System Π: A Hypergraph Based Native RDF Repository[*]

Gang Wu, Juanzi Li, Kehong Wang
Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China
{wug, ljz, wkh}@keg.cs.tsinghua.edu.cn

ABSTRACT

To manage the increasing amount of RDF data, an RDF repository should provide not only necessary scalability and efficiency, but also sufficient inference capabilities. In this paper, we propose a native RDF repository, System Π, to pursue a better tradeoff among the above requirements. System Π takes the hypergraph representation for RDF as the data model for its persistent storage, which effectively avoids the costs of data model transformation when accessing RDF data. In addition, a set of efficient semantic query processing techniques are designed. The results of performance evaluation on the LUBM benchmark show that System Π has a better combined metric value than the other comparable systems.

Categories and Subject Descriptors: H.3.0 [Information Storage and Retrieval]: General

General Terms: Design, Management, Performance

Keywords: Hypergraph, RDF, Repository

1. INTRODUCTION

With the rapid growth of RDF data on the Web, more and more Semantic Web applications turn to RDF repositories for help in pursuing better data management performance in system scalability, query efficiency, and inference capabilities. Traditional data management systems cannot fulfill the requirements directly, because the data models of them are different from that of RDF, and most of them do not provide any inference capability. Hence, some RDF repositories are developed specially. Currently, these repositories are at their infant stages, and hence there is still sufficient room for improving the performance. A practical direction is to make the persistent storage represent the RDF data model more efficiently.

The data model of RDF is the RDF graph which allows several representations. The most popular representations are the *triple sets* and the *directed labeled graph*. However, they all have some limitations as the data model of RDF repository persistent storages. Jonathan Hayes proposes a hypergraph representation to deal with this situation [4]. By letting an edge be composed of three vertices corresponding to the elements of a triple, the hypergraph representation can support efficient traversals on the RDF graph, and overcome the above two limitations. Suppose T is an RDF graph. The hypergraph representation of T is $\mathcal{G} = (V, \mathcal{E})$, where $V = \{v_x | x \in U \cup B \cup L\}$, $\mathcal{E} = \{(v_s, v_p, v_o) | (s, p, o) \in T\}$.

[*]This work is supported by the National Natural Science Foundation of China under Grant No.90604025 and the Major State Basic Research Development Program of China (973 Program) under Grant No.2003CB317007 and No.2007CB310803.

1.1 Hypergraph Based Persistent Storage

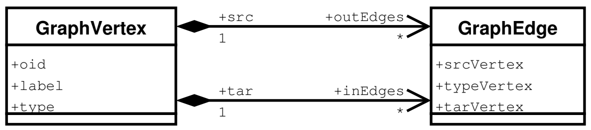

Figure 1: The class diagram of hypergraph based persistent storage

In System Π, we design a concise storage scheme as the class diagram shown in Figure 1. Here, class GraphVertex represents a vertex in V, which has three fields: oid, label, and type. Field oid has the type of integer with the length of 64 bits, and is used to uniquely identify a vertex. Field label is a variable length string used to record the value of URI reference or literal. Field type is also a 64 bits integer designed to encode specific semantic information of the vertex in a bitmap manner. Class GraphEdge represents an edge in $\mathcal{E}$, where srcVertex, typeVertex, and tarVertex correspond to the oids of three elements of a triple. The edges starting from and ending at a vertex can be visited through field outEdges and inEdges of the vertex. The total size costed by the persistent storage is the same as that of the repositories directly storing RDF triples if not taking inEdges into account. There are three advantages: **1)** Class GraphVertex, GraphEdge, and their relationship together reflect the hypergraph representation for RDF. The idea is intuitive, easy to understand, and the implementations of the algorithms from graph theory are straightforward. **2)** Further reducing the overhead of storage space is possible. As edges in a vertex's outEdges (or inEdges) have the same value of srcVertex (or tarVertex), we can omit srcVertex (or tarVertex) to simplify the representation of GraphEdge. **3)** Edges are clustered by vertices so that edges having the same srcVertex or tarVertex values can be accessed by a less database operation.

1.2 Physical Query Processing

1.2.1 Hypergraph Traversal

The fundamental RDF data access approach in System Π is to traverse on the underling hypergraph. Analogizing pointer chasing operations in object database systems, hypergraph traversal could also be an implementation method for join operation.

1.2.2 Vertex Value Index

The hypergraph storage only supports accessing a vertex by its oid, which means we have to traverse the hypergraph to filter out the vertex with the specific URI value in its label field. Since such operation is frequent, a hash index structure, named *Vertex Value Index* (*VVI*), is designed for quick access. The key of VVI is a URI or a literal, and the value of VVI is the oid of a vertex.

1.2.3 Triple Indices

Triple Indices are B-Tree index structures built on three triple sets to facilitate join operation between triple sets. One triple set contains triples in the origin form, i.e. (subject, predicate, object). The other two reorder the triple set in forms of (object, subject, predicate) and (predicate, object, subject). Each indexes a triple set by taking each triple as a key of the B-Tree. Given a triple pattern, no matter how many and where variables are, all matches can be found by means of one of the indices. When getting two triple sets bound to two triple patterns, a sort merge join is enough to work out the final results. There are two principles in the choice of join approach between hypergraph traversal and triple indices: **1)** If the predicate of a triple pattern has a owl:cardinality property valued 1, priority should be given to hypergraph traversal. **2)** If there exist more than one variable in a triple pattern, priority should be given to triple indices based approach.

1.2.4 Index for Transitive Properties

rdfs:subClassOf, rdfs:subPropertyOf, and other properties owning owl:TransitiveProperty property are all transitive. Transitive closure computation is one of the most basic inference capabilities for an RDF repository. In System Π, we employ an index structure based on the *Prime number Labeling Scheme for Directed graph* (*PLSD*). Comparing with our previous work [10], the new labeling scheme can support arbitrary directed graph even those with cycles. In [9], formal definitions, proof, algorithm description, and detail experimental results are presented. Given a transitive property p, if all edges are removed from the hypergraph $\mathcal{G}$ except those taking p as the predicates, we call the remaining sub-graph $\mathcal{G}_p$. Then, with the help of PLSD labels for vertices in $\mathcal{G}_p$, transitive closure computation can be evaluated efficiently using only simple arithmetic operations like division and unique factorization of composite integer. In System Π, we index PLSD labels with a B-Tree structure.

1.3 Inference Strategy

In order to tradeoff between inference capabilities and the computational complexity, the set of inference rules for pD^* semantics [8] are implemented in System Π. Together with the following proposed hybrid inference strategies (involving partial forward chaining, backward chaining, and labeling scheme), System Π represents the OWL-Lite compatible inference capabilities with a NP-complete computational complexity. For the sake of convenience, we refer the rules for RDFS with a prefix “rdfs” and their rule numbers[1], and refer the rules for pD^* semantics with a prefix “owl” and their rule numbers[2]. **Strategy 1**: For triples involved in rule rdfs2(a), rdfs3(a), rdfs6, rdfs7, owl4, owl7, owl12 and owl13, System Π indexes them with PLSD Index at the stage of RDF data loading. **Strategy 2**: Rule rdfs4, rdfs5, owl1, owl2, owl5, owl6, owl9, owl10, owl11, owl14, owl15 and owl16 are also processed at the stage of RDF data loading, which may further trigger inferences for rules in Strategy 1 and 2. The closure of inference is conducted in a forward chaining manner. **Strategy 3**: For a semantic query that needs inferences for the rules in Strategy 1 or any rule not mentioned in Strategy 1 and Strategy 2, System Π will process the rules in a backward chaining manner.

1.4 Logical Query Plan

SPARQL is the default query interface of System Π. According to the specification of SPARQL, a query string is converted into a SPARQL algebra expression constructed with a set of operators in the query parser. There are 13 operators totally, including Filter, Join, Diff, LeftJoin, Union, ToList, OrderBy, Project, Distinct, Reduced, Slice, BGP, and Graph. However, SPARQL is defined to support only simple entailment for matching RDF graphs. In other words, the above operators are not enough for expressing inference semantics. Hence, for purpose of inference, we define another three operators, i.e., Transitive, Symmetric, and Inverse.

[1]The rule numbers can be found in Definition 3 of [6].

[2]The rule numbers can be found in Table 7 of [8].

2. PERFORMANCE EVALUATION

We compared System Π with other RDF repositories against Lehigh University Benchmark (LUBM) [2]. LUBM is the de facto standard benchmark for evaluating RDF repositories. We generated four data sets that contain OWL files describing information of one, five, ten, and twenty universities (named LUBM(1,0), LUBM(5,0), LUBM(10,0), and LUBM(20,0)). All experimental results can be found in [9]. Here, we only show the results of combined metric which is introduced in [2] to tradeoff the performance between inference capabilities and query response time. A larger combined metric value indicates a better overall performance. In this experiment, we set $a = 500, b = 5, \alpha = 1, \beta = 1, w_i = w_j, (i, j = 1, ..., 14)$. The final results are illustrated in Figure 2.

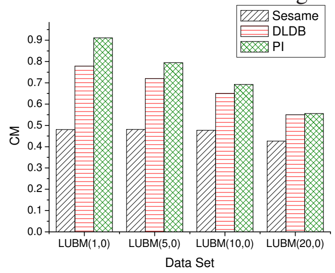

Figure 2: Combined metric values

The combined metric value of System Π is 0.91127 for LUBM(1,0) which is 1.17 times and 1.897 times that of DLDB [7] and Sesame [1] respectively. For the other three data sets, the combined metric values are still above 0.5 and higher than both DLDB and Sesame.

Most of the current RDF repositories, like Sesame [1], DLDB-OWL [7], 3Store [3], and RStar [5], are based on traditional database models. They have good scalability, acceptable query response time and inference capabilities. However, the performance bottleneck is inevitable because of extra costs for data model transformation in accessing RDF data within a non-RDF persistent storage. System Π solves the problem by storing RDF data directly in RDF data model, which enables more optimization techniques based on the RDF data model characteristics to improve the performance.

3. REFERENCES

[1] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *ISWC 2002*, pages 54–68, London, UK, 2002.

[2] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2-3):158–182, 2005.

[3] S. Harris and N. Gibbins. 3store: Efficient bulk rdf storage. In *PSSS*, 2003.

[4] J. Hayes. A graph model for rdf. Master’s thesis, August 2004.

[5] L. Ma, Z. Su, Y. Pan, L. Zhang, and T. Liu. Rstar: an rdf storage and query system for enterprise resource management. In *CIKM 2004*, pages 484–491.

[6] S. Muñoz, J. Pérez, and C. Gutierrez. Minimal Deductive Systems for RDF. In *ESWC2007*, 2007.

[7] Z. Pan and J. Heflin. Dldb: Extending relational databases to support semantic web queries. Technical Report, Lehigh University, 2004.

[8] H. J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2-3):79–115, 2005.

[9] G. Wu. *Research on Key Technologies of RDF Graph Data Management*. PhD Thesis, Tsinghua University, January 2008.

[10] G. Wu, K. Zhang, C. Liu, and J.-Z. Li. Adapting Prime Number Labeling Scheme for Directed Acyclic Graphs. In *DASFAA 2006*, pages 787–796, 2006.