ONTOMS2: an Efficient and Scalable ONTOlogy Management System with an Incremental Reasoning

Min-Joong Lee[1], Jong-Ryul Lee[1], Sangyeon Kim[1], Myung-Jae Park[1], and Chin-Wan Chung[2]

Department of Computer Science , KAIST, Daejeon, Republic of Korea
{mjlee,jrlee,sangyeon,jpark}@islab.kaist.ac.kr[1] chungcw@kaist.edu[2]

Abstract. We present ONTOMS2, an efficient and scalable ONTOlogy Management System with an incremental reasoning. ONTOMS2 stores an OWL document and processes OWL-QL and SPARQL queries. Especially, ONTOMS2 supports SPARQL Update queries with an incremental instance reasoning of inverseOf, symmetric and transitive properties.

1 Introduction

In order to efficiently manage ontology data, various ontology data management systems [4–6] are proposed based on RDBMS. However, existing ontology data management systems do not support updates incrementally. To efficiently manage such large sized ontology data, we need a way of incremental updating for ontology data. We propose an incremental update strategy to efficiently handle a large amount of ontology data and the frequent updates of such ontology data. Our incremental update strategy provides an insertion and deletion based on SPARQL Update with the support of some important semantics of ontologies such as inverseOf, symmetric, and transitive. We apply our incremental update strategy to ONTOMS which is an efficient and scalable ONTOlogy Management System proposed in [6].

ONTOMS efficiently manages large sized OWL data based on RDBMS and using OWL-QL. It stores OWL data into a class based relational schema to increase the query processing performance. Figure 1 describes an example of OWL data stored into relational tables in ONTOMS. Unlike other approaches, ONTOMS generates a class based relational schema, where one relation is created for each class. Each class relation contains associated properties as its attributes. For more details of ONTOMS, please refer [6].

We denote the new version of ONTOMS where our incremental update strategy and SPARQL processor are applied as ONTOMS2. We designed the architecture of ONTOMS2 as depicted in Figure 2. The core modules of ONTOMS2 are OWL Data Storage Module, Instance Inference Module, SPARQL Module, and OWL-QL Module. To apply our incremental update strategy to ONTOMS, we create SPARQL Module in Query Processing Module and modified Instance Inference Module for the incremental reasoning. OWL Data Storage Module obtains class and property hierarchy information from OWL Reasoner, Pellet, and

GraduateStudent

UID	degreeFrom	degreeFrom_S	degreeFrom_E
GS1			

Professor

UID	degreeFrom	degreeFrom_S	degreeFrom_E
Prof1	Univ1	6	7

Course

UID
C1
C2

GraduateStudent_ takesCourse

UID	Value
GS1	C1
GS1	C2

Professor_ teachesCourse

UID	Value
Prof1	C1
Prof1	C2

Fig. 1. Relational Tables in ONTOMS

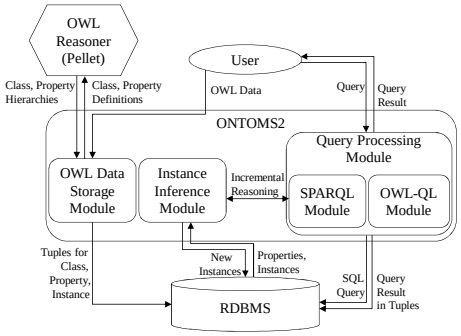


Fig. 2. The Architecture of ONTOMS2

generates relationship information among classes and properties when parsing the given OWL data. Once OWL data is stored, Instance Inference Module performs instance reasoning with properties which are collected along with their values from RDBMS, and stores newly generated instances into RDBMS. The details of instance reasoning for initial OWL data can be found at Section 6 in [6]. The Query Processing Module processes OWL-QL query and SPARQL query. When a SPARQL update query is given, SPARQL Module processes it through Instance Inference Module for the incremental reasoning.

2 Ontology Data Update

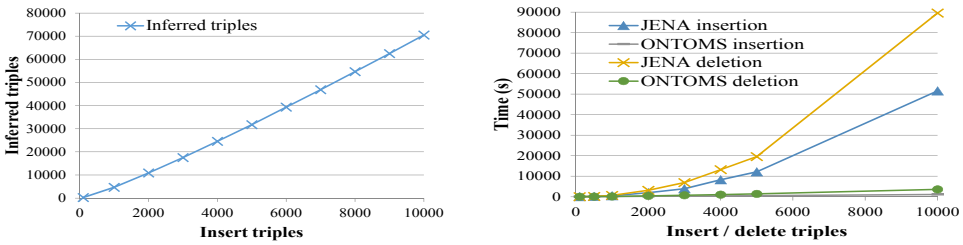
SPARQL Update There are several query languages for OWL such as OWL-QL and SQWRL . However, all of them support a read-only(select) query only. The previous version [6] of ONTOMS2 uses a OWL-QL to retrieve instances. We enhanced ONTOMS to have an ability to update ontology data by adding the SPARQL processor in addition to the OWL-QL processor. This is because OWL is developed as a vocabulary extension of RDF, SPARQL is a de facto query language for RDF, and recently W3C published a recommendation [2] for the SPARQL update.

According to the SPARQL Update recommendation proposed by W3C, SPARQL Update provides three operations related to update : INSERT DATA, DELETE DATA, and DELETE/INSERT. The INSERT DATA operation is to add new triples into the ontology data while the DELETE DATA operation is to remove triples from the ontology data. Lastly, the DELETE/INSERT operation is to remove triples and add new triples into the ontology data with the WHERE clause. ONTOMS2 supports all INSERT DATA, DELETE DATA, and DELETE/INSERT operations. Due to the space limitation, we omitted the detailed syntax. Please refer Chapter 3 SPARQL 1.1 Update Language in [2] for the detailed syntax.

Incremental Reasoning Only some of existing ontology data management systems support the update for ontology data. Even though there are some existing systems with the update feature, the instance reasoning for the updates has to be conducted from the scratch due to the constraints of the system while ONTOMS supports an incremental reasoning. Thus, the instance reasoning of the existing systems for each update is performed by processing all the stored triples. Eventually, it degrades the update processing performance.

OWL defines several types of properties. However, only inverseOf, symmetric and transitive properties may generate new facts(triples). The incremental reasoning for the inverseOf and symmetric properties can be done in a straight forward way. For the insertion and deletion of transitive property triples, we adapt the SQL-based transitive closure maintenance algorithm presented in [3] to effectively maintain the transitive properties. We cut down the step for finding truly new tuples among the generated tuples by unifying the transitive closure table and multi-value class property table. This reduces costly table join operations and it also reduces a storage size.

3 Experiments



(a) The number of inferred triples (b) The update processing time

Fig. 3. Experimental results

The most important enhanced point of ONTOMS2 over ONTOMS is that ONTOMS2 supports an efficient ontology data update with an incremental reasoning. Therefore, we focused on the incremental reasoning performance when ontology data is updated. We compare ONTOMS2 with JENA which is one of the most popular ontology data management systems. JENA provides the SPARQL update feature through ARQ. However, ARQ only supports the instance reasoning with non-update(SELECT) queries[1]. Therefore, we implemented a simple SPARQL update interface for JANA which uses OntModel. Among reasoners in JENA, $OWL\_MEM\_MICRO\_RULE\_INF$ is used.

Note that, in JENA, the instance reasoning should be re-run from the scratch for each update query and the update processing time for JENA does not contain the time for storing the results of the instance reasoning into the relational tables while ONTOMS2 does the incremental reasoning and the update processing time for ONTOMS2 contains the storing time. Our experiments were performed on 2.27GHz Intel Xeon with 12GB of main memory. We implemented ONTOMS2 using MS SQL Server 2008 and Java. For JENA [1], Jena-2.6.4 version is used.

Figure 3(a) shows the number of triples to be inferred for various sizes of insertion triples(facts). All fact triples have same transitive property. New triples to be inserted as a facts are generated such that their subjects and objects are randomly selected from subjects and objects of previously inserted fact triples.

[1] ARQ supports a SPARQL update query on basic Model only, not on OntModel or InfModel. However, JENA uses OntModel or InfModel for the reasoning.

In JENA, the instance reasoning for each insertion is conducted from the scratch while the instance reasoning of ONTOMS2 is incrementally conducted. As a result, ONTOMS2 outperforms JENA for the insertion and deletion due to the incremental update strategy. The results are as shown in Figure 3(b).

4 Demo

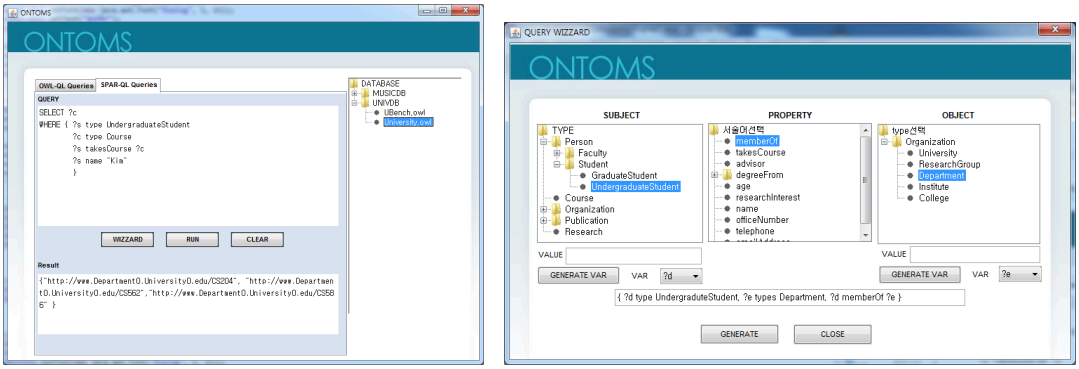


Fig. 4. Query interface and Query wizard of ONTOMS2

During the demonstration, we will illustrate how our system handles SPARQL queries over ontology data such as retrieving instances of classes or properties, inserting/removing instances which satisfy a specific condition with an incremental reasoning. Our system also contains a GUI-based wizard to help a user to create a SPARQL query easily. The screen shots of ONTOMS2 are depicted in Figure 4 and the recorded video can be found at our demo page (http://islab.kaist.ac.kr/ONTOMS2/).

Acknowledgments This work was supported by the National Research Foundation of Korea grant funded by the Korean government (MSIP) (No. NRF-2009-0081365).

References

1. Jena - a semantic web framework for java. http://jena.sourceforge.net/index.html/.
2. Sparql 1.1 update, w3c recommendation 2013. http://www.w3.org/TR/sparql11-update/.
3. G. Dong, L. Libkin, J. Su, and L. Wong. Maintaining the transitive closure of graphs in sql. *Int. J. Information Technology*, 5:46–78, 1999.
4. S. Kang, J. Shim, and S. goo Lee. Tridex: A lightweight triple index for relational database-based semantic web data management. *Expert Syst. Appl.*, 40(9):3421–3431, 2013.
5. Z. Pan, X. Zhang, and J. Heflin. Dldb2: A scalable multi-perspective semantic web repository. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 1, pages 489–495. IEEE, 2008.
6. M.-J. Park, J. Lee, C.-H. Lee, J. Lin, O. Serres, and C.-W. Chung. An efficient and scalable management of ontology. In *Proceeding of DASFAA '07*, pages 975–980, 2007.