A Semantic Data Fragmentation Approach for RDF Data Management

Meisam Booshehri, Peter Luksch
Institute of Computer Science
University of Rostock
Rostock, Germany
e-mail: {firstname.lastname}@uni-rostock.de

Abstract— **Efficient management of Resource Description Framework (RDF) data is one of the significant factors in realizing the semantic web vision. However, current RDF data management systems scale poorly, having performance limitations. In this PhD work, a new kind of data fragmentation in the context of RDF data is proposed based on the idea of ontology modularization. The proposed approach indicates dividing an ontology into several modules, applying RDF storing methods on ontology modules rather than on the whole ontology. By using this approach, three contributions can be introduced as follows. First, it will reduce the amount of data to be worked on at any specific point of time in order to achieve less load time and higher performance. Second, it will provide some kind of improved locality that reduces the need for interaction across the nodes of a distributed system, resulting in less message traffic. Third, according to the nature of data fragmentation we will expect higher concurrency as well. In order to show the feasibility of the approach, the main components of a suitable architecture is proposed and discussed in detail. For the evaluation, we intend to implement our proposed architecture as a layer over existing prominent open source storage systems to support the proposed fragmentation and verify the contributions. The proposed metrics would be query-time and system throughput. The former is expected to decrease while the latter is expected to increase.**

Keywords-RDF data; The semantic web database systems; Data fragmentation; Ontology modularization; Concurrency; Load time; Data traffic on the network; Performance.

I. INTRODUCTION

In order to realize the semantic web vision, it is essential to provide high-performance and scalable solutions for RDF data storage and retrieval. On the other hand, current state-of-the-art solutions have yet to be improved regarding the tremendous influx of RDF data. Current state-of-the-art methods that can be used for RDF storage and indexing could be classified into four categories:

1- *Relational Schemes*, which use Relational Database Management Systems (RDBMS) for storing RDF data.
2- *Native Schemes*, which build RDF-specific stores and indexes from scratch.
3- *Not Only SQL (NoSQL) database systems*, which are not built primarily on tables, and generally do not use SQL for data manipulation[1].
4- *Hybrid storage approaches*, which originally are aimed at the integration of NoSQL systems (such as Hadoop) with relational database technology in order to make an analytical platform for Big Data [2][3][4]. Obviously, this approach can be used for storing RDF triples.

As for the RDF data, native schemes perform well because of their tailored design, which makes the reasoning process over the semantic data easier and more straightforward. This is because of eliminating the need for some extra processes during the query process, such as query rewriting and the transformation of data to a suitable semantic format; however, relational schemes are preferred yet from the perspective of maturity, generality and scalability [5]. On the other hand, NoSQL database systems are generally more scalable than the relational database systems while NoSQL systems have some disadvantages including lack of ACID (Atomicity, Consistency, Isolation, Durability) properties and lack of SQL support. Consequently, the hybrid storage approaches have emerged as aforementioned. This evolution shows the significance of relational database technology insofar as they are being integrated into new technologies, such as NoSQL systems. Consistently, in this research we are exploring new ways for improving both for relational schemes and hybrid storage approaches.

One of the key questions in this context is the following: "How should we design tables for storing RDF triples?" The most well-known storage methods for row-oriented relational database systems are Horizontal Table [6], Vertical Table [7][8], Horizontal Class [9], Table per Property [9] and Hybrid Designs [9]. As for the column-oriented relational database systems, several prominent storage and indexing techniques have been proposed, including vertically partitioning method [10] and sextuple indexing technique [5][11], which beats row-oriented methods in terms of performance according to the recent experiments [12]. The common characteristic among all the above-mentioned methods is that they all are applied to the whole ontology data.

In this study, we specifically intend to explore the effect of a new semantic data fragmentation approach for storing RDF triples, which is elementally based on ontology modularization. As maintaining large ontologies is a difficult task and reusing the whole ontology is time-consuming and costly, the notion of an *ontology module* has been proposed.

[13]. The assumption we consider as a basis for our discussion in this paper is that “a module is considered to be a significant and self-contained sub-part of an ontology” [14]. Therefore, the vocabulary of an ontology module is a sub-set of the whole ontology vocabulary. And a module would represent a smaller ontology plus inter-module links. Moreover, a module is considered to be self-contained whenever reasoning tasks over a module can be done within the module without having accessing to other modules[15][16].

Overall, the hypothesis we are going to verify is the following: *we could use the ontology modules as the database design basis in the* ***Relational Data base Management Systems*** *or in the* ***Hybrid Storage Systems*** *instead of considering the whole ontology in order to decrease the amount of data to be worked on at any specific point of time. This will result in increasing concurrency and performance and reducing the message traffic on the network at the same time. This approach is considered as a new type of data fragmentation in the context of RDF data management systems.*

The rest of the paper is organized as follows. The second section is to review the background and some popular related works. Next, in the third section our proposed approach is described. Then, in the fourth section a customized evaluation design is proposed, and the expected results are discussed. Finally, the fifth section is to present the conclusion.

II. RELATED WORK

We categorize the related work to this PhD thesis into four groups: ontology modularization strategies, Criteria for ontology modularization, modularity and databases, and ontology based data access systems. A detailed discussion of each category comes in the sections below.

A. Ontology Modularization Strategies

According to Parent and Spaccapietra, Ontology modularization strategies fall into three classes: Semantics-Driven Strategies, Structure-Driven Strategies, and Machine Learning Strategies [17]. There are also another classifications and interpretations regarding ontology modularization, including logic-based approaches and Graph theory based approaches [14][15][17][18][19][20][21][22][23][24]. However, all the classifications fall into the categories introduced by Parent and Spaccapietra on which we draw mainly in our whole research. Semantics-Driven Strategies let the ontologies be driven by the semantics of the application domain. This method relies on human expert knowledge regarding the application domain while the responsibility of the machine is usually limited to recording the allocation of knowledge items to the modules. Structure-driven strategies, on the other hand, do not rely on the human input. These methods look at the ontology as a graph structure and use graph partitioning techniques to extract ontology modules. Machine learning strategies establish another category for ontology modularization, which is considered as an alternative to human-driven modularization. In this approach, a combination of machine learning techniques can be used for knowledge processing in order to extract the ontology modules.

B. Criteria for Ontology Modularization

According to Mathieu d’Aquin et al. [14], there are different criteria for modularization, including logical criteria and structural criteria.

Logical criteria can be expressed in terms of local correctness and local completeness. Local correctness states that every axiom being entailed by the module should also be entailed by the original ontology, meaning that nothing has been added in the module that was not originally in the ontology. Local completeness, on the other hand, indicates the reverse property of local correctness.

Structural criteria include some measures like the the size of a module and the intra-module distance. Indeed, the relative size of a module (number of classes, properties and individuals) has a strong effect on its maintainability and, therefore, on the robustness of the applications relying on it. The intera-module distance is another important structural measure, which computes how the terms described in a module move closer to each other compared to the original ontology, for instance, by counting the number of relations in the shortest path from one entity to the other.

C. Modularity and Databases

Abadi et al. propose vertically partitioned method for storing RDF triples in a column-oreinted relational database system where they have observed that the query-time have dropped from minutes to several seconds [12]. Accordingly, Booshehri et. al. propose the vertically partitioned module method for the column-oriented databases in order to achieve better performance by creating the tables based on ontology modules [25]. The perspective proposed by Booshehri et. al's approach is the most related work to this PhD thesis. However, the new perspective described in this PhD work is a thoroughly refined idea of Booshehri et. al's approach. In contrast to Booshehri et. al's, the new approach described in this paper is a more generalized approach, which is not limited to relational database management systems and can be adapted with different database systems, including NoSQL systems, native schemes for RDF storage, and hybrid storage systems as well. In this research, however, we focus on relational schemes and hybrid schemes. As further is discussed, this is going to be realized by implementing different ontology based data access systems.

D. Ontology Based Data Access Systems

Ontology based data access (OBDA) is a technology for mapping a relational database into an ontology so that we can answer queries over the target ontology. Currently, there are two approaches for implementing an OBDA [26]: query rewriting and materialization. In the materialization approach, the input relational database is used to derive new facts based on an ontology and a set of mapping rules; then, it will be stored in a new database, which is the materialization of the data in the first database.

Sequeda et. al [26] provide a new OBDA system called Ultrawrap^{OBDA}, which combines the query rewriting and materialization approach. This combinatorial approach has been shown to achieve better performance comparing against another prominent OBDA system, namely Ontop [27]. On the other hand, to the best of our knowledge, most of OBDA systems aim at mapping a relational database into an ontology while a question will still remain open: How could we design an OBDA system for a hybrid storage system, such as Hadapt [28], which provides the capability of running SQL queries over Hadoop. Therefore, we have proposed the notion of an OBDA system for hybrid storage systems, as further is discussed in the next section.

III. Proposed Approach

The main idea of the proposed approach is to make use of ontology modularization as a semantic data fragmentation. Consequently, we expect less load time and higher performance, higher degree of concurrency and system throughput, and less message traffic on the network. We discuss these objectives in detail in the next sections.

We are motivated to design and implement our proposed approach as a layer over existing traditional RDBMSs and Hybrid Storage Systems. Accordingly, the main components of an architecture, which can show the feasibility of the approach is proposed and discussed in detail. First, a component is needed to create a *partitioned schema* based on the ontology modules instead of the whole ontology. Next, an *OBDA* system should be provided to convert the queries into queries over the new partitioned schema. Finally, a data fragmentation unit should be provided in order to fragment the ontology data according to the portioned schema. We discuss these components in more details in the sections below.

A. Schema Partitioning Component

The *Schema Partitioning component* converts the original schema which is based on the whole ontology into a partitioned schema which is based on the ontology modules. For the proposed system, we intend to provide two options for the end users. The first option is to introduce the ontology modules to the system manually and the second option is to make use of prominent approaches for automatic ontology modularization.

B. OBDA Unit

When the database schema is converted into a partitioned schema, consequently, a query rewriter should be provided in order to convert the original SQL queries into queries over the new partitioned schema. As discussed in the related work section, materialization is another approach for implementing an OBDA system, which also can be used in combination with the query rewriting techniques. We intend to design optimized OBDA systems, which support embedding our fragmentation approach into both relational database systems and hybrid storage systems. Moreover, we aim at implementing an OBDA system, which combines query rewriting and materialization in order to achieve better performance.

Regarding the partitioned schema, two types of properties can be defined for ontologies: *intra-module properties* and *inter-module properties*. Intra-module properties refer to those which are only related to the concepts and individuals within an ontology module and inter-module properties are those which connect couples of concepts or individuals from different ontology modules. It is obvious that we may have both of these two types of properties within an ontology. Accordingly, considering this classification the queries also can be classified into two categories which are *intra-module queries* and *inter-module queries*. An intra-module query is applied only on the data and ontology elements within a specific module. On the other hand, an inter-module query is applied on the information and ontology elements that connect different ontology modules. Of course, an inter-module query could be a combination of some intra-module queries as well as some inter-module queries.

Considering these classifications, whenever a query is applied to a database, the OBDA unit is responsible to recognize whether the query is inter-module or intra-module.

C. Data Fragmentation Unit

Now that we have a portioned schema, the ontology data should be fragmented and allocated to different workstations in the network.

As for a RDBMS, we have to redesign the tables according to the extracted modules. Then, it would be the responsibility of the OBDA unit to reason over the fragmented database.

In case of exploiting a hybrid storage system, the responsibility of the data fragmentation unit would be generating specialized map-reduce functions in order to fragment the data according to the ontology modules. Then, the OBDA unit will be responsible for reasoning over the database.

IV. Research Objectives and Discussions

There are three main objectives for this research:

A. Less load time and higher performance.

The proposed approach emphasizes on ontology modules as the database design basis. It is obvious that the number of extracted tables from an ontology module is less than the number of extracted tables from the whole ontology. Consequently, existing data in the tables of an ontology module is less than existing data in the corresponding tables of the whole ontology. It means that focusing on modules instead of the whole ontology, may result in a decrease in the size of the information to be worked on at any specific point of time. Hence, we expect ontology modularization to cause less load time and higher performance for column-oriented RDBMSs, row-oriented RDBMSs and Hybrid Storage Systems.

B. Increasing the degree of concurrency and system throughput.

As previously mentioned, in this research module extraction is considered as a new type of data fragmentation in the context of RDF database systems. Therefore, the more precise the module extraction algorithms are the more suitable semantic data fragmentation we have. Naturally, increasing the degree of concurrency and system throughput are two important benefits of data fragmentation in distributed databases [29][30]. Therefore, module extraction and use of ontology modules as database design basis is expected to make us closer to these two benefits. Considering the self-contained feature of the ontology modules, dividing ontologies into modules is a justifiable data fragmentation.

On the other hand, there are two important disadvantages for data fragmentation as follows:

- If there are some requirements which are in conflict with data fragmentation, the performance would decrease. For instance it is costly to retrieve several different parts of data that must be joined or unioned from different sites [30].
- During data fragmentation some attributes that is related to an association relationship may be separated into several parts and located in distinct sites. This will cause the problem of difficulty in semantic control of data and difficulty in integrity control as well [30].

However, according to the self-contained feature of an ontology module, we can say that the problems mentioned above are not serious about ontology modules.

C. Reducing the message traffic on the network with respect to intelligent allocation of data to the cluster nodes in distributed systems.

As discussed before an ontology module is self-contained meaning that special reasoning tasks such as inclusion relation or query answering within a module are possible without need to access other modules. Concerning the self-contained feature of an ontology module, it seems that allocating the data of each ontology module to a single cluster node in distributed database systems is an intelligent allocation that brings us less message traffic on the network over a specified period. This is because of the majority of intra-module queries in comparison to inter-module queries. The fewer inter-module queries leads to less message traffic between cluster nodes on the network.

V. EVALUATION

To formulate the evaluation methodology we will consider the following tasks:

1. Design a suitable benchmark to generate large-scale datasets based upon a big ontology like SWEET ontology. SWEET Ontology [31] is a highly modular ontology containing more than 200 modules.
2. Design several benchmark queries that cover all important RDF join patterns.
3. Selecting a storage system. It could be an open source column-oriented RDBMS, an open source row-oriented RDBMS, or a hybrid storage system.
4. Implementing an OBDA system so that we could implement our approach and reason over the selected ontology.
5. Evaluation of the proposed approach in terms of performance (query time) and system throughput in several working periods of the system.

After performing the above mentioned steps we expect to have the following outcomes:

1. Decrease in the running time of queries
2. Increase in the system throughput over specified working periods.

In case of achieving the expected outcomes, we will replace the first step of the evaluation methodology with an alternative step in which instead of selecting an ontology which has specified modules at the beginning, we will divide an ontology into modules automatically by using both structure driven strategies and machine learning techniques in order to test the effect of ontology modularization algorithms on the achieved outcome.

VI. CONCLUSION

Tremendous influx of RDF data calls for highly scalable and high-performance storage methods which is essential for realizing the semantic web vision. In this proposal, we are investigating the answer to the following question: “Could we improve current state-of-the-art methods for RDF storage by using ontology modules as the database design basis instead of considering the whole ontology? “

We consider the process of dividing large ontologies into modules as a kind of semantic data fragmentation. Based upon this perspective, a general architecture is proposed in order to show the feasibility of the approach. The fragmentation approach is not limited to one kind of storage system; however, we deepen our research by focusing on relational schemes and hybrid storage schemes. Next, we will try to improve them in terms of performance, concurrency and data traffic on the network.

As for the evaluation of the proposed approach, we have suggested an evaluation methodology in which different aspects of the proposed approach are verified thoroughly.

REFERENCES

[1] A. Bialecki, M. Cafarella, D. Cutting, and O. O'MALLEY, "Hadoop: a framework for running applications on large clusters built of commodity hardware," Wiki at http://hadoop.apache.org, vol. 11, 2005.

[2] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads," Proceedings of the VLDB Endowment, vol. 2, 2009, pp. 922-933.

[3] K. Bajda-Pawlikowski, D. J. Abadi, A. Silberschatz, and E. Paulson, "Efficient processing of data warehousing queries in a split execution environment," in Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, 2011, pp. 1165-1176.

[4] A. Abouzied, D. J. Abadi, and A. Silberschatz, "Invisible loading: access-driven data transfer from raw files into database systems," in Proceedings of the 16th International Conference on Extending Database Technology, 2013, pp. 1-10.

[5] X. Wang, S. Wang, P. Du, and Z. Feng, "Storing and Indexing RDF Data in a Column-Oriented DBMS," in DBTA, 2010, pp. 1-4.

[6] R. Agrawal, A. Somani, and Y. Xu, "Storage and querying of e-commerce data," in VLDB, 2001, pp. 149-158.

[7] D. BeckettandJ. Grant, "Swad-europe deliverable 10.2: Mapping semantic web data with rdbmses," W3C Semantic Web Advanced Development for Europe (SWAD-Europe), 2003.

[8] M. Stonebraker, *et al.*, "C-store: a column-oriented DBMS," in Proceedings of the 31st international conference on Very large data bases, 2005, pp. 553-564.

[9] Z. PanandJ. Heflin, "Dldb: Extending relational databases to support semantic web queries," DTIC Document2004.

[10] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach, "Scalable semantic web data management using vertical partitioning," in Proceedings of the 33rd international conference on Very large data bases, 2007, pp. 411-422.

[11] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: sextuple indexing for semantic web data management," Proceedings of the VLDB Endowment, vol. 1, 2008, pp. 1008-1019.

[12] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach, "SW-Store: a vertically partitioned DBMS for Semantic Web data management," The VLDB Journal—The International Journal on Very Large Data Bases, vol. 18, 2009, pp. 385-406.

[13] A. SchlichtandH. Stuckenschmidt, "A flexible partitioning tool for large ontologies," in Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on, 2008, pp. 482-488.

[14] M. d'Aquin, A. Schlicht, H. Stuckenschmidt, and M. Sabou, "Criteria and evaluation for ontology modularization techniques," in Modular ontologies: Springer, 2009, pp. 67-89.

[15] P. Doran, "Ontology reuse via ontology modularisation," in KnowledgeWeb PhD Symposium, 2006.

[16] B. Konev, C. Lutz, D. Walther, and F. Wolter, "Logical Difference and Module Extraction with CEX and MEX," in Description Logics, 2008.

[17] C. ParentandS. Spaccapietra, "An overview of modularity," in Modular Ontologies: Springer, 2009, pp. 5-23.

[18] H. StuckenschmidtandM. Klein, "Structure-based partitioning of large concept hierarchies," in The Semantic Web–ISWC 2004: Springer, 2004, pp. 289-303.

[19] J. Bao, D. Caragea, and V. G. Honavar, "Modular ontologies–a formal investigation of semantics and expressivity," in The semantic web–ASWC 2006: Springer, 2006, pp. 616-631.

[20] B. Cuenca Grau, "Automatic Partitioning of OWL Ontologies Using E− connections," 2005.

[21] J. Pathak, T. M. Johnson, and C. G. Chute, "Survey of modular ontology techniques and their applications in the biomedical domain," Integrated computer-aided engineering, vol. 16, 2009, pp. 225-242.

[22] I. Palmisano, V. Tamma, T. Payne, and P. Doran, "Task Oriented Evaluation of Module Extraction Techniques," in The Semantic Web - ISWC 2009. vol. 5823, A. Bernstein, *et al.*, Eds.: Springer Berlin Heidelberg, 2009, pp. 130-145.

[23] J. SeidenbergandA. Rector, "Web ontology segmentation: analysis, classification and use," in Proceedings of the 15th international conference on World Wide Web, 2006, pp. 13-22.

[24] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler, "Extracting modules from ontologies: A logic-based approach," in Modular Ontologies: Springer, 2009, pp. 159-186.

[25] M. Booshehri, K. Zamanifar, and S. Shariatmadari, "A new approach for storing RDF triples based on ontology modularization," in The 2011 International Conference on Semantic Web and Web Services, Las Vegas, Nevada, 2011, pp. 119-125.

[26] J. F. Sequeda, M. Arenas, and D. P. Miranker, "OBDA: Query Rewriting or Materialization? In Practice, Both!," in The Semantic Web–ISWC 2014: Springer, 2014, pp. 535-551.

[27] M. Rodríguez-Muro, R. Kontchakov, and M. Zakharyaschev, "Ontology-based data access: Ontop of databases," in The Semantic Web–ISWC 2013: Springer, 2013, pp. 558-573.

[28] [retrieved:June, 2015]. http://hadapt.com/

[29] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database system concepts, Sixth ed.: McGraw-Hill, 2010.

[30] M. T. Özsuand, P. Valduriez, Principles of distributed database systems: Springer Science & Business Media, 2011.

[31] [retrieved:June, 2015]. https://sweet.jpl.nasa.gov/