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

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

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 Edutella Network Datalog-based ECDM RDF-QEL RDF-based Query Result Exchange Format

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

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.

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

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”.

P(arg1, arg2)  P

\[ arg1 \arg2 \]

\[ head :- literal1, literal2, ..., literals \]

head literal1 literaln

Figure 6. An example Datalog query

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 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.

4. References


Figure 8. QBE-based SCORM query GUI