Efficiently Querying RDF Data in Triple Stores

Ying Yan [†], Chen Wang [‡], Aoying Zhou [†§], Weining Qian [§], Li Ma [‡], Yue Pan [‡]

[†] *Department of Computer Science and Engineering, Fudan University*
{yingyan, ayzhou}@fudan.edu.cn

[‡] *IBM China Research Laboratory*
{chwang, malli, panyue}@cn.ibm.com

[§] *Institute of Massive Computing, East China Normal University*
{ayzhou, wnqian}@sei.ecnu.edu.cn

ABSTRACT

Efficiently querying RDF [1] data is being an important factor in applying Semantic Web technologies to real-world applications. In this context, many efforts have been made to store and query RDF data in relational database using particular schemas. In this paper, we propose a new scheme to store, index, and query RDF data in triple stores. Graph feature of RDF data is taken into considerations which might help reduce the join costs on the vertical database structure. We would partition RDF triples into overlapped groups, store them in a triple table with one more column of group identity, and build up a signature tree to index them. Based on this infrastructure, a complex RDF query is decomposed into multiple pieces of sub-queries which could be easily filtered into some RDF groups using signature tree index, and finally is evaluated with a composed and optimized SQL with specific constraints. We compare the performance of our method with prior art on typical queries over a large scaled LUBM and UOBM benchmark data (more than 10 million triples)in [3]. For some extreme cases, they can promote 3 to 4 orders of magnitude.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Query processing; C.4 [**Performance of Systems**]: Design studies

General Terms

Experimentation, Performance

Keywords

RDF, Signature, Graph Partitioning, Indexing

1. INTRODUCTION

The Semantic Web is an effort by the World Wide Web Consortium (W3C) to enable data integration and sharing across different applications. It is designed to evolve the general web which mostly consists of markup or other formats perceived by people into the machine-readable data web. The Semantic Web data model of Resource Description Framework [1] (RDF), recommended by W3C, represents data as a labelled graph connecting resources and their property values with labelled edges representing properties. The graph can be structurally parsed into a set of triples or statements in the form of $< subject, predicate, object >$. RDF data model is very general and is easy to express any type of data. Though RDF data representation is flexible, it potentially results in serious performance issues since RDF queries involve intensive self-joins over the triple table. When the number of triples is so large that they can not be cached in memory for each filter, the joins will be very expensive because disk scan and index lookup are required. As a general data model, RDF triple table stores any type of data in a column. For example, values of age, weight, or even given name of all persons co-exist in the object column. Accordingly, the statistics collected in these columns complicates selectivity estimation, which will further disable relational database query optimizer to deliver a nice query plan.

At present, creating indices, splitting data into property tables (2-column schema), and materializing join views (e.g., subject-subject and subject-object) are common methods for improving RDF query performance on the vertical database structure. However, it is still urgently needed to figure out new storage and indexing schemes that make relational database much efficient to support RDF query. In this paper, we propose a novel idea to optimize executable SQL for general-purposed triple stores by considering graph feature of RDF data. An intuitive example is given as follows to illustrate the idea and our motivation.

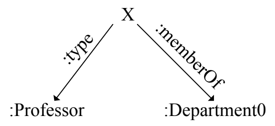
Given a SPARQL [2] query of Figure 1(a) which logically equals to the query graph shown in Figure 1(b), we issue it over the RDF data in Figure 1(c). Suppose that the triples stored in the table are not well sorted intentionally in advance and no index is created as well. After the SPARQL query is translated and submitted to relational database, query engine might determine a nested-loop self-join on the column of subject accordingly. Roughly estimated, the join will cost $4 \times 4 = 16$ times of string comparison and $4 + 4 = 8$ I/O times. Once the query and data are much more complex, the cost will increase dramatically.

Observing the same example from perspective of graph model shown in Figure 2(a), even though the triples (or edges bridging two vertices) like `<:D :Type :Professor>`

Copyright is held by the author/owner(s).
WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

```
Select ?X
Where
{ ?X :type          :Professor,
  ?X :memberOf :Department0,
  }
```

(a) SPARQL Query



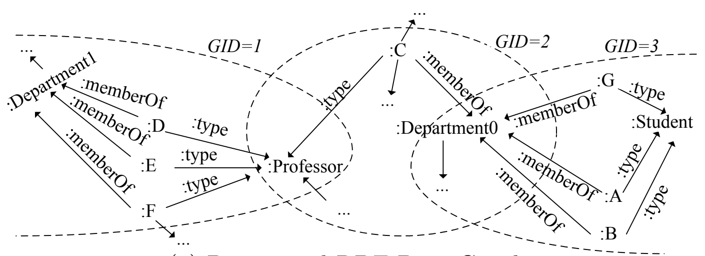
(b) Query Graph

Subject	Predicate	Object
:A	:type	:student
:B	:type	:student
:G	:type	:student
:C	:type	:Professor
:D	:type	:Professor
:E	:type	:Professor
:F	:type	:Professor
:A	:memberOf	:Department0
:B	:memberOf	:Department0
:C	:memberOf	:Department0
:G	:memberOf	:Department0
:D	:memberOf	:Departmen1
:E	:memberOf	:Departmen1
:F	:memberOf	:Department1

Join

(c) Triple Table

Figure 1: A Motivated Example



(a) Partitioned RDF Data Graph

Subject	Predicate	Object	GID
:D	:type	:Professor	1
:E	:type	:Professor	1
:F	:type	:Professor	1
:D	:memberOf	:Department1	1
:E	:memberOf	:Department1	1
:F	:memberOf	:Department1	1
:C	:type	:Professor	2
:C	:memberOf	:Department0	2
:A	:type	:student	3
:B	:type	:student	3
:G	:type	:student	3
:A	:memberOf	:Department0	3
:B	:memberOf	:Department0	3
:G	:memberOf	:Department0	3

Join

(b) Partitioned Triple Table

Figure 2: Our Proposed Idea

are totally disconnected with those including `<:A :MemberOf :Department0>`, `<:B :MemberOf :Department0>` and `<:G :MemberOf :Department0>`, they are still selected to compare with each other by string. Here, we are motivated to decrease the join cost by considering the fact that the query results must be connected subgraphs embedded in the RDF data graph and able to match query graph. Therefore, we first partition the data graph into three groups as shown by the dotted lines in Figure 2(a) and add an additional column for the triple table (Figure 1(c)) to store the group (subgraph) identities illustrated in Figure 2(b). Thus, we only perform the join operations within each group, and finally merge the results together and eliminate redundancy. In this case, only two triples of `<:C :Type :Professor>` and `<:C :MemberOf :Department0>` are filtered as candidates in group 2. The join cost is reduced to $1 \times 1 = 1$ time of string comparison and $1 + 1 = 2$ I/O times. Furthermore, to fast locate the candidate groups probably containing the query graph, we build up signature for all partitions and filter them using the query graph signature.

2. GRAPH PARTITIONING AND INDEXING

We propose a new scheme to store, index, and query RDF data in this paper. Graph feature of RDF data is taken into consideration which might help reduce the join costs on the vertical database structure. We would partition RDF triples into overlapped groups, store them in a triple table with one more column of group identity, and build up a signature tree to index them. Based on this infrastructure, a complex RDF query is decomposed into multiple pieces of sub-queries which could be easily filtered into some RDF groups using signature tree index, and finally is evaluated with a composed and optimized SQL having specific constraints. In [3], we compare the performance of our method with prior art on typical queries over a large-scale of LUBM and UOBM benchmark data (more than 10 million triples). The experimental results in [3] show that our method can help dramatically improve query efficacy of triple stores. For some extreme cases, it can promote 3 to 4 orders of magnitude. We highlight our contributions as below:

Graph partitioning for RDF triples Filtering and joining of each RDF query always happen within the smaller scope than the whole RDF graph. It would dramatically reduce the self-join cost.

Signature indexing Bit vector matching instead of string matching between query graph and data graph is used to fast filter out candidate partitions, which greatly save the cost on locating partitions.

SQL rewriting We slightly change the schema of the triple table and only rewrite the translated SQLs by common RDF triple stores with our techniques, which can be easily integrated as an optimization component into varying types of triple stores.

Early stop strategy Null result can be returned immediately without the need of consulting RDBMS, once while no candidate partition is found to potentially contain query graph by signature matching. The query evaluation is finished outside relational database.

3. REFERENCES

[1] F. Manola and E. Miller. RDF primer. W3C recommendation, Feb 2004.
[2] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C candidate recommendation, April 2006.
[3] Y. Yan, C. Wang, A. Zhou, W. Qian, L. Ma, and Y. Pan. Efficiently querying rdf data in triple stores. Technique report, 2008.