORM query GUI, a user can firstly choose interesting SCORM metadata entries directly from the SCORM XML binding DOM tree

through drag & drop manipulation. He can then compose sets of visual templates corresponding to different SCORM metadata entries in a table according to the expected Boolean logic between the queries to construct arbitrary queries. The output of the SCORM query GUI is RDF-QEL, which is then sent to the Edutella network and expected to get query results from all SCORM metadata repositories.

### 3. Conclusions

Acknowledging the inherent advantages of RDF, e.g., the easy composability of schemas, the extendability and modularity of distributed RDF metadata, etc., Edutella adopts a fully RDF-based design to manage learning resource metadata, aiming at becoming part of the future Semantic Web. Still, we do want to use large amounts of XML-based learning resource metadata, which have been produced in E-Learning in the last years and are still populating today. In this context, our proposed approach for searching SCORM metadata in Edutella constitutes a beneficial exploration to the management of XML metadata in the Semantic Web environment.

### 4. References

- [1] ADL Technical Team, SCORM Specification V1.2, <http://www.adlnet.org>
- [2] Boag, S., D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Siméon, XQuery 1.0: An XML Query Language, <http://www.w3.org/TR/xquery/>
- [3] Chaudhri, A., R. Zicari, and A. Rashid, XML Data Management: Native XML and XML Enabled DataBase Systems, Addison-Wesley, USA, 2003.
- [4] Garcia-Molina, H., J. D. Ullman, and J. Widom, Database Systems: The Complete Book, Prentice Hall, USA, 2001.
- [5] IEEE LTSC, IEEE 1484.12.1: Draft Standard for Learning Object Metadata, <http://ltsc.ieee.org>
- [6] IMS, IMS Learning Resource Metadata Specification V1.2.1, <http://www.imsproject.org>
- [7] Nejd, W., B. Wolf, C. Qu, S. Decker, M. Stintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, Edutella: A P2P Networking Infrastructure Based on RDF, in Proc. of the 11<sup>th</sup> International World Wide Web Conference (WWW 2002), Hawaii, USA, May 2002.
- [8] Qu, C., and W. Nejd, Towards Interoperability and Reusability of Learning Resource: a SCORM-conformant Courseware for Computer Science Education, in Proc. of the 2<sup>nd</sup> IEEE International Conference on Advanced Learning Technologies (IEEE ICALT 2002), Kazan, Tatarstan, Russia, Sept. 2002.
- [9] Qu, C., W. Nejd, and H. Schinzel, Integrating Schema-specific Native XML Repositories into a RDF-based E-Learning P2P Network, in Proc. of the 2<sup>nd</sup> International Conference on Dublin Core and Metadata Applications (DC 2002), Florence, Italy, Oct. 2002.
- [10] Zloof, M. M., Query by Example: a database language, IBM Systems Journal, 16(4), 1977.