Update Semantics for Interoperability among XML, RDF and RDB - A Case Study of Semantic Presence in CISCO's Unified Presence Systems

Update Semantics for Interoperability among XML, RDF and RDB*

A case study of Semantic Presence in CISCO's Unified Presence Systems

Muhammad Intizar Ali[1], Nuno Lopes[1], Owen Friel[2], and Alessandra Mileo[1]

[1] DERI, National University of Ireland, Galway, Ireland
{ali.intizar,nuno.lopes,alessandra.mileo}@deri.org
[2] Cisco Systems, Galway, Ireland
{ofriel}@cisco.com

Abstract. XSPARQL is a transformation and querying language that provides an integrated access over heterogeneous data sources on the fly. It is an extension of XQuery which supports a subset of SPARQL and SQL to provide unified access over XML, RDF and RDB formats. In practical applications, data integration does not only require the integrated access over distributed heterogeneous data sources, but also the update of underlying data.
XSPARQL in its present state is only a querying and transformation language, hence lacking the update facility. In this paper, we propose an extension of the XSPARQL language with update facility. We present the syntax and semantics for this extension, and we use the real world scenario of semantic presence in CISCO's Unified Presence Systems to demonstrate the requirement of update facility. Preliminary evaluation of the *XSPARQL Update Facility* is also presented.

Keywords: XSPARQL, SPARQL , XQuery, XMPP, Updates, DML

1 Introduction

Relational databases (RDB) are the most common way of storing and managing data in majority of the enterprise environments [1]. Particularly in the enterprises where integrity and confidentiality of the data are of the utmost importance, relational data model is still the most preferred option. However, with the growing popularity of the Web and other Web related technologies (e.g. Semantic Web), various data models have been introduced. Extensible Markup Language (XML) is a very popular data model to store and specify the semi-structured data over the Web [7]. It is also considered as a de-facto data model for data exchange. XPath and XQuery are designed to access and query data in XML

* This work has been funded by Science Foundation Ireland, Grant No. SFI/08/CE/l1380 (Lion2) and CISCO Systems Galway, Ireland.

format[10, 5]. Semantic Web is built upon data represented in Resource Description Format (RDF) [4]. RDF is a standard data format for establishing semantic interpretability and machine readability among the various data sources scattered over the Web. SPARQL is a query language designed to query linked data in RDF data format [17]. All of the above mentioned data models were designed to perform a specific task and their importance in their own domains can not be denied. Several query languages including SQL, XQuery and SPARQL have been designed to access and query RDB, XML and RDF data respectively. It is inevitable for modern data integration applications to avoid such heterogeneity of data formats and query languages. Therefore, often these applications require integrated access over distributed heterogeneous data sources.

XSPARQL is a W3C Member Submission (`http://www.w3.org/Submission/2009/01/`) which combines XQuery, SPARQL and SQL to provide an integrated access over XML, RDF and RDB data sources [2]. However, in practice the integrated applications not only require the access of integrated data, but in many scenarios an update in the existing data is also desired. XSPARQL in its present state only provides Data Query Language (DQL) while lacking Data Manipulation Language (DML). In this paper we define syntax and semantics of XSPARQL Update Facility which extends the capabilities of the XSPARQL language to perform update operations over RDB, XML and RDF data on the fly while keeping the data sources in their original data formats. XSPARQL Update Facility is fully compatible with the update semantics defined individually for SQL, XQuery and SPARQL [16, 18, 12]. Our main motivation for this extension was the real world scenario of semantic presence in the CISCO Unified Presence (CUP) systems, based on processing and integrating information retrieved from XMPP(`http://xmpp.org`) messages and update the underlying data in RDB and RDF data sources to keep track of up-to-date semantic presence history of registered users. In this scenario we require not only the simultaneous access to heterogeneous data represented in RDB, XML and RDF, but also the update operations are desired to update the underlying RDB and RDF data on the fly [8].

Our main contributions in this paper can be summarised as follows: (i) we extend XSPARQL with updates, providing what we call *XSPARQL Update Facility*, (ii) we define formal semantics of the XSPRQL Update Facility, (iii) we demonstrate the need for XSPARQL Update Facility by presenting a use case scenario for semantic presence in the CUP systems, and (iv) we successfully implement and evaluate XSPARQL Update Facility for dealing with group chats history in the same scenario.

2 Semantic Presence in Cisco's Unified Presence System

Cisco Unified Presence (CUP) systems is a standards-based enterprise platform that aims to provide an effective way of communication among people in and across the organisation using instant messaging (IM) over XMPP standard protocol. A general architecture to collect several presence information from various

heterogeneous and dynamic sources has been introduced in [14],which provides some insights on how richer semantic presence services and applications can be deployed. As a first step towards the enhancement of CUP systems with richer semantic presence, a mapping of the core XMPP messages into RDF is formally defined in [8]. Figure 1 depicts a simplified high-level version of the architecture of semantic presence in the CUP systems focusing only on heterogeneous data formats that need to be manipulated during communication. All XMPP messages are mapped into RDF using various ontologies e.g SIOCC (`http://rdfs.org/sioc/chat#`), Nepomuk (`http://www.semanticdesktop.org/ontologies/2007/03/22/nmo#`) and SIOC (`http://rdfs.org/sioc/ns#`). CUP server stores history of all the group chat messages and other communications in an external PostgreSQL (`http://www.postgresql.org/`) database, while semantic enhancement component stores the semantic information in a graph database using Virtuoso Server (`http://openlinksw.com/virtuoso`). Listing 1 shows a sample XMPP message generated by IM client while requesting for the creation of a chat room. Listing 2 depicts mapping of XMPP messages into RDF using SIOC and SIOCC ontologies. We use the XSPARQL Update Facility to interpret the XMPP message and generate/update RDB and RDF data on the fly. This enables a semantically enriched CUP system to update/generate data from one data model to any other data model. Later, in section 4 we evaluate our XSPARQL Update facility by using the use case of semantic presence in CUP systems.

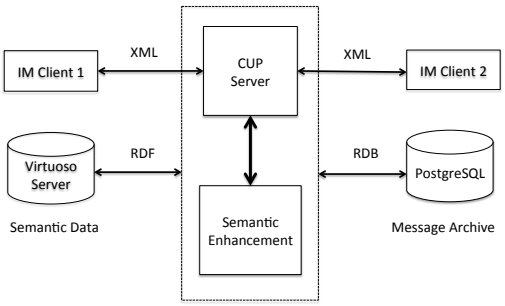

Fig. 1. Semantic Presence in CUP Systems

```
#intizar@deri.org seeks to enter
#in the DERICollab room

<presence
        from=
"intizar@deri.org/intizar-notebook"
        to=
"DERICollab@conference.corp.com/
intizar">
<x xmlns="http://jabber.org/protocol/
muc"/>
</presence>
```

Listing 1. XMPP request message for chat room creation

```
@base <http://lion2pcisco.com
/example#>.
 :DERICollab    a
sioct:ChatChannel ;
     sioc:has_owner
:intizar@corp.com    .
 :DERICollabChatSession    a
siocc:ChatSession;
     sioc:has_container
:DERICollab;
 siocc:has_publishing_source
:intizarNotebook
   siocc:has_nickname   "ali"
```

Listing 2. Mapping XMPP Message of Listing 1 into RDF [8]

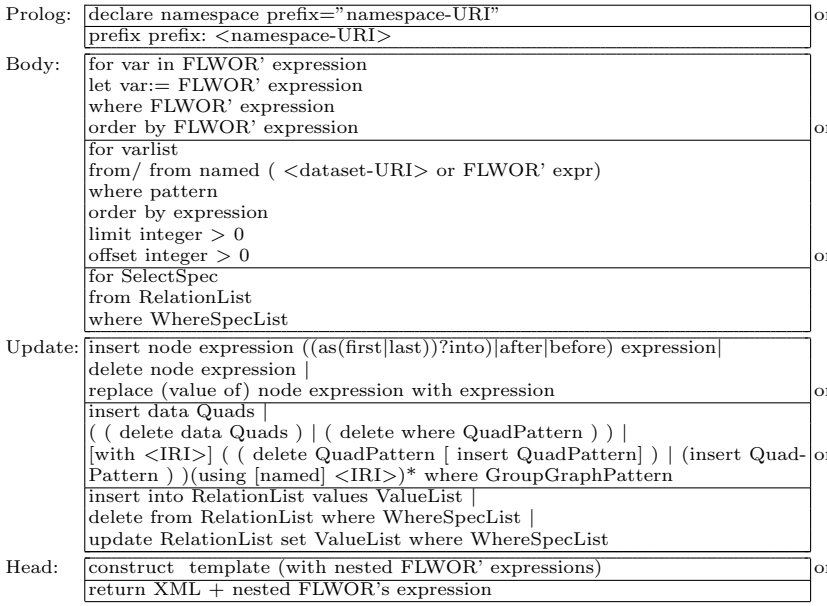

Fig. 2. Schematic View of XSPARQL with Updates

3 XSPARQL Update Facility

In this section we specify the XSPARQL Update Facility by giving a detailed description of its syntax and semantics.

3.1 Syntax

XSPARQL is built on XQuery semantics, XSPARQL Update Facility is also extended using formal semantics of XQuery Update Facility. It merges the subset of SPARQL, XQuery and SQL update clauses. We limit ourself to three major update operations of *INSERT:* to insert new records in the data set, *DELETE:* to delete already existing records from data set and *UPDATE:* to replace the existing records or their values with the new records or values. Figure 2 presents a schematic view of the XSPARQL Update Facility which allows its users to select data from one data source of any of its data format and update the results into another data source of different format within a single XSPARQL query while preserving the bindings of the variables defined within query. Contrary to the select queries, the update queries have no return type. On successful execution of the update queries no results are returned, only a response is generated which can be either *successful* upon successful execution of the query or an appropriate *error* is raised if for some reasons the XSPARQL query processor is unable to update the records successfully. However if a valid updating or delete query with or without *where* clause does not match any relevant tuple in the data source, the XSPARQL query processor will still response with successful execution, but

there will be no effect on the existing data source. The basic building block of XQuery is the expression, an XQuery expression takes zero or more XDM instances and returns an XDM instance. Following new expressions are introduced in XSPARQL Update Facility

- A new category of expression called *updating expression* is introduced to make persistent changes in the existing data.
- A basic updating expression can be any of the *insert*, *delete* or *update*.
- *for*, *let*, *where* or *order by* clause can not contain an updating expression.
- *return* clause can contain an updating expression which will be evaluated for each tuple generated by its *for* clause.

```
RevalidationDecl::=  "declare" "revalidation" ("strict" | "lax" | "skip" )
XSPARQLExpr     ::=  (FLWORExpr | SPARQLForClause | SQLForClause) |
                     (InsertClause | DeleteClause | UpdateClause)
                     (ReturnClause | ConstructClause)
InsertClause    ::=  XQueryInsert | SPARQLInsert | SQLInsert
DeleteClause    ::=  XQueryDelete | SPARQLDelete | SQLDelete
UpdateClause    ::=  XQueryUpdate | SPARQLUpdate | SQLUpdate
...
```

Fig. 3. XSPARQL Update Facility Grammar Overview

Figure 3 presents an overview of the basic syntax rules for XSPARQL Update Facility, for complete grammar of XSPARQL Update Facility, we refer our reader to the latest version of the XSPARQL available at `http://sourceforge.net/projects/xsparql`.

3.2 Semantics

Similar to the semantics of the XSPARQL language [6], for the semantics of updates in XSPARQL we rely on the semantics of the original languages: SQL, XQuery, and SPARQL. Notably, the semantics for SPARQL updates are presented only in the upcoming W3C recommendation: SPARQL 1.1 [12]. We start by presenting a brief overview of the semantics of update languages for the different data models: relational databases, XML, and RDF.

RDB: updates in the SQL language rely on a procedural language that specify the operations to be performed, respectively: $ins_{rdb}(r,t)$, $del_{rdb}(r,C)$, and $mod_{rdb}(r,C,C')$, where r is a relation name, t is a relational tuple, and C, C' are a set of conditions of the form $A = c$ or $A \neq c$ (A is an attribute name and c is a constant). Following [1], $ins_{rdb}(r,t)$ inserts the relational tuple t into the relation r, $del_{rdb}(r,C)$ deletes the relational tuples from relation r that match the condition, and $mod_{rdb}(r,C,C')$ updates the tuples in relation r that match conditions C to the values specified by C'.

XML: For XML data, the XQuery language is adapted such that an expression can return a sequence of nodes (as per the XQuery semantics specification [9]) or *pending update list* [18], which consists of a collection of *update*

primitives, representing state changes to XML nodes. For this paper we are focusing on *insert*, *delete*, and *replace* operations, whose semantics are procedurally defined in [18] and in this paper we represent these procedural semantics by the functions $ins_{xml}(SourceExpr, InTargetChoice, TargetExpr)$, $del_{xml}(TargetExpr)$, and $mod_{xml}(TargetExpr, ExprSingle)$.

RDF: SPARQL Update's semantics are similarly defined in terms of changes to the underlying triple store [12], where the result of an update operation is a new triple store. We rely on the semantics functions ins_{rdf}, del_{rdf}, and mod_{rdf} (shorthand for the functions *OpInsertData*, *OpDeleteData*, and *OpDeleteInsert*, respectively). The functions signatures are as per the SPARQL specification: $ins_{rdf}(GS, QuadPattern)$, $del_{rdf}(GS, QuadPattern)$, and $mod_{rdf}(GS, DS, QuadPattern_{del}, QuadPattern_{ins}, P)$ presented in [12]. The functions ins_{rdf} and del_{rdf} insert and delete triples matching *QuadPattern* from the triple store DS, respectively. The mod_{rdf} evaluates the graph pattern P against the dataset DS and apply these bindings to $QuadPattern_{del}$ in order to determine the triples to be removed from the graph store GS and to $QuadPattern_{ins}$ for the inserted triples.

4 Implementation and Evaluation

Implementation: We use java platform for the implementation of the XSPARQL Update Facility. XSPARQL grammar rules are defined using ANTLR (`http://www.antlr.org`). XSPARQL uses Saxon (`http://www.saxonica.com`) as a query processor for XQuery, Jena ARQ (`http://jena.apache.org`) for SPARQL queries and for SQL queries we provide option to connect to multiple relational database management systems including PostgreSQL and MySQL (`http://www.mysql.com`).

Evaluation of XSPARQL Update Facility: In order to demonstrate and evaluate performance and practicality of XSPARQL Update Facility, we consider a generic use case of group chat of the semantic presence in CUP systems as described in Section 2. However XSPARQL Update Facility can be used by any application which requires access and updates in distributed heterogeneous data stored in any of the RDB, XML or RDF data models.

```
for $x in //*[name='presence']
if $x/status[@code=201] then
insert data
 {Graph <SemanticPresence>
       {
         :DERICollab    a    sioct:ChatChannel ;
         sioc:has_owner    :intizarali@corp.com .
         :DERICollabChatSession    a    siocc:ChatSession;
         sioc:has_container    :DERICollab;
         siocc:has_publishing_source    :intizaraliNotebook
         siocc:has_nickname    "ali"
      }}
```

Listing 3. A Sample XSPARQL Update Query

A user creates a chat group: Consider a scenario where a user creates a persistent chat room over a specific topic as shown in Listing 1 in the Section 2. When an IM client sends a request to create a chat room an XMPP message will be generated by IM client and sent to the CUP Server. CUP Server will process the message. The semantic presence component of CUP systems wants to update the semantic data stored in Virtuoso Server. Listing 3 shows an XSPARQL query that interprets XMPP messages using XQuery and if certain condition is met, the XSPARQL update query will insert the relevant information into RDF data store using SPARQL update query.

5 Related Work

Several efforts are made to integrate distributed XML, RDF and RDB data on the fly. These approaches can be divided into two types, (i) *Transformation Based:* Data stored in various format is transformed into one format and can be queried using a single query language [11]. The W3C GRDDL working group addresses the issues of extracting RDF from XML documents. (ii) *Query Rewriting Based:* Query languages are used to transform or query data from one format to another format. SPARQL queries are embedded into XQuery/XSLT to pose against pure XML data [13]. XSPARQL was initially designed to integrate data from XML and RDF and later extended to RDB as well. DeXIN is another approach to integrate XML, RDF and RDB data on the fly with more focus on distributed computing of heterogeneous data sources scattered over the Web [3]. Realising the importance of the updates, the W3C has recommendations for XQuery and SPARQL updates [18, 12]. In [15], SPARQL is used to update the relational data. However, to the best of our knowledge their is no work available which can provide simultaneous access over the distributed heterogeneous data source with updates.

6 Conclusion and Future Work

In this paper we have extended XSPARQL to provide the Update Facility which further strengthens the capabilities of XSPARQL by enabling simultaneous access and update of heterogeneous data sources. We have defined syntax and semantics of XSPARQL Update Facility, which merges update operations of SQL, XQuery and SPARQL. Using XSPARQL Update Facility user not only can query, integrate and transform heterogeneous data on the fly, but can also update the already stored data in the data sets or in-memory data generated as a result of a query before an update operation. XSPARQL Update Facility will lead to the wide adoption of XSPARQL in many applications (e.g. similar to semantic presence in CUP systems) which require integrated access over distributed heterogeneous data sources with updates.

In the future, we plan to further investigate query optimisation techniques that would make it possible to evaluate even complex XSPARQL queries in tractable time, that would make XSPARQL usable for large-scale applications.

In this paper, we elaborated the practicality of XSPARQL Update Facility using real world scenario, however in future, we plan to define detailed formal semantics of the language, evaluation of static and dynamic rules and experimental evaluation considering various parameters including scalability and query executaon time. We also plan to extend XSPARQL Update Facility to include Data Definition Language (DDL) for its subsequent languages.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. W. Akhtar, J. Kopecký, T. Krennwallner, and A. Polleres. XSPARQL: Traveling between the XML and RDF Worlds - and Avoiding the XSLT Pilgrimage. In *Proc. of ESWC 2008*, pages 432–447, 2008.
3. M. I. Ali, R. Pichler, H. L. Truong, and S. Dustdar. DeXIN: An Extensible Framework for Distributed XQuery over Heterogeneous Data Sources. In *Proc. of ICEIS 2009*, LNBIP, pages 172–183. Springer, 2009.
4. D. Beckett and B. McBride. RDF/XML Syntax Specification (Revised), Feb. 2004. W3C Proposed Recommendation.
5. A. Berglund, S. Boag, D. Chamberlin, M. F. Fernández, M. Kay, J. Robie, and J. Siméon. XML Path Language (XPath) 2.0, Dec. 2010. W3C Recommendation.
6. S. Bischof, S. Decker, T. Krennwallner, N. Lopes, and A. Polleres. Mapping between RDF and XML with XSPARQL. *Journal on Data Semantics*, 1:147–185, 2012.
7. T. Bray, J. Paoli, E. Maler, F. Yergeau, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 (Fifth Edition), Nov. 2008. W3C Recommendation.
8. M. Dabrowski, S. Scerri, I. Rivera, and M. Leggieri. Dx– Initial Mappings for the Semantic Presence Based Ontology Definition, Nov. 2012. http://www.deri.ie/publications/technical-reports/.
9. D. Draper, P. Fankhauser, M. Fernández, A. Malhotra, K. Rose, M. Rys, J. Simeón, and P. Wadler. XQuery 1.0 and XPath 2.0 Formal Semantics, Jan 2007. W3C Recommendation.
10. M. F. Fernández, D. Florescu, S. Boag, J. Robie, D. Chamberlin, and J. Siméon. XQuery1.0: An XML query language, Apr. 2009. W3C Proposed Recommendation.
11. F. Gandon. GRDDL Use Cases: Scenarios of extracting RDF data from XML documents, Apr. 2007. W3C Proposed Recommendation.
12. P. Gearon, A. Passant, and A. Polleres. SPARQL 1.1 Update, Jan. 2012. W3C Working Draft.
13. S. Groppe, J. Groppe, V. Linnemann, D. Kukulenz, N. Hoeller, and C. Reinke. Embedding SPARQL into XQuery/XSLT. In *Proc. of SAC*, 2008.
14. M. Hauswirth, J. Euzenat, O. Friel, K. Griffin, P. Hession, B. Jennings, T. Groza, S. Handschuh, I. P. Zarko, A. Polleres, and A. Zimmermann. Towards Consolidated Presence. In *Proc. of CollaborateCom 2010*, pages 1–10, 2010.
15. M. Hert, G. Reif, and H. Gall. Updating relational data via SPARQL/update. In *Proc. of EDBT/ICDT Workshops*, 2010.
16. M. Negri, G. Pelagatti, and L. Sbattella. Formal Semantics of SQL Queries. *ACM Trans. Database Syst.*, 16(3):513–534, 1991.
17. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF, Jan. 2008. W3C Proposed Recommendation.
18. J. Robie, D. Chamberlin, M. Dyck, D. Florescu, J. Melton, and J. Siméon. XQuery Update Facility 1.0, Mar. 2011. W3C Recommendation.