An RDF Storage and Query Framework with Flexible Inference Strategy

Wennan Shen and Yuzhong Qu

Department of Computer Science and Engineering,
Southeast University, Nanjing 210096, P.R. China
{wnshen, yzqu}@seu.edu.cn

Abstract. In the Semantic Web, RDF (Resource Description Framework) and RDF Schema are commonly used to describe metadata. There are a great many RDF data in current web, therefore, efficient storage and retrieval of large RDF data sets is required. So far, several RDF storage and query system are developed. According to the inference strategy they used, they can be classified into two categories, one exclusively use forward chaining strategy; the other exclusively use backward chaining strategy. In most cases, the query performance of the former is superior to that of the latter. However, in some cases, the disadvantage of larger storage space may at some point outweigh the advantage of faster querying. Further, the existing systems that exclusively use forward chaining strategy have not yet presented a good solution to the deletion operation by now. In this paper, we design a RDF storage and query framework with flexible inference strategy, which can combine forward and backward chaining inference strategies. In addition, a new solution to the deletion operation is also given within our framework. The feasibility of our framework is illustrated by primary experiments.

1 Introduction

The Web is a huge collection of interconnected data. Managing and processing such information is difficult due to the fact that the Web lacks semantic information. The Semantic Web has emerged as the next generation of the World Wide Web, and it is envisioned to build an infrastructure of machine-readable semantics for the data on the Web. In the Semantic Web [14], RDF [12] (Resource Description Framework) and RDF Schema [3] are commonly used to describe metadata.

The Resource Description Framework (RDF) is the first W3C recommendation for enriching information resources of the Web with metadata descriptions. Information resources are, for example, web pages or books. Descriptions can be characteristics of resources, such as author or content of a website. We call such descriptions metadata. The atomic constructs of RDF are statements, which are triples consisting of the resource being described, a property, and a property value. A collection of RDF statements can be intuitively understood as a graph: resources are nodes and statements are arcs connecting the nodes.

The RDF data model has no mechanism to define names for properties or resources. For this purpose, the RDF schema is needed to define resource types and

X. Zhou et al. (Eds.): APWeb 2006, LNCS 3841, pp. 166 – 175, 2006.

property names. Different RDF schemas can be defined and used for different application areas. RDF Schema [3] is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. RDF schema statements are valid RDF statements because their structure follows the syntax of the RDF data model.

There are a great many RDF data on current web, therefore, efficient storage and retrieval of large RDF data sets is required. So far, several RDF storage and query system are developed. According to the inference strategy they used, they can be classified into two categories, one exclusively use backward chaining strategy, such as Jena [17]; the other exclusively use forward chaining strategy, such as RStar [15] and Sesame [5].

The inference engine that uses forward chaining strategy is triggered when triples are inserted into an RDF storage, the generated triples by the inference engine are inserted into the storage together with the original triples. This will unavoidably increase the need for disc memory. However, the task of processing a query is reduced to simple lookup without inference. On the contrary, backward chaining inference engine is triggered when the query is evaluated. The main advantage of backward chaining inference is the decrease in required storage size and import data time, while the main disadvantage is the decrease in performance of query processing.

In addition, a forward chaining based system needs a truth maintenance system (TMS) to maintain consistency as well as make derivations available. Consider a situation in which a triple insert into an RDF storage, and the triple match the premise part of a rule used in the inference engine, then the rule is fired, consequently, additional triples generated by the rule shall be insert into the RDF storage. If at some time later the triple needs to be deleted from the RDF storage, in order to maintain the consistency of the storage, the triples derived from it also should be deleted from the RDF storage. To cope with this scenario, a TMS system that records the justifications of triples should be built into forward chaining based system. As far as backward chaining based system concerned, this is not a problem since the insertion operation can't result in additional derived triples.

Many performance tests were conducted for current RDF storage and query systems [15,9], the results show that forward chaining based systems are superior to backward chaining based systems. However, in literature [4], the authors indicated that when RDF data consists exclusively of a large class or property hierarchy that is both broad and deep, or the complexity of the model theory and expressiveness of the modeling language increase (for example when moving from RDF Schema to OWL [16]), the disadvantage of larger storage space may at some point outweigh the advantage of faster querying.

Based on the above considerations, we feel that the inference strategy employed by existing systems is not flexible enough for semantic web applications. Further, the existing systems that exclusively use forward chaining strategy have not yet presented a good solution to the deletion operation by now. Therefore, we design a RDF storage and query framework with flexible inference strategy, which can combine forward and backward chaining inference strategies. In addition, a new solution to the deletion operation is also given within our framework. The feasibility of our framework is illustrated by primary experiments.

2 An RDF Storage and Query Framework

2.1 Overview

Fig 1 shows an RDF storage and query framework with flexible inference strategy. There are three functions for the end user, that is inserting data, deleting data and querying data. Two kinds of inference engines, namely forward chaining inference engine and backward chaining inference engine, are designed for the functionality of data insertion and data query respectively. The framework has an inference rule controller to control rules used in the inference engines. As mentioned in the previous section, a forward chaining inference engine has special needs for truth maintenance system. Therefore, a truth maintenance system is built into the framework to maintain consistency as well as make derivations available. The TMS controller is designed to determine whether or not the truth maintenance system should be called. The key issues related to these components will be addressed in the following subsections.

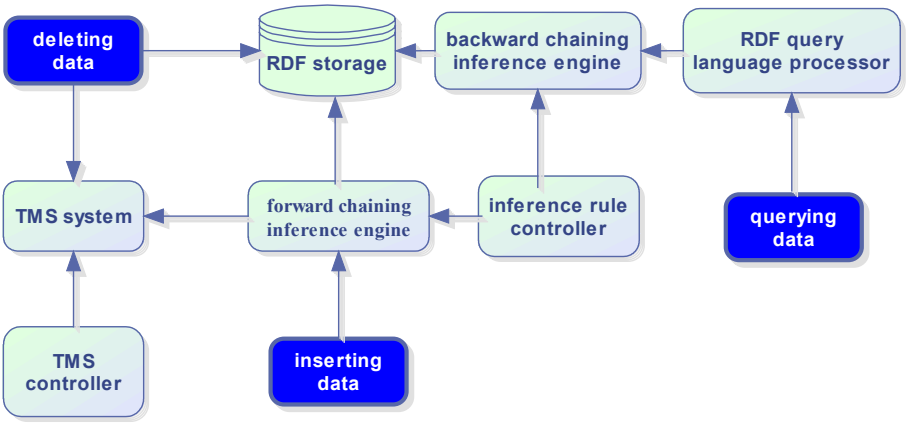

Fig 1. An RDF storage and query framework

2.2 Inference Rule Controller and Inference Engines

As discussed in section 1, forward chaining inference strategy and backward chaining inference strategy have their strong strength. Therefore, the framework uses a mixed strategy, which combines both of the inference strategies. There are two inference engines in the framework, forward chaining inference engine for data insertion and backward chaining inference engine for data query. The rules used in each inference engine are controlled by the inference rule controller. Applications can configure the inference engines through the controller according to their own characteristics.

To insert a triple into the RDF storage, data insertion sends the triples to the forward chaining inference engine. The rules in the forward chaining inference engine may be fired by the triples inferred by the rules in the backward chaining inference engine. In order to make the query results complete, the forward chaining inference

engine make an inference based on the current RDF storage state and the rules in both of the two engines, then insert both inferred triples and original triples into the RDF storage except for the triples directly inferred by the rules in the backward inference engine, meanwhile, it inserts the dependence of triples into the TMS system. The whole procedure runs iteratively.

To query information from the RDF storage, the backward chaining inference engine receives query from RDF query language processor, then it draws conclusions in terms of the current RDF storage state and the rules which inference rules controller specifics.

2.3 TMS Controller and TMS System

A forward chaining inference system has special need for a truth maintenance system. Most of TMS systems are associated with forward chaining inference [6,7,8]. There are two related reasons for this need. One is to keep the consistency of RDF storage, the other is to help to deal with deletion operation. The TMS system in this framework records the justifications for each triple inferred by an RDFS rule. When a triple is removed from the RDF storage, any justifications in which it plays a part are also removed. The triples justified by removing justification are checked to see if they are still supported by other justification. If not, then these triples are also removed.

Sometimes such a TMS system is too expensive to use and it is not needed for some applications. Consequently, applications can choose whether or not to use TMS system through the TMS controller component.

2.4 RDF Storage

Most of existing RDF storage systems use relational or object-relational database management systems as backend stores [1,2]. This is a straightforward approach since it is appropriate to represent RDF triples in a relational table of three columns and the relational DBMS (RDBMS) has been well studied.

Other components access RDF storage through standard SQL sentence. As to forward chaining inference engine, if the triples in the current storage match the premise part of a RDFS rule, then the rule is fired, newly derived triples are recorded into the storage, the justifications that justify these derived triples are inserted into the TMS system, then do the same actions to the derived triples until no additional triples are generated. As far as Backward chaining inference engine is concerned, if the search target matches the conclusion part of a RDFS rule, search the storage, if triples match the premise part of the rule, the matched triples add to the result set, then take the premise part of the rule as sub target, carry out the same actions.

2.5 RDF Query Language Processor

Several languages for querying RDF data have been proposed and implemented, some in the form of traditional database query languages (e.g. SQL, OQL), others based on logic and rule languages. Judging from the impact of SQL to the database community, standardization of RDF query language will definitely help the adoption of RDF query engines, make the development of applications a lot easier, and will thus help the Semantic Web in general [10]. W3C set up RDF Data Access Working Group

(DAWG) in Feb. 2004. DAWG devotes to developing specifications for RDF query language and access control protocol. SPARQL is a RDF query language developed by DAWG according to the technology requirement and design objectives referred above.

RDF query language processor receives the request in a specific RDF query language form, analyzes and checks whether the submitted query accords with the syntax of the query language. A valid query is parsed and transformed into a medium state. Then send the result to the backward chaining inference engine.

3 A New Solution to the Deletion Operation

The existing systems that exclusively use forward chaining strategy have not yet presented a good solution to the deletion operation by now. RStar [15] did not provide the deletion operation. In order to deal with the cyclic dependency problem in TMS system, Sesame [5] gives a complex algorithm, with which each deletion operation needs recalculating the closure of TMS system, so it isn't applicable to applications with large TMS systems. Therefore, in this section, we give a new solution to this problem, which consists of two algorithms, including insertion algorithm and deletion algorithm. The insertion algorithm copes with the cyclic dependency problem, while the deletion algorithm is relative simple.

At first, we give three definitions.

Definition 1: Dependency between rules: Let rule 1 is a11, a12→b1; rule 2 is a21, a22→b2. If some conclusions in rule1 match some premises in rule2, then we can say that rule 2 depends on rule 1.

Definition 2: Dependency between triples: If triple 3 can be inferred by triple 1 and triple 2 through certain rule, then we can say that triple 3 depend on triple 1 and triple 2.

Definition 3: Justification in the TMS system has the form (inf, dep1, dep2, rule), where inf is a triple justified by the justification, dep1 and dep2 are triples justifying inf, and rule is the RDFS rule that produces the justification. When dep1=null and dep2=null, it indicates that inf is an explicit triple.

3.1 Dependency Between RDFS Entailment Rules

The RDF Semantics [11] is a specification of a model-theoretic semantics for RDF and RDF Schema, and it presents a set of entailment rules. In [13], the author characterizes these rules as follow:

- Type Rules assign default ("root") types for resources (rules rdf1, rdfs4a and rdfs4b).
- Subclass Rules generate the transitive closures of subclass (rules rdfs8, rdfs9, rdfs10).
- Subproperty Rules are used to generate the transitive closure resulting from subproperty (rules rdfs5, rdfs6, rdfs7).

– Domain/Range Rules infer resource types from domain and range assignments (rules rdfs2 and rdfs3).

The RDF Semantics specification was published on February 10,2004. It added rules related to rdfs:ContainerMembershipProperty (rdfs12) and rdfs:Datatype (rdfs13).

Table 2 shows the dependency between RDFS entailment rules in terms of the RDFS Semantics specification. In the table, the rules on the horizon direction are triggering rules, and on the vertical direction are triggered rules. If one rule depend on another one, place correspond to those rules is filled with a token *. For example, definitions of rdfs3 and rdfs9 are presented in table 1. We can see that rdfs9 depend on rdfs3, so we place a token * in row 9, column 3.

Table 1. Definitions of rdfs3 and rdfs9

rdfs3:	aaa rdfs:range xxx ,uuu aaa vvv → vvv rdf:type xxx
rdfs9:	uuu rdfs:subClassOf xxx ,vvv rdf:type uuu → vvv rdf:type xxx

Table 2. Dependency between RDFS entailment rules

Rule:	1	2	3	4a	4b	5	6	7	8	9	10	11	12	13
1								*						
2		*	*			*	*	*	*	*	*	*	*	*
3		*	*			*	*	*	*	*	*	*	*	*
4a		*	*			*	*	*	*	*	*	*	*	*
4b		*	*			*	*	*	*	*	*	*	*	*
5						*	*						*	
6		*	*											
7						*	*	*					*	
8		*	*											
9		*	*						*	*	*	*		*
10		*	*											
11									*		*	*		*
12		*	*											
13		*	*											

3.2 Cyclic Dependency of Rules

When compute justification in the TMS system, cyclic dependencies [4] may occur. The following two examples present the problem.

Example 1: 1. (uuu, rdf:type, rdfs:Resource) (explicit)
2. (rdf:type, rdfs:domain, rdfs:Resource) (explicit)

Example 2: 1. (rdfs:subClassOf, rdfs:domain, rdfs:Class) (explicit)
2. (uuu, rdf:subClassOf, rdfs:Resource) (explicit)
3. (uuu, rdf:type, rdfs:Class) (derived)

Example 1 shows that a justification (1, 1, 2, rdfs2) added into the TMS system in term of the rdfs2, namely triple 1 is justified by itself. As to Example 2, the following

justifications, (3, 1, 2, rdfs2) (2, 3, null, rdfs8), are added into the TMS system according to rdfs2 and rdfs8 respectively. This presents that triple 2 depends on triple 3 justified by triple 2. All these examples have cyclic dependencies.

The cyclic dependencies result in a problem. When delete a triple, if the TMS system contains a justification for it, it can't be deleted. Therefore, in Example 1, triple 1 cannot be deleted because it is justified by itself. In Example 2, it seems that triple 2 cannot be deleted because the TMS system contains a justification that justifies it. However, the justification says that triple 3 depends on triple 2, so this deletion can be conducted.

3.3 Algorithm

Two algorithms, namely insertion algorithm and deletion algorithm, are presented in Table 3 and Table 4 respectively.

The following terms are used in the two algorithms.

S: the set of justifications in TMS system.
T: the set of triples including both explicit triples and derived triples.
A: the set of triples that will be inserted into RDF storage.
D: the set of triples that will be deleted from RDF storage.
V: the set of triples that depend on the current inserted triple.
I: the set of triples that were inferred by the current insert triple.

Table 3. insertion algorithm

Step1. For each triple t in set A, insert (t,null,null,null) to S, then determine whether t is in set T, if yes, delete t from A, otherwise let V empty, bind V to t. Go to step 2.
Step2. Determine whether A is empty, if yes, terminate,otherwise select a triple t2 from A, go to step 3.
Step3. Insert t2 into T, and compute I of t2, meanwhile, bind t2's V to each triple in I and get the dependency. Go to step 4.
Step4. Determine whether I is empty. If yes, go to step 2, otherwise select a triple t4 from I, go to step 5.
Step5. Insert the dependent triples (produce in step 3) of t4 to t4's V, then determine whether t4 is in set T, if yes, add dependence of t4 to S when set V does not contain t4 (this action eliminates the cyclic dependency), otherwise add t4 to A. Go to step 4.

Table 4. deletion algorithm

Step1. For each triple d in D, if d is explicit, then let d is derived. Otherwise, delete d form D. Go to step 2.
Step2. Let a variable named removed is false. Go to step 3.
Step3. If D is null or removed is false, terminate. Otherwise, go to step 4.
Step4. For each triple t in D, if for any justification s (fs,d1s,d2s,rule) in S, fs is not equal t, delete t from D and T, let removed is true, then for each justification q(fq,d1q,d2q,rule) in S, if d1q is equal t or d2q is equal t, delete q from S, if fq is derived, add fq to D. Go to step 3.

4 Experiment

We have developed a prototype system of the presented framework. In order to evaluate the feasibility of our framework, we conduct an experiment on Wordnet data set. Wordnet is a lexical resource that defines terms as well as their descriptions and semantic relations between them. In our experiment, we choose the Wordnet 1.6 schema (wordnet-20000620.xml) and the set of nouns (wordnet-20000620.xml).

The experiment was run on a 2.0GHz PC with 512M physical memory. The operating system is Windows XP Professional and the backend database is mysql 4.1.12.

The full set of RDFS rules is highly redundant. The feathers of some rules are rarely used, such as rdfs1, rdfs4a, rdfs4b, etc. In this experiment, we take rdfs2, rdfs3, rdfs5, rdfs7, rdfs9, rdfs11 into account. At first, we configure the forward chaining inference engine with all of these rules. Table 5 shows the number of triples inferred by each rule. We see that the size of the triples inferred by rdfs3 is more than half of all triples inferred.

Table 5. The number of triples inferred by each rule with first configuration

rdfs2	rdfs3	rdfs5	rdfs7	rdfs9	rdfs11
0	122678	0	0	110554	1

Table 6. The number of triples inferred by each rule with second configuration

rdfs2	rdfs3	rdfs5	rdfs7	rdfs9	rdfs11
0	0	0	0	110554	1

Then, we configure the forward chaining inference engine with rdfs2, rdfs5, rdf7, rdfs9, rdfs11, and backward chaining inference engine with rdfs3. Table 6 shows the number of triples inferred by each rule. The following present two query examples used in our experiment. Query 2 relates to the rdfs3, but Query 1 doesn't.

Query 1: return comment of the verb in Wordnet.
PREFIX wn: < http://www.cogsci.princeton.edu/~wn/schema/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select ?comment where{wn:Verb rdfs:comment ?comment}
Query 2: return the type of the word in the form of "learning".
PREFIX wn: < http://www.cogsci.princeton.edu/~wn/schema/>
PREFIX rdf: < http://www.w3.org/1999/02/22-rdf-syntax-ns>
select ?type where {?ID wn:wordForm 'leaning'. ?ID rdf:type ?type}

We evaluate the systems according to the first and second configuration with the query samples referred above. The query results generated by both systems are same. As demonstrated in Table 7, for queries relating to rdfs3, e.g. Query 2, the former configuration is superior to the latter configuration. However, for queries that don't relate to rdfs3, e.g. Query 1, the advantage of the latter configuration is obvious. In addition, the latter system needs less storage space. The experiment illustrates that our framework is feasible.

Table 7. Comparative result of query time with different configuration

	Query 1 (second)	Query 2 (second)
First configuration	1.130	1.412
Second configuration	0.812	1.627

5 Conclusion

In this paper, we present an RDF storage and query framework with flexible inference strategy, which can combine forward and backward chaining inference strategies. In addition, a new solution to the deletion operation is also given within our framework. The feasibility of our framework is illustrated by primary experiments.

This work is a primary research in combing two inference strategies. More experiments are needed to figure out which kinds of configurations can best benefit from our framework. And automatic or semi-automatic configuration is very valuable to exploit the practical usage of our framework. These will be our future work.

Acknowledgments

The work is supported in part by National Key Basic Research and Development Program of China under Grant 2003CB317004, and in part by the Natural Science Foundation of Jiangsu Province, China, under Grant BK2003001. We would like to thank Dr. Yuqing Zhai and Dr. Yangbing Wang for their suggestions on this paper.

References

1. Beckett, D.: Scalability and Storage: Survey of Free Software / Open Source RDF storage-systems. Latest version is available at http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report/
2. Beckett, D., Grant, J.: Mapping Semantic Web Data with RDBMSes. Latest version is available at http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/
3. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. Latest version is available at http://www.w3.org/TR/rdf-schema/
4. Broekstra, J., Kampman, A.: Inferencing and Truth Maintenance in RDF Schema. In Workshop on Practical and Scalable Semantic Systems (2003)
5. Broekstra, J., Kampman, A., Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. Proc. of the 1st International Semantic Web Conference (2002) 54-68,
6. Doyle, J.: A truth maintenance system. Artificial Intelligence (1979) 231-272
7. Doyle, J.: The ins and outs of reason maintenance. In 8th International Conference on Artificial Intelligence (1983) 349-351
8. Finin, T., Fritzson. R., Matuszek, D.: Adding Forward Chaining and Truth Maintenance to Prolog. Artificial Intelligence Applications Proceedings of Fifth Conference (1989) 123 – 130

9. Guo, Y.B., Pan, Z.X., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In Proceedings of the 3rd International Semantic Web Conference. Lecture Notes in Computer Science 3298 Springer (2004)
10. Haase1, P., Broekstra, J., Eberhart1, A., Volz1, R.: A Comparison of RDF Query Languages. In Proceedings of the Third International Semantic Web Conference, volume 3298. Lecture Notes in Computer Science, Hiroshima, Japan. Springer-Verlag (2004)
11. Hayes, P.: RDF Semantics. W3C Recommendation 10 February 2004. Latest version is available at http://www.w3.org/TR/rdf-mt/
12. Klyne, G., Carroll, J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. Latest version is available at http://www.w3.org/TR/rdf-concepts/
13. Lassila, O.: Taking the RDF Model Theory Out for a Spin. In Ian Horrocks and James Hendler, editors, Proceedings of the First International Semantic Web Conference, ISWC 2002, Sardinia, Italy, number 2342. Lecture Notes in Computer Science (2002) 307–317
14. Lee, T.B., Handler, J., Lassila, O.: The Semantic Web. In Scientific American, vol. 184(2001) 34-43
15. Ma, L., Su, Z., Pan, Y., Zhang, L., Liu, T.: RStar: An RDF Storage and Query System for Enterprise Resource Management. In Proceedings of the Thirteenth ACM conference on Information and knowledge management, Washington, D.C., USA (2004) 484 – 491
16. McGuinness, D.L., Harmelen, F.V.: OWL Web Ontology Language Overview. W3C Recommendation. Latest version is available at http://www.w3.org/TR/owl-features/
17. Wilkinson, K., Sayers, C., Kuno, H., Reynolds, D.: Efficient RDF Storage and Retrieval in Jena2. Proc. of the 1st International Workshop on Semantic Web and Databases (2003) 131-151