Scalable Online Analysis of Semantic Web Data

Pingpeng Yuan, Pu Liu, Buwen Wu, Delong Wu, Huizhong Zhong, Hai Jin

Service Computing Technology and System Lab,
Cluster and Grid Computing Lab,
School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@mail.hust.edu.cn

Abstract. In order to explore RDF data in decision-making, there is an increasing demand for online analytical processing of such data. Since RDF data model is a graph model and highly flexible, it poses some challenges for analytical tasks. Although many systems have been developed to store RDF data, few of them can perform analytical tasks due to many reasons. Here, we present an online analysis system - DBLink for RDF data. DBLink can be united for enabling large-scale RDF data analysis. Since SPARQL does not have language elements to support analytical operations, we design a language DQE in our RDF data store. Experimental results show the system achieves a performance improvement in analytical tasks than MySQL. Especially, the system greatly improves the execution time of those queries which need to analyze a large sub-graph of RDF data set.

1. Introduction

More and more organizations, such as *White House* and *New York Times* are making their data available on the Web. And more techniques for annotating Web content with metadata are available. Thus, the Semantic Web or the Data Web enhanced by RDF-based representations is growing rapidly. The data web shows many interesting interactions between entities including persons, revealing huge network structures. For example, collaboration network among researchers indicate who co-author papers and which pairs have the largest collaboration frequency. People's relationships online are formed through people profiles described by FOAF. In fact, many more networks are emerged by the help of Web or Semantic Web. In above networks, not only individual entities but also the interacting relationships among them are important and interesting. Researchers need to analyze the networks in order to answer the following questions; "Which are top *n* papers referred mostly?" "Which conferences/journals are mostly related with a conference/journal?" etc. The information required to answer those queries are formed as a graph. This demands a tool or a system that can manage and analyze such data efficiently.

It is interesting to analyze the semantic web data in an OLAP manner. However, traditional OLAP mainly focus on supporting the data cube processing tasks with the limited support for graph structured data such as RDF. The structure of the RDF data poses some challenges for analytical tasks. First, RDF data generally also include

schema data, which describe properties and classes of RDF resources, with semantics for generalization-hierarchies of such properties and classes. Query analyzer need to understand schema in order to execute queries correctly. Secondly, how are RDF data stored efficiently for performing analytical queries. RDF data are consisted of many individual triples, each of which is a statement. There may be many statements about an entity which are independent. In order to execute queries on RDF data, join operations are required to assemble the statements and then multi-pass aggregations need to be computed, thus making the execution of these queries inefficient. Further, analytical queries are cumbersome, even impossible to express using SPARQL because the language does not support grouping and aggregations operations, which are very common in analytical tasks.

Here, we present an Online Analysis Oriented Semantic Web Data Storage System – DBLink which is designed with fast low level graph operation primitives as well as a query interface designed for high level graph based queries. If a single system of DBLink cannot process large scale RDF data due to physical limitation from computers, several systems of DBLink can be united to answer users' queries.

The remainder of this paper is organized as follows. In section 2 the overview of DBLink are presented. Section 3 describes the query language of DBLink. The experiments and the evaluation of the performance of DBLink are presented in Section 4. Finally, conclusions are given in Section 5.

2. Overview of DBLink

In order to process RDF data in an OLAP manner, we make the following design decisions:

Store, operate and query RDF data in main memory. We choose to design a main-memory based RDF triple store for two reasons: Firstly, due to the advances in memory technology, the memory size is bigger and bigger. Secondly, main memory stores provide high throughput and fast response time. If computer memory places physical constraint on our system, several systems of DBLlink, each of which manages a subset of data set can be united to execute queries. Thus, our system can be extended according to data volume.

Grid partition and regroup RDF data. Different from triple store or vertically partitioning, DBLink partition RDF triples according to schema information, such as type of class and property. The approach partitions the RDF data vertically and horizontally. We name this partition as Grid partition. Then DBLink regroups triples of multi-valued property type. By doing that, redundancy can be reduced, and certain triple retrieval and aggregation operations can be executed much faster.

Use a compact storage structure. As triples contain string literals which require large storage space, we adopt the approach of replacing all literals by *id*s using a mapping dictionary, where strings are id encoded.

The triples are sorted lexicographically by ID after grouping columns by predicate and are directly stored in chunks, fixed-size storage spaces. Although this means of data organization is similar with vertically partitioning [1] in some sense, we partition RDF data into small fixed-size chunks. So we would not have potential

scalability problems when the size of tables varied. Moreover, we adopt delta compression to compress data since the collation order may cause values of neighboring triples to be similar.

To gain fast access to records, different from many triple stores, such as RDF-3X which maintain all six possible permutations of S, P and O in six separate indexes [2], we only need keep two permutations of S, O. So, the total size for indexes is also much less than the size the general triple store systems require for indexes. We adopt hash to index the chunks. We compute the hash value of id and find the chunks that contain the largest search-key value. Moreover, storing triple matrix according to predicate also provides some kind of index.

Schema-aware RDF storage system, RDF data can be divided into instance data and schema data. RDF schema (RDFS/OWL) data and instance data have different characteristics. DBLink adopts a schema aware approach rather than schema oblivious approach in which all data including schema data are stored together. In DBLink, for the support of RDFS, four kinds of instance tables: *Class*, *ObjectProperty*, *DatatypeProperty* and *Datatype* for storing corresponding instances are designed. Each record of the instance tables points to the instance table which maintains the instances of the corresponding resource types.

Optimized for read mostly scenarios. Current semantic web applications rely on querying, inference or analysis of RDF data heavily, which are all read mostly operations. Write operations, such as inserting and updating are relatively rare or periodic. Delete operation can be done in a lazy deletion and periodic clean way. DBLink adopts grouping multi-valued attributes, and indexes etc. for optimizing the performance under read mostly scenarios.

Provide a low-level API to support basic graph operations. So, user can construct high level graph based algorithms using the API, such as graph searching, matching, link analysis and so on. Some graph operations must be implemented really fast, such as to get all the successor of a node with specified edge type.

Provide a high-level query interface, which supports structured queries for extracting data from the RDF repository, and provide some special syntaxes for search function and graph based retrieval or analytical queries.

3. Query language

The World Wide Web Consortium (W3C) recommends SPARQL for querying RDF data. However, SPARQL is not expressive enough to meet requirements for queries, such as result aggregation and path computation which are missing from the standard SPARQL definition [5]. Considering the requirements of analytical processing in graph, we designs a graph based query language - DBLink Query Expression (DQE) in addition of support of SPARQL. The DBLink Query Expression (DQE) uses a simple graph based syntax. Currently several basic operators in DQE are implemented as shown in Table 1. Complex queries can be constructed using these basic operators.

The execution process of a common DQE starts form a set of staring nodes, then expands those starting nodes with paths described by expand operators, and filter

those nodes by filtering operators, or perform special operations like aggregation on those nodes or paths. The expanded nodes are now considered new starting nodes, and is passed to a new phase of query execution. Finally, the whole expanded subgraph is returned as query results.

Table 1. Basic operators of DQE

Operator	Notation	Description
Select	*property*	Retrieval specific property values of the start nodes, use property name as notation
Count	C	Count triples of specific property whose subjects are start nodes
Filter	F	Filter the start nodes by various conditions
Expand	E	Expand start nodes by graph edges of specific property type
Expand	N	Expand start nodes by graph edges of specific property type
Closure	L	Expand start nodes by graph paths consisted by specific property type
Connect	*	Binary operator, pass the expanded nodes of left DQE to the right DQE as start nodes
Plus	+	Binary operator, union the two query results and connect the tow expanded node list
Self	1	Do nothing, just return input as output
Search	S	No input, full text search against the specific property value of instances of specific class type

4. Evaluation

Below, we provide the performance details when our system executes SPARQL queries and performs analytical tasks on data extracted from the BTC2010 dataset and other data sources.

4.1 SPARQL query evaluation

We chose MonetDB [1] and RDF-3X (v0.3.3) [2] for our evaluation, as they show better performance than others. The recent emerging system BitMat [3] has a high performance on query execution as said in paper [3]. However, due to its public unavailability, we have to ignore it in the experiments.

We have done the experiments on a server, the configuration of which is as follows: Intel Xeon CPU E7420 4 Way 4-core 2.13GHz, 64GB memory; Red Hat Enterprise Linux Server 5.1 (2.6.18 kernel). Because no queries for BTC2010 data set are available, we generated a dataset using LUBM [4] which is widely used by the Semantic Web community for benchmarking triple stores. All the queries are listed in Appendix A.

To account for caching, each of the queries is executed for five times consecutively. We took the best result to avoid artifacts caused by OS activity. For warm caches we ran the queries five times without dropping caches, again taking the best run-time. Experimental results are shown in Table 2, 3. Note that our current DBLink does not use any sophisticated cache management and also does not employ

high performance join technologies. Due to this, as opposed to RDF-3X and MonetDB, in the queries the difference between our cold and warm cache times is not high, most of them are almost same. According to the results, for Q1, Q2, Q4, Q5, Q6, Q7, DBLink excelled over RDF-3X. In cold cache cases. In warm cache cases, DBLink excelled over RDF-3X for Q1, Q2, Q5, Q7. In all cases, DBLink performs better than MonetDB.

Table 2. LUBM 69M Triples (time in seconds, best times are boldfaced)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Cold caches							
RDF3X	0.289	0.036	**2.061**	5.051	111.212	5.150	0.126
MonetDB	7.924	2.665	36.090	20.501	132.502	8.440	0.013
DBLink	**0.001**	**0.002**	2.688	**3.854**	**7.602**	**4.950**	**0.002**
Warm caches							
RDF3X	0.0017	0.0035	**1.374**	**1.893**	111.141	**4.036**	0.016
MonetDB	0.139	0.0029	21.625	16.245	124.007	8.101	0.005
DBLink	**0.0009**	**0.0021**	2.701	3.697	**5.934**	4.855	**0.002**

Table 3. LUBM 1000M Triples (time in seconds, best times are boldfaced)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
Cold caches							
RDF3X	0.489	0.371	**43.633**	92.486	>1500	158.77	1.579
DBLink	**0.012**	**0.0045**	47.9	**76.3**	**184.7**	**131.46**	**0.0029**
Warm caches							
RDF3X	0.0017	0.0035	**34.006**	**40.766**	>1500	**89.247**	0.265
DBLink	**0.0011**	**0.0023**	45.3	65.18	**123.4**	114.52	**0.0027**

4.2 Analytical query evaluation

The dataset used in the experiments are consisted of literatures which are mainly extracted from BTC2010 dataset. Other sources, such as Libra, CiteSeer and Google Scholar are also extracted. The detail information about the data set is listed in table 4.

Now, few RDF stores excluding RDBMS can execute analytical queries because SPARQL does not support analytical operations as said in section 3. And many RDF storage systems generally store RDF data in a RDBMS, such as MySQL. Thus, MySQL are chosen to compare in the experiments because MySQL is considered one of the fastest RDBMS. In the experiments, MySQL version 5.1.29 for Linux x86_64 is used. All the tables use MyISAM storage engine. DBLink is compiled using GCC 3.4.6, with optimization parameter -O2. The experiments are performed on the same server as the previous experiments.

The experiments are consisted of 5 queries (see Appendix B), which require to analyze large scale graph, and each is executed 10 times. The query execution time for all 5 queries and the average query execution time of 5 queries on both MySQL and DBLink are shown in Fig. 1. All execution time presented in Fig. 1 is the average of 10 runs of the queries. All queries on MySQL are executed many times so that the

data to answer queries are cached in main memory. Thus, the disk access time of MySQL can be ignored.

Table 4. Data set for analytical query

Item	Number
Articles in Journal	386481
Articles in Proceeding	613760
Book chapters	4718
Conference/Journal/Book	13009
Persons	615416
Citations	3665053
Terms	108983
Categories	6676
URIs	1750283

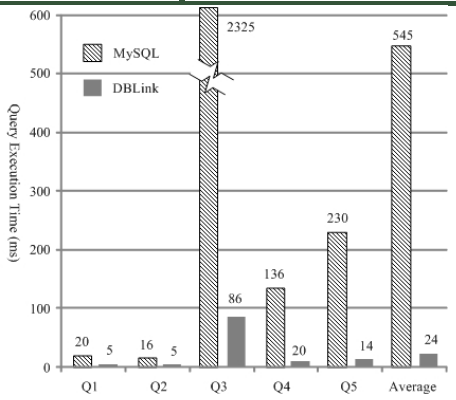

Fig. 1. Query execution time comparison of MySQL and DBLink

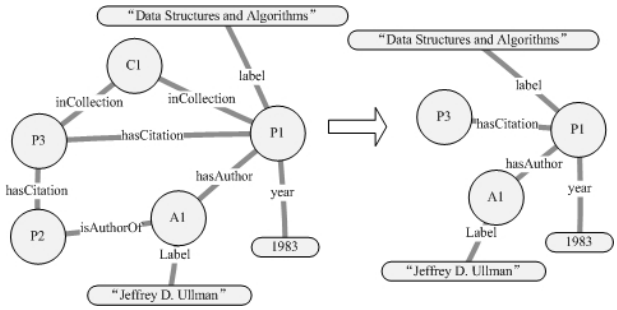

Fig. 2. Subgraph of conferences and publications related to Q3

According to Figure 1, DBLink performs a factor of 4-27 times faster than MySQL, and the average speed up is 22.7. As shown in Figure 1, DBLink gets better speed up factors comparing with MySQL when the query needs more time to execute. For example, Q3 is to query conferences or journals which are mostly related to a

conference or journal. The subgraph related to Q3 is shown in Figure 2. There are hundreds, even thousands of publications in conferences/journals, and each publication usually cites or is cited by tens, even hundreds of other publications. So the subgraph Q3 needs to analyze usually has thousands of edges. However, the subgraph related with Q1 is relatively small because an author usually published tens to hundreds of publication. So DBLink gains more speed up factor on Q3 than on Q1.

5. Conclusions

With the rapid growing of RDF data volume, it is essential to develop the systems to process, specially analyze large scale RDF data efficiently. DBLink was designed to enable large-scale RDF data analysis. DBLink shows high performance in performing analytical tasks and SPARQL queries. The mainly contributions of the paper are: 1) we present an Online Analysis Oriented Semantic Web Data Storage System which can analyze the semantic web data in an OLAP manner. 2) Considering the requirements of analytical workloads in RDF graph, we design an expression language - DQE. 3) We design an optimized storage structure of RDF data which is suitable for analytical tasks..

References

1. Abadi, D. J., Marcus, A., Madden, S. R., Hollenbach, K., Scalable Semantic Web Data Management Using Vertical Partitioning. In Proc. of VLDB 2007, pp: 411-422
2. Neumann, T., Weikum, G.: The RDF-3X Engine for Scalable Management of RDF Data. The VLDB Journal, 2009, DOI: 10.1007/s00778-009-0165-y
3. Atre, M., Chaoji, V., Zaki, M. J., Hendler, J. A.: Matrix "Bit"loaded: A Scalable Lightweight Join Query Processor for RDF Data, In: WWW 2010, pp. 41-50. ACM Press(2010)
4. LUBM. http://swat.cse.lehigh.edu/projects/lubm/
5. Erétéo, G., Buffa M., Gandon F., Corby O., Analysis of a Real Online Social Network using Semantic Web Frameworks, In Proc. Of 8th International Semantic Web Conference, 2009, pp:177-192

Acknowledgments. This research was supported by National Science Foundation of China (61073096) and Graduate Innovation Fund of Huazhong University of Science & Technology (HF-07-01-2010-210).

Appendix A: Seven SPARQL queries from LUBM

Q1: SELECT ?x WHERE { ?x ub:subOrganizationOf <http://www.Department0.University0.edu> . ?x rdf:type ub:ResearchGroup .}
Q2: SELECT ?x WHERE {?x ub:worksFor <http://www.Department0.University0.edu> . ?x rdf:type ub:FullProfessor . ?x ub:name ?y1 . ?x ub:emailAddress ?y2 . ?x ub:telephone ?y3 .}

Q3: SELECT ?x ?y ?z WHERE { ?x rdf:type ub:UndergraduateStudent . ?y rdf:type ub:University . ?z rdf:type ub:Department . ?x ub:memberOf ?z . ?z ub:subOrganizationOf ?y . ?x ub:undergraduateDegreeFrom ?y . }
Q4: SELECT ?x WHERE { ?x rdf:type ub:Course . ?x ub:name ?y . }
Q5: SELECT ?x ?y ?z WHERE { ?z ub:subOrganizationOf ?y . ?y rdf:type ub:University . ?z rdf:type ub:Department . ?x ub:memberOf ?z . ?x rdf:type ub:GraduateStudent . ?x ub:undergraduateDegreeFrom ?y . }
Q6: SELECT ?x ?y ?z WHERE {?y ub:teacherOf ?z . ?y rdf:type ub:FullProfessor . ?z rdf:type ub:Course . ?x ub:takesCourse ?z . ?x rdf:type ub:UndergraduateStudent . ?x ub:advisor ?y .}
Q7: SELECT ?x ?y WHERE {?y ub:subOrganizationOf <http://www.University0.edu> . ?y rdf:type ub:Department . ?x ub:worksFor ?y . ?x rdf:type ub:FullProfessor .}

Appendix B: five benchmark queries for analytical tasks.

Q1: query top-10 cooperators of an author
DQE: $start*N$isAuthorOf*[$hasAuthor GroupByCountDesc 10:group]
Q2: query top-10 conferences or journals which are related with research interest of an author
DQE: N$isAuthorOf[$inCollection GroupByCountDesc 10:group]
Q3: query top-10 conferences or journals which are related with a conference or journal
DQE: N$hasPublication(N$hasReference+N$hasCitation)[$inCollection GroupByCountDesc 10:group]$label
Q4: query top-10 active authors in a conference or journal
DQE: N$hasPublication[$hasAuthor GroupByCountDesc 10:group]
Q5: query： count yearly reference of an author
DQE: N$isAuthorOfN$hasCitation[$year GroupByValue:filter]

Appendix C: outline of the system

Categories	Requirements or Features	Overview of the System
Requirements	Make use of the BTC2010 dataset	For the purpose of presenting an interesting application of the BTC2010 dataset, the system mainly analyzes the data about literature from the dataset which are also stored in our systems.
	Allowed to use other data.	Although a small amount of data from the other sources, such as Libra, CiteSeer and Google Scholar are also extracted, we focus on BTC2010 dataset.
	Usability	Our system analyzes dataset and visualizes the analysis results and can answer queries such as top-*n* authors/conferences/journals/co-authors etc.
Additional Desirable Features	more than store/retrieve triples	Our system not only stores/retrieves large numbers of triples, but also analyzes dataset and visualizes the analysis results.
	scalable	When a single system of DBLink can not fulfill the tasks due to physical limitation from computers, several systems can be united for enabling very large-scale RDF data analysis.
	use of the very large, mixed quality data set	The experimental results in section 4 show our system can process very large, mixed data set.
	either function in real-time or have a real-time realization	When performing analytical tasks on literature, the system can respond users' query in real-time. If the data volume is larger, the system also responds quickly although the response time may be longer.
	Web Site	http://202.114.10.134:8080/DBLink/ or http://www.semrex.cn/DBLink