An Efficient and Scalable Management of Ontology

Myung-Jae Park[1], Jihyun Lee[1], Chun-Hee Lee[1], Jiexi Lin[1], Olivier Serres[2], and Chin-Wan Chung[1]

[1] Korea Advanced Institute of Science and Technology, Korea
{jpark,hyunlee,leechun,jesse,chungcw}@islab.kaist.ac.kr
[2] University of Technology of Belfort-Montbéliard, France
olivier.serres@utbm.fr

Abstract. OWL is a recommended language for publishing and sharing ontologies on the Semantic Web. To manage the ontologies, several OWL data management systems have been proposed. However, the existing systems have limitations of the scalability and the reasoning. In this paper, we propose an OWL data management system, ONTOMS, which stores OWL data into class based relations, performs complete inverseOf, symmetric, and transitive reasoning for instances, and efficiently evaluates OWL-QL queries against ontologies in a relational database.

1 Introduction

Web Ontology Language(OWL) [1] is a semantic markup language for publishing and sharing ontologies on the Web. OWL is developed as a vocabulary extension of RDF [2] and RDFS [3] to increase the expressive power of ontology data, which leads OWL as a recommended ontology language for the Semantic Web. OWL data can be represented by a graph like RDF as shown in Fig. 1.

To support the expressive power of OWL data, several OWL reasoners [4–6] have been proposed. However, those reasoners confronted the scalability issue, due to the use of memory. To overcome this problem, RDBMS based OWL data management systems [7–9] have been proposed. Since RDBMSs do not support the reasoning ability, those systems cannot obtain complete class and property hierarchies and perform the instance reasoning. As a result, those systems incorporated OWL reasoners to obtain such hierarchies completely. However, due to the scalability drawback of OWL reasoners, instance reasoning is not supported.

To retrieve instances of classes or properties, OWL Query Language(OWL-QL) [10] has been proposed. OWL-QL is based on query patterns, in the form of (property, subject, object). To evaluate query patterns over OWL data stored in RDBMS, a proper relational schema is required. However, existing systems do not incorporate efficiency consideration in designing their relational schemas. Thus, in this paper, we propose ONTOMS, an efficient and scalable ONTOlogy Management System, to efficiently manage large sized OWL data. ONTOMS stores OWL data into a class based relational schema to increase query processing performance. On the average, the query performance of ONTOMS is about

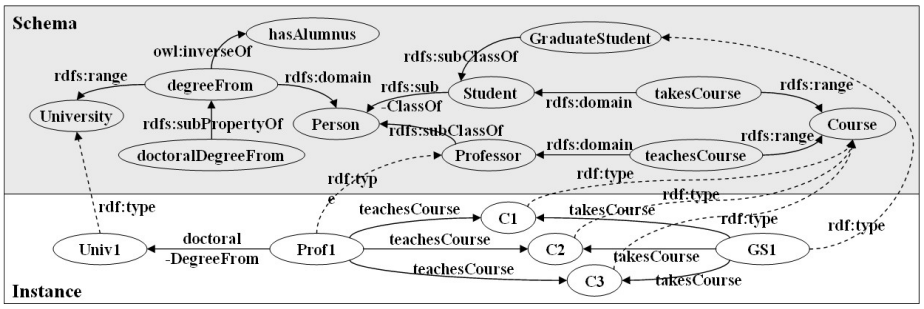

Fig. 1. An example of OWL data (a simple university ontology)

90 times better than DLDB [7]. To provide the complete results, ONTOMS supports instance reasoning for inverseOf, symmetric, and transitive properties. To our best knowledge, ONTOMS is the first RDBMS based OWL data management system which supports the complete instance reasoning.

2 Related Work

There are several OWL reasoners to manage OWL data. FaCT [5] performs class and property related reasoning only. RACER [4] and Pellet [6] support class and property hierarchy reasoning, as well as instance reasoning.

SnoBase [9] stores class and property definitions (e.g., subClassOf and subPropertyOf) into Fact relation. SnoBase also stores every triple (i.e., (subject, property, object)) into Fact relation. To provide reasoning, SnoBase utilizes SQL triggers. However, the runtime depth level of trigger cascading supported in RDBMSs is limited. Also, SnoBase does not support instance reasoning.

Instance Store [8] uses Descriptions relation to store class definitions, Primitives relation to store individuals, and other four relations (Type, Equivalents, Parents and Children) to maintain class hierarchy information. Instance Store uses FaCT or RACER to only obtain class hierarchies. In addition, Instance Store only supports classes without any consideration on properties.

DLDB [7] maintains one class relation for each class and one property relation for each property. For class and property hierarchies, DLDB uses FaCT. However, DLDB does not support any instance reasoning. Thus, DLDB cannot provide complete query results for properties which require instance reasoning.

3 OWL Data Storage

The class hierarchy and the property hierarchy are generated from Pellet. To maintain containment relationship information among nodes of each hierarchy, a pair of (start, end) values is assigned to each node according to the node's position, which was originally proposed for XML data [11]. Fig. 2 shows class and property hierarchies for the OWL data in Fig. 1.

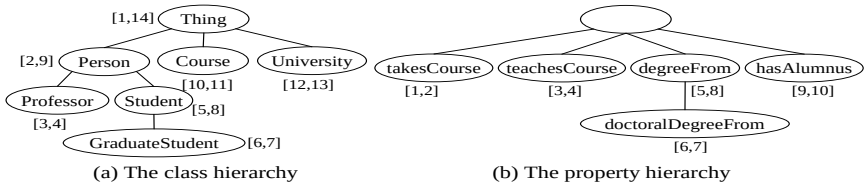

Fig. 2. Class and Property Hierarchies

ONTOMS generates a class based relational schema, where one relation is created for each class. Each class relation contains associated properties as attributes. Associated properties are the properties that have the class as domains.

To efficiently utilize the property hierarchy, we only retain properties which do not have any super properties (called highest properties) among associated properties. Class relations also have start and end attributes for each highest property. If the highest property has no subproperties, those are omitted.

Student

UID	takesCourse	degreeFrom	degreeFrom_S	degreeFrom_E

Course

UID

GraduateStudent_takesCourse

UID	Value
GS1	C1
GS1	C2
GS1	C3

Professor_teachesCourse

UID	Value
Prof1	C1
Prof1	C2
Prof1	C3

Person

UID	degreeFrom	degreeFrom_S	degreeFrom_E

University

UID	hasAlumnus
Univ1	

GraduateStudent

UID	degreeFrom	degreeFrom_S	degreeFrom_E
GS1			

Professor

UID	degreeFrom	degreeFrom_S	degreeFrom_E
Prof1	Univ1	6	7

Fig. 3. Relational Tables in ONTOMS

A number of redundant tuples would be generated for a multi-valued property. The multi-valued property indicates where one instance of property's domain has more than one values. For example, in Fig. 1, Prof1 has three different values (i.e., C1, C2, and C3) for teachesCourse property. As a result, ONTOMS separates multi-valued properties from class relations. A new relation is assigned to each multi-valued property. Fig. 3 shows the relations[3], generated for the OWL data presented in Fig. 1. Note that ONTOMS stores new tuples generated after performing instance reasoning. The (Univ1, null) tuple in University relation should be updated as (Univ1, Prof1) for the inverseOf property, hasAlumnus.

Note that the translation process of OWL-QL queries to SQL queries for the class based relations can be found in [12].

4 Instance Reasoning

OWL defines five types of properties: inverseOf, symmetric, transitive, functional and inverseFunctional properties. Only the first three properties may generate a

[3] Internally, ONTOMS assigns a unique label identifier (UID) to each instance.

large number of new facts[4]. Therefore, we focus on reasoning for inverseOf, symmetric and transitive properties (which we will refer to as IST properties). Note that the definitions of the IST properties are given in the OWL Reference [1].

Definition 1. *Relation R of a property P is the set of (x,y) in P.*

Definition 2. *Let property P be inverseOf property P', R be the relation of P, and R' be the relation of P'. The inverseOf reasoning for P or R is the process of adding (y,x) to R for all (x,y) in R', if (y,x) is not in R.*

Definition 3. *Let P be a symmetric property and R be the relation of P. The symmetric reasoning for P or R is the process of adding (y,x) to R for all (x,y) in R, if (y,x) is not in R.*

Definition 4. *Let P be a transitive property and R be the relation of P. The transitive reasoning for P or R is the process of computing the transitive closure of R.*

A relation R after inverseOf reasoning is written as R^I.[5] Similarly, a relation R after symmetric and transitive reasoning is written as R^S and R^T, respectively.

In this paper, we propose an IST reasoning algorithm which does not require recursive or iterative processing. The algorithm guarantees that the complete set of new facts can be obtained by performing reasoning only once for each type of property in a certain sequence, i.e., first for inverseOf property, then for symmetric property, and last for transitive property.

Lemma 1. *Suppose relation R is inverseOf relation R'. After symmetric reasoning for R and R', R^S is inverseOf R'^S.*

Lemma 2. *Suppose relation R is inverseOf relation R'. If R is symmetric, then R' is symmetric. If R is transitive, then R' is transitive.*

Theorem 1. *Suppose property P is inverseOf property P', P is symmetric and transitive. Let R be the relation of P and R' the relation of P'. By following the sequence <inverseOf reasoning, symmetric reasoning, transitive reasoning> for R and R', the resulting relations of R and R' are inverseOf of each other, symmetric and transitive.*

The proof for Theorem 1 and the algorithm for IST reasoning, which can be found in [12], are not included due to the page limitation.

5 Experiments

We implemented ONTOMS using IBM DB2 UDB 8.2. We interfaced DLDB with IBM DB2 since DLDB uses MS-Access. Experiments were performed on 3GHz

[4] Referred to as newly generated instances in Sect. 3.

[5] Here,we introduce R^I to indicate the change of R as a result of inverseOf reasoning.

Pentium 4 with 1024MB of main memory. We used Lehigh University Benchmark Data(LUBM)[6]. We generated various sizes of OWL data: 1MB, 5MB, 10MB, 50MB, 100MB, and 500MB. Since LUBM queries (Q1 to Q14) are not sufficient, we added three queries (Q15 to Q17). Detailed information on the query set, which can be found in [12], is not included due to the page limitation.

We compared the total query processing time of ONTOMS and that of DLDB using 17 queries. We show the total query processing time for only 50MB data in Fig. 4 since the shapes of graphs for different sizes are similar.

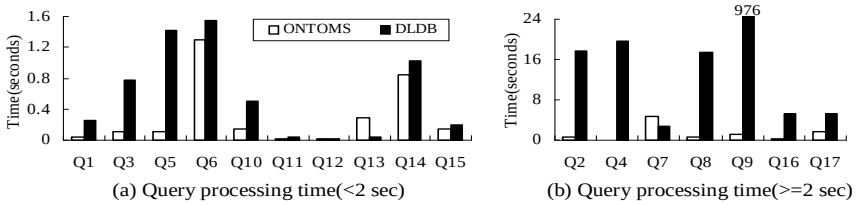

Fig. 4. Query Processing Time for 50MB OWL Data

The number of joins in ONTOMS is less than or equal to that of DLDB. Therefore, in Fig. 4, ONTOMS is better than DLDB for most of 17 queries. However, for Q7 and Q13, ONTOMS is worse than DLDB. Since Q7 and Q13 have values as their subjects, there are just a few bindings satisfying those queries.

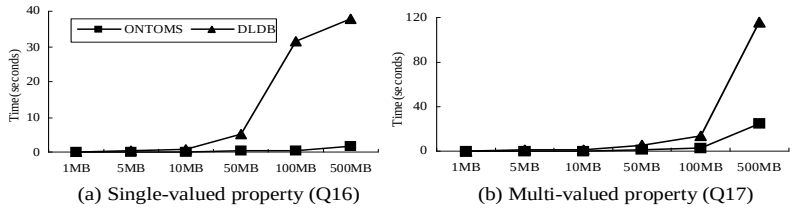

Fig. 5. Query Processing Time over Differently Sized OWL Data

In Fig. 5, all properties of Q16 are single-valued properties while those of Q17 contain multi-valued properties. Thus, the performance gap between ONTOMS and DLDB is much larger in Q16.

Consequently, ONTOMS outperforms DLDB for most queries in spite of its support of instance reasoning. ONTOMS is 90 times faster than DLDB on the average, calculated by averaging performance differences for queries over 1MB, 5MB, 10MB, 50MB, 100MB, and 500MB OWL data.

[6] Available in http://swat.cse.lehigh.edu/projects/lubm/index.htm

6 Conclusion

In this paper, we proposed ONTOMS, an OWL data management system using an RDBMS. ONTOMS generates the class based relational schema in which a relation is created for each class and contains associated properties as its attributes. To avoid data redundancy, ONTOMS creates class-property relations for multi-valued properties. Thus, this schema is of great advantage to queries having less multi-valued properties. In addition, ONTOMS supports the reasoning on the IST properties and the class and property hierarchies. The experimental results show that ONTOMS outperforms DLDB in the query response time.

Acknowledgments. This research was supported by the Ministry of Information and Communication, Korea, under the College Information Technology Research Center Support Program, grant number IITA-2006-C1090-0603-0031.

References

1. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. W3C Recommendation, http://www.w3.org/TR/owl-ref (Feb. 2004)
2. Manola, F., Miller, E., McBride, B.: RDF Primer. W3C Recommendation, http://www.w3.org/TR/rdf-primer (Feb. 2004)
3. Brickley, D., Guha, R.V., McBride, B.: RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema (Feb. 2004)
4. Haarsley, V., Moller, R.: RACER System Description. In: Proc. of 1st International Joint Conference on Automated Reasoning. (June 2001) 701–706
5. Horrocks, I., Sattler, U.: A Tableaux Decision Procedure fore SHOIQ. In: Proc. of 19th International Joint Conference on Artificial Intelligence. (Aug. 2005) 448–453
6. Parsia, B., Sirin, E.: Pellet: An OWL DL Reasoner. In: Proc. of 3rd International Semantic Web Conference. (Nov. 2004)
7. Guo, Y., Pan, Z., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In: Proc. of 3rd International Semantic Web Conference. (Nov. 2004) 274–288
8. Horrocks, I., Li, L., Turi, D., Bechhofer, S.: The Instance Store: Description Logic Reasoning with Large Numbers of Individuals. In: Proc. of 2004 International Workshop on Description Logic. (June 2004) 31–40
9. Lee, J., Goodwin, R.: Ontology Management for Large-Scale Enterprise Systems. IBM Technical Report, RC23730 (Sep. 2005)
10. Fikes, R., Hayes, P., Horrocks, I.: OWL-QL-A Language for Deductive Query Answering on the Semantic Web. Journal of Web Semantics **2**(1) (Dec. 2004) 19–29
11. Zhang, C., Naughton, J., DeWitt, D., Luo, Q., Lohman, G.: On Supporting Containment Queries in Relational Database Management Systems. In: Proc. of the 2001 ACM SIGMOD Conference. (May 2001) 425–436
12. Park, M.J., Lee, J.H., Lee, C.H., Lin, J., Serres, O., Chung, C.W.: ONTOMS: An Efficient and Scalable Ontology Management System. In: KAIST CS/TR-2005-246. (Dec. 2005)