Persistent Storage System for Efficient Management of OWL Web Ontology

Dongwon Jeong[1], Myounghoi Choi[1], Yang-Seung Jeon[2], Youn-Hee Han[3], Laurence T. Yang[4], Young-Sik Jeong[2], and Sung-Kook Han[2]

[1] Department of Informatics and Statistics, Kunsan National University, Gunsan, Jeollabuk-do, 573-701, Korea
{djeong, cmh775}@kunsan.ac.kr
[2] Department of Computer Engineering, Wonkwang University, Iksan, Jeollabuk-do, 570-749, Korea
{globaljeon, ysjeong, skhan}@wku.ac.kr
[3] School of Internet-Media, Korea University of Technology and Education, Cheonan, Chungnam, 330-708, Korea
yhhan@kut.ac.kr
[4] Department of Computer Science, St. Francis Xavier University, Antigonish, NS, B2G 2W5, Canada
lyang@stfx.ca

Abstract. This paper proposes a new persistent storage to efficiently manage OWL Web ontologies. The Semantic Web is recognized as a next direction for progress of the current Web. To realize the Semantic Web, various technologies have been developed. Especially, OWL (Web Ontology Language) is the most important and state-of-the-art technology toward the ideal Semantic Web. However, most of research is being focused on designing and building OWL documents (OWL Ontology). One of the most important issues is how to efficiently and persistently store a very large OWL data into database management systems. In this paper, we propose a persistent storage system to achieve the purpose. This paper describes the performance evaluation result on the load time. The experiment result explicitly shows that our proposal provides more enhanced performance comparing with the existing systems.

1 Introduction

OWL (Web Ontology Language) is recognized as a next technology for the Semantic Web. Even though various technologies (For example, XML, RDF, RDF Schema, OIL, and DAML) have been developed [1, 2, 3, 4], we are faced with several issues such as no reasoning function and high implementation complexity. For resolving these problems, OWL has been proposed as an international standard of W3C [5]. Much research is concentrated on OWL-based Web ontology tools or systems for building, editing, displaying, and so on.

* This work was supported in part by MIC & IITA through IT Leading R&D Support Project.

One of the most important issues is to efficiently load OWL ontologies into a specific permanent management system such as relational database management systems. Generally, we can store OWL data using file systems or database management systems. In case of the file system, an OWL document is stored as a plain text. However, this approach has many disadvantages in management and performance aspects. The other approach is to use databases such as the relational database or object-oriented database. Especially, most OWL storage systems are based on the relational database, because it is widely used for data management and is stable through improvement over several decades.

Permanent OWL ontology storage systems (including APIs) using relational databases have been developed and the representative storage system includes Sesame-DB, DLDB-OWL, and Jena [6, 7, 8, 9, 10]. However, these systems have several issues to deal with very large OWL data sets (OWL ontologies). Most systems are based on the triple (3-tuple) model. In other words, their load processing is time-consuming and requires high response time. Therefore, we cannot fully use many advantages of the relational database model.

In this paper, we propose a new enhanced persistent storage model to efficiently load a very large OWL ontology. To show prominence of our proposal, comparative evaluations are described.

2 Definition of Persistent Storage System

2.1 Framework

Fig. 1 depicts the framework for the persistent storage system.

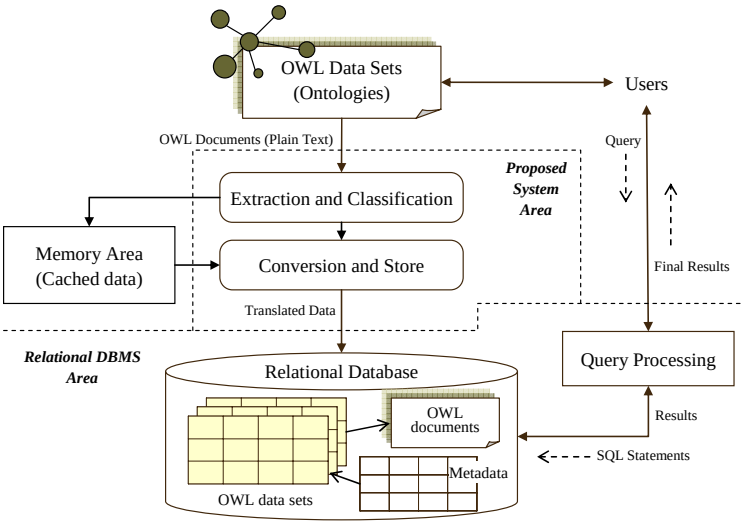

Fig. 1. Framework of the proposed storage system

2.2 Permanent Storage System Model Definition

PSSM denotes the persistent storage system proposed in this paper and is defined as Definition 1.

Definition 1. PSSM = (I, OP, MDM, PDM),
where - I: OWL ontology (Plain text files)
- OP: Operations for extraction, classification, mapping and store
- MDM: Memory data structure for the classified OWL elements
- PDM: Persistent data model

Our system conceptually consists of five components: Input OWL documents, operations, memory data structure, and relational data model. More details are described in Section 3.

Definition 2. PDM, denoting the relational database model, is defined as follows.
PDM = (RDM, Meta, Plain),
where - RDM means the defined relational data model for OWL data (i.e., classes and instances)
- Meta indicates meta-information such as ontology type
- Plain is the set of original OWL documents

To manage OWL ontologies, the relational database needs to accept actual ontology (classes and instances), metadata such as ontology type and its source file, and original OWL files (plain text type).

Fig. 2 illustrates a metamodel to store OWL data sets into relational databases. The defined relational database model is composed of three parts. The first part is to store classes and instances. OWL specification names both Class and Individual respectively. The meta-information part is to manage header information, full prefix names, and relevant OWL sources. The final part is the set of OWL document files.

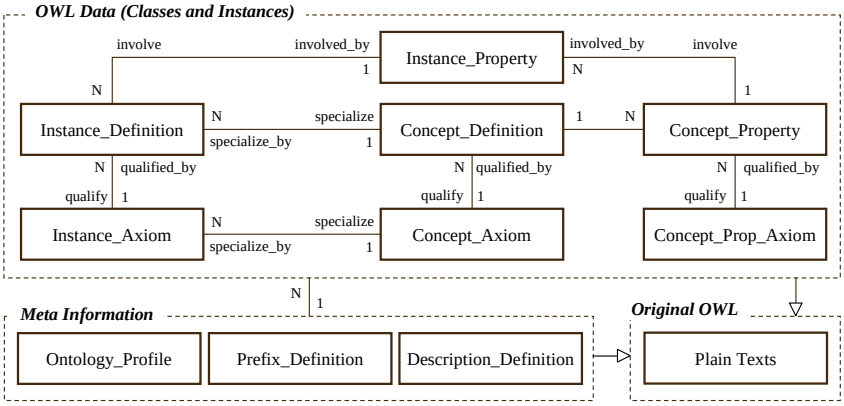

Fig. 2. Metamodel for the relational schema structure

3 System Design for Implementation

This section describes the system architecture for implementation of the persistent storage system including algorithms of key operations.

3.1 System Architecture

The system architecture for implementation of the proposed storage system is illustrated in Fig. 3. The system consists of two layers: Storage system layer and DBMS layer. The storage layer is also composed of OWL Ontology Loader and Result Manager. The ontology loader is the core of our proposal and the result manager is designed to show the persistent store result.

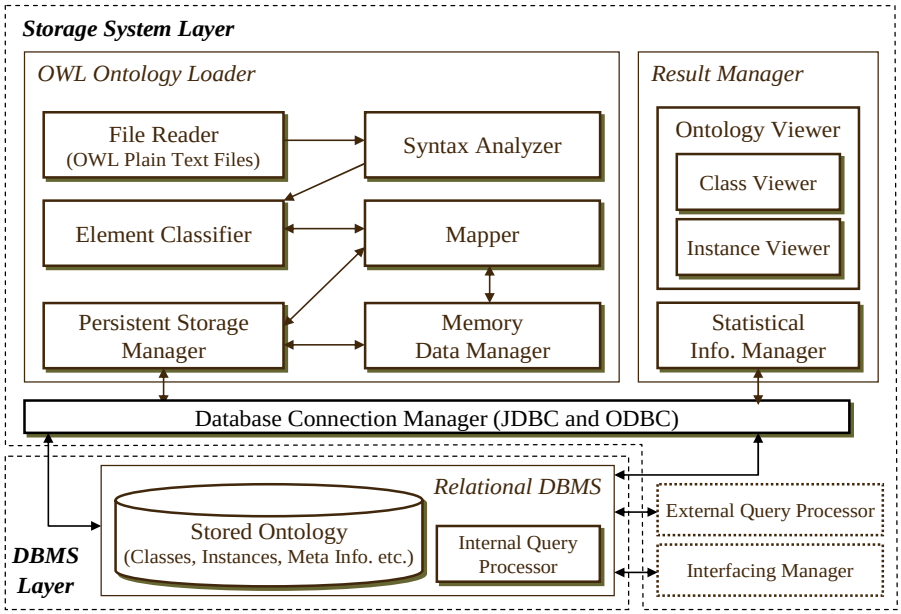

Fig. 3. System architecture for implementation of the proposed system model

The OWL ontology loader has six components: File Reader, Syntax Analyzer, Element Classifier, Mapper, Memory Data Manager, and Persistent Storage Manager. The file reader plays a role to read OWL documents (plain text format) and the syntax analyzer checks the validation of OWL data. The element classifier extracts OWL elements from the valid OWL documents and then classifies them into each correspondent group. The classified OWL elements are used as input data of Mapper. The mapper first sends each element to Persistent Storage Manager and Memory Data Manager to load them on memory and into a persistent database. The persistent storage manager and the memory data manager load the elements into a database and on memory respectively.

The result manager has three components: Class viewer, Instance Viewer and Statistical Information Manager. The class viewer and instance viewer show the stored data (OWL data) to users. The statistical information manager has a displaying role of statistical information such as the number of stored instances or classes, the size of ontology, and so on.

Finally, additional two functions (query processing function and interfacing function to other systems or APIs) could be added to the system. The query processing function is to accept other query language for OWL ontology and to provide query language independent access to users. Relational database management systems have their query processor and they can understand and execute just SQL query statements. However, user queries could be composed in KIF, RQL, or SPARQL style including SQL [11, 12, 13]. In fact, in our prototyping system, only queries by SQL are available. It means that users should compose their queries using SQL. The external query processor should be realized for this function in future.

As aforementioned, various similar systems, APIs, and tools to our proposal have been developed. Also there are diverse ontology models. For interoperation between heterogeneous systems, an interfacing module is required to support the interoperation. This function remains as a further work.

3.2 Overall Loading Algorithm

The loading process mainly consists of two steps: (Step-1) Classification (Analysis and Extraction) step and (Step-2) Conversion (Mapping and Storing) step. The overall algorithm is described as follows:

Overall algorithm for OWL ontology loading

```
INPUT: I          /** OWL ontology (OWL documents) */
OUTPUT: R         /** R indicates PDM in the definition 1 */
START Algorithm {
 1: /** Reading a specific OWL document  */                    /** Step-1 */
 2: owlText = read(I);    //owlText: Stream object for the OWL document
 3: boolean vr = validator.check(owlText);
 4:
 5: if (!vr) {// if the OWL document is not valid, then return an error message
 6:     ExceptionHandler.show(INVALID_ERROR);
 7:     return;
 8: }
 9:
10: /** Extracting and classifying OWL elements */
11: EClassifier classifier new EClassifier(owlText);
12: Mapper mapper = classifier.createMapper();  //Creating a Mapper object
13:
14: /** Loading OWL data on memory and into the persistent database */    /** Step-2 */
15: if (MEM_TAG) {
16:     memManager.load(mapper);
17:     persistentManager.load(memManager); // Creating the final output, R
18: }
19: else {  // Loading OWL data into only persistent database
```

```
20:     persistentManager.load(memManager); // Creating the final output, R
21:  }
22:
23:  /** Generating statistical information of the loading result */
24:  StatisticalManager.createInfo(persistentManager);
}END Algorithm
```

4 Performance Evaluation

This section describes the experiment environment and performance result, and the performance evaluation is presented to show prominence of the proposed system.

4.1 Target Systems and Experiment Environment

This section includes the descriptions on the target systems for comparison, experiment data sets (Ontology sets), and system environment for experiment.

Target Systems

The target system for this comparative evaluation includes Sesame-DB and DLDB-OWL. Jena also provides a method to store OWL data into relational databases. However, Jena is one of APIs for development of systems or tools such as storage, viewer, editor, and so on. Hence, Jena is excluded from the target systems.

Data Sets for Experiment

For experiment, this paper uses data sets generated from UBA (Univ-Bench Artificial Data Generator), which is provided by Lehigh University. The OWL data generator has been developed for SWAT project evaluation [15]. There are two reasons why we use the OWL data sets by UBA as follows: (1) Even though we gathered several OWL ontologies, its ontology size is not acceptable for our experiment on performance evaluation; (2) The SWAT project team already published their evaluations using an ontology set by UBA [16, 17].

We can also generate the same ontology set through using the UBA and the generated data is five OWL ontology sets: LUBM(1,0), LUBM(5,0), LUBM(10,0), LUBM(20,0), and LUBM(50,0). In the notations LUBM(N,S), N and S denote the number of universities and the seed value respectively [16, 17].

System Environment

System environment is as follows:

(1) CPU: Pentium 4 (1.60 GHz);
(2) Memory size (RAM): 256MB;
(3) Heap memory size: 512MB;
(4) Hard disk size: 80GB;

(5) Platform: Windows XP Professional OS;
(6) Java SDK version: Java SDK 1.5.0;
(7) DBMS: Access.

The system requirement is similar to the setting environment in [17]. Our evaluation result is based on the result of Guo's proposal [16, 17]. Thus we prepare for the almost same system setting. In Guo's experiments, the CPU capability is 1.8GHz and it is more powerful than our setting. It means our experiment result is reliable.

4.2 Experiment Result and Discussion

The experiment result is summarized in Table 1. The table presents the load times of each system for four OWL data sets, i.e., {LUBM(1,0), LUBM(5,0), LUBM(10,0), LUBM(20,0)}, except LUBM(50,0).

For LUBM(50,0), Sesame-DB and our system cannot load because of heap memory error. Therefore, this paper finally shows the experiment result with four OWL data sets. We thought the reason of this error is the ontology size in memory. Memory size is also one of important comparative items, but this paper remains it as one of further study issues. For exact estimation of our experiment, the experiment result with our system for each data set has been done two times. Hence, the result in Table 1 describes the average values of both experiment results.

With LUBM(1,0) data set, Sesame-DB requires 542 seconds to load it. On the other hand, DLDB-OWL and our system take 343 seconds and 270 seconds respectively. For LUBM(5,0), Sesame-DB:DLDB-OWL:Proposed = 10,811:3,117:1,691.

Table 1. Experiment results

(Unit: Second)

Data Set	DLDB-OWL	Sesame-DB	Proposed (Avg)
LUBM(1,0)	343	542	270
LUBM(5,0)	3,117	10,811	1,691
LUBM(10,0)	6,881	44,870	3,341
LUBM(20,0)	15,773	167,753	7,546

Fig. 4 graphically depicts the experiment result in Table 1. This figure shows intuitively how the loading time grows as the OWL set size increases. Our system's performance is more efficient than the others.

Table 2 shows the performance ratio. In this table, in case of the first data set, DLDB-OWL and Sesame-DB require much cost (load time) than our system (1.27 times and 2.00 times). For LUBM(20,0), both systems consume much more time (2.09 times and 22.23 times). Especially, Sesame-DB's load time exponentially increases.

In addition, we carried out an experiment on the load time for comparison with the Jena persistent storage model and our proposal. Jena provides two storage models-

Jena1 version and Jena2 version. Jena2 changed the Jena1 storage model for performance improvement and provides an abnormalized database structure. In our experiment result, our system showed the efficient performance comparing with Jena. However, this comparative evaluation has no critical meaning, because Jena is an API developed for general purpose. Even though this experiment is not useful, we can use the evaluation result when we try to design and develop a Jena plug-in for enhanced management under the Jena development environment.

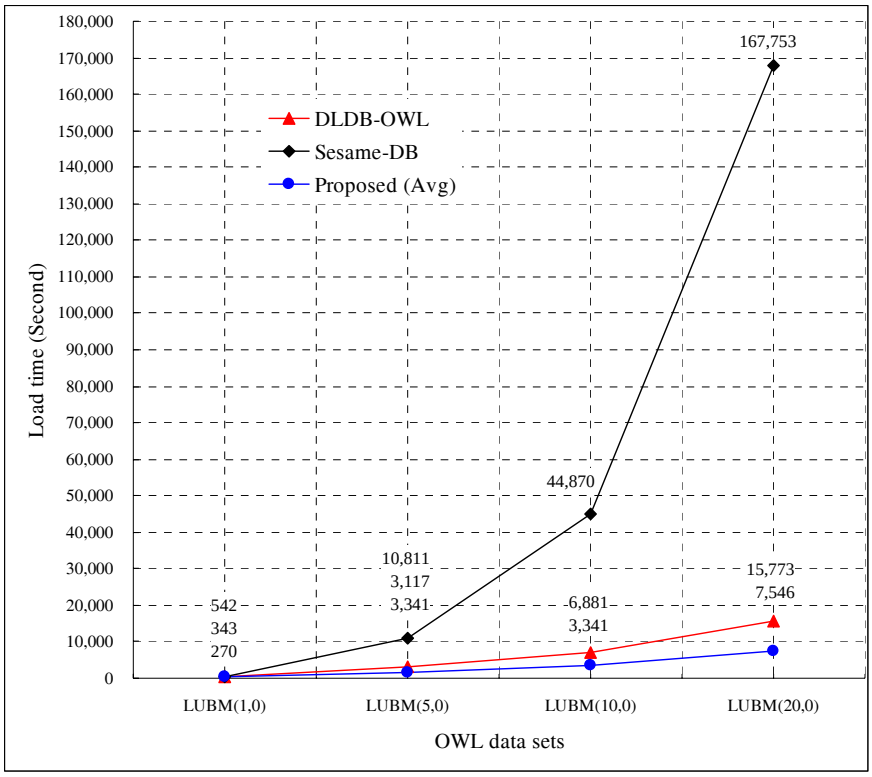

Fig. 4. Graph model-based experiment result

Table 2. Performance analysis: Ratio (Pivot model: The proposed)

Data Set	DLDB-OWL	Sesame-DB	Proposed (Avg)
LUBM(1,0)	1.27	2.00	1.00
LUBM(5,0)	1.84	6.39	1.00
LUBM(10,0)	2.06	13.43	1.00
LUBM(20,0)	2.09	22.23	1.00

5 Conclusion and Further Study

This paper proposed a new persistent storage system. To achieve our goal, we first defined a relational data structure for permanently storing OWL ontologies. The structure has been defined considering its usability and performance. Second, we designed the processing system to check OWL files, extract and classify elements, and store OWL data (Classes and Individuals) into a specific relational database. And the system has been implemented under the following environment: (1)Java SDK: JDK 1.5; and (2)RDBMS: MS Access. Finally, we conducted the performance experiment. In the performance result, our persistent storage system is better than the target systems, Sesame-DB and DLDB-OWL.

There still remain several considerations. First, the performance is dependent on data characteristics, database servers, and so on. We have a research plan to define the relationships between the variable factors. Second, for applying storage systems to real applications, query response time is one the most important key factors. This paper didn't deal with this issue. It is a critical further study and should be handled to show the usability of storing systems including our proposal.

References

1. Yergeau, F., Cowan, J., Bray, T., Paoli, J., Sperberg-McQueen, C. M., and Maler, E.: Extensible Markup Language (XML) 1.1, W3C Recommendation, February (2004)
2. Resource Description Framework (RDF), http://www.w3.org/RDF/.
3. Bechett, D.: RDF/XML Syntax Specification (Revised), W3C Recommendation, 10 February (2004)
4. Connolly, D., Harmelen, F.V., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., and Stein, L.A.: DAML+OIL Reference Description, W3C Note, 18 December (2001)
5. Smith, M.K., Welty, C., and McGuinness, D.L.: OWL Web Ontology Language Guide, W3C Recommendation, 10 February (2004) http://www.w3c.org/TR/2004/REC-owl-guide-20040210/
6. Roldan-Garcia, M.M. and Aldana-Montes, J.F.: A Tool for Storing OWL Using Database Technology, November (2005)
7. Pan, Z. and Heflin, J.: DLDB: Extending Relational Databases to Support Semantic Web Queries, In Workshop on Practical and Scaleable Semantic Web Systems, The 2nd International Semantic Web Conference (ISWC2003) (2003)
8. Broekstra, J. and Kampman, A.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, The 1st International Semantic Web Conference (ISWC2002) (2002)
9. SourceForge.net, Jena2 Database Interface - Database Layout, November (2004) http://jena.sourceforge.net/DB/layout.html
10. Jena: A Semantic Web Framework for Java, http://jena.sourceforge.net/
11. Gensereth, M. and Fikes, R.: Knowledge Interchange Format, Stanford Logic Report Logic, Stanford University, http://logic.stanford.edu/kif/kif.html
12. Karvounarakis, G., Magkanaraki, A., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M., and Tolle, K.: Querying the Semantic Web with RQL, Elsevier Science, Computer Networks and ISDN Systems Journal, Vol. 42, No. 5, pp. 617-640, August (2003)
13. Prud'hommeaux, E. and Seaborne, A.: SPARQL Query Language for RDF, W3C Candidate Recommendation, 6 April (2006)

14. Carroll, J.J. and Roo, J.D.: OWL Web Ontology Language Test Cases, W3C Recommendation, 10 February (2004)
15. Guo, Y.: Data Generator(UBA): UBA1.7, http://swat.cse.lehigh.edu/projects/lubm/.
16. Guo, Y., Pan, Z., and Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets, Vol. LNCS 3298 (2004)
17. Guo, Y., Pan, Z., and Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems, Journal of Web Semantics (2005)