RDF Data Management is an Analytical Market, not a Transaction One

Olivier Curé[1(✉)], Christophe Callé[1,2], and Philippe Calvez[2]

[1] LIGM Univ. Paris Est Marne la Vallée, CNRS, 77454 Marne la Vallée, France
{olivier.cure,christophe.calle}@univ-eiffel.fr
[2] ENGIE LAB CRIGEN, Saint-Denis, France
philippe.calvez1@engie.com

Abstract. In recent years, the Resource Description Framework data model has seen an increasing adoption in Web applications and IT in general. This has contributed to the establishment of standards such as the SPARQL query language and the emergence of production-ready database management systems based on this data model. In this paper, we however argue that by concentrating on transaction related functionalities rather than analytical operations, most of these systems address the wrong data market. We motivate this claim by presenting several concrete arguments.

1 Introduction

The Resource Description Framework (RDF) data model has attracted lot of attention during these last few years. It enabled the design and implementation of many Web and IT applications. For instance, it supports innovative approaches for artificial intelligence's knowledge representation and web search. This is due to the publication of some of the largest and most popular knowledge graphs (KG), *e.g.*, DBpedia, Wikidata, Bio2RDF, UniProt, which are represented using this data model. As such, RDF is now recognized has one of the leading data model in the graph database management ecosystem. Compared to its direct competitor, the Labelled Property Graph (LPG) data model, RDF presents great support for data integration and reasoning services. These two features are due (i) to the omnipresence of Internationalized Resource Identifiers (IRI) in RDF graphs which serve as a common identifying solution at the scale of the Web and (ii) to associations with semantically-rich vocabularies, generally denoted ontologies, which together with reasoners enable to compute inferences.

This increase of interest for RDF has led to the emergence of efficient, production-ready RDF stores (see Table 1's first column for a list of the most prominent systems). In addition to natively providing reasoning services in different ontology languages, these systems possess some important functionalities that one can expect from a standard relational database management system (DBMS), *e.g.*, optimized query processing for a declarative language (typically SPARQL). These functionalities also include on line transaction process-

M. Golfarelli et al. (Eds.): DaWaK 2021, LNCS 12925, pp. 109–115, 2021.
https://doi.org/10.1007/978-3-030-86534-4_9

ing (OLTP) through the support of ACID properties. Hence, they have been designed to process high rates of (update) SPARQL queries per second.

In this paper, we argue that the support for ACID transactions is not frequently used in RDF stores. This is mainly due to the fact it is not being considered as a critical feature for most RDF-based application development. We motivate this claim in Sect. 2.1 from technical and contextual aspects as well as interviews conducted with project leaders. In Sect. 2.2, we highlight that analytical operations, not generally present in RDF stores, are highly expected in many applications. In Sect. 3, we present two general forms of analytics and emphasize that one is more needed than the other. Finally, we conclude the paper and present some perspectives.

Table 1. RDF stores characteristics, considering reasoning, RDFS:r, RDFS+: r+, OWL Lite: l n OWL QL: q, OWLRL: d, OWL Horst: h, OWLDL: dl)

Name	Transaction (ACID)	Reasoning	Analytic features	Materialized views
AllegroGraph[a]	Yes	r+, d	No	No
AnzoGraph DB[b]	Yes	r+	Graph analytics + OLAP style	Yes
Blazegraph[c]	Yes	r+, l	No	No
GraphDB[d]	Yes	r, q, d, h	No	No
MarkLogic[e]	Yes	r, r+, h	No	No
Oracle[f]	Yes	r, q, d	No	No
RDFox[g]	Yes	r+, d	No	No
Stardog[h]	Yes	All	No	No
Virtuoso[i]	Yes	r+	No	No

[a] https://franz.com/agraph/allegrograph/.
[b] https://www.cambridgesemantics.com/anzograph/.
[c] https://blazegraph.com/.
[d] https://www.ontotext.com/products/graphdb/.
[e] https://www.marklogic.com/.
[f] https://www.oracle.com/.
[g] https://www.oxfordsemantic.tech/product.
[h] https://www.stardog.com/.
[i] https://virtuoso.openlinksw.com/.

2 Arguments for OLAP RDF Stores

2.1 Why RDF Stores Do Need ACID Transactions?

In this section, we motivate the fact that many RDF-based applications do not require full ACID transaction guarantees.

SPARQL and Update Operations. Since 2008 and its first W3C recommendation release, SPARQL is the established query language for RDF data. In 2013, a set of new features to the query language were added, leading to the SPARQL 1.1 recommendation. Among these features, update operations were introduced. This means that for over five years, end-users had to either programmatically

update RDF data sets or use a non-standard, *i.e.*, system specific, declarative update solution. In a data management market anchored in transaction processing is seems unreal to leave application developers in such a situation for over five years. Hence, one can doubt that OLTP is really the market for RDF stores. One obvious question is: are the workload generally dealt with in RDF stores transactional in nature?

Linked Data Update Frequency. The popularity of RDF is partly due to its ability to integrate data and thus to break data silos. This is mainly made possible by linking these silos based on relating nodes of different graphs. The Linked Data movement and the creation of very large KGs are the most concrete examples of such an approach. The Linked Open Data cloud[1] currently connects over 1.200 graphs with an estimation of tens of billions triples over domains as diverse as life sciences, media, social networking, government, etc. Some of these KGs are produced out of extractions from open data repositories and Web scraping from Web sites. Although these sources, *e.g.*, Wikipedia, are updated at a per second rate, popular KGs are generally updated in term of days to weeks, *e.g.*, a release frequency for DBpedia, UniProt and Wikidata occurs respectively every 6 months, 4 weeks and every couple of days. So, we are far from the kind of transaction rates that ACID-compliant DBMS can support, *e.g.*, in the range of several thousands to millions of transactions per second. In fact, we are closer to the bulk loading approach of data warehouses where the main purpose of the system is to analyze data sets rather than manage the freshest data. We can then ask ourselves what is inherently complex about executing updates on an RDF graph?

Reasoning Issues. Together with data integration, the main benefit of using the RDF data model for a graph database is to benefit from reasoning services. These services depend on ontologies that can be defined with more or less expressive ontology languages (from RDFS to OWLDL to at least stay within decidable fragments). As displayed in the 'reasoning' column of Table 1, most production-ready RDF stores address rather low ontology expressiveness. This is mainly due to the practical cost of computing inferences which is already high for the least expressive languages, *e.g.*, RDFS entailment is a NP-complete problem [6]. In RDF stores, reasoning is commonly addressed via either a materialization or a query rewriting approach. In the former, whenever some updates are submitted to the DBMS, a reasoner computes the corresponding inferences and updates the stored graph accordingly. We can easily understand that this approach, although efficient considering query processing, has its limitations when the rate of updates is high. In fact, for any updates, the system needs to check whether some inferences can be deduced. So reasoning is invoked for each update operation. Incremental reasoning as proposed in RDFox [4] potentially improves the cost of materialization but it nevertheless does not enable high rate

[1] https://lod-cloud.net/.

of update transactions. In the latter approach, *i.e.*, query rewriting, reasoning is performed at query run time. Hence, handling updates is much more efficient than in the materialization approach but query processing is much less efficient. For this reason, more systems are adopting materialization than query rewriting, but some systems adopt both (*e.g.*, Allegrograph). We can thus understand that computing inferences at the rate of tens to hundreds of transactions per second is not realistic.

Real-World Use Cases. During the last few years, we have conducted several interviews and discussions with large organizations that are intensively managing RDF databases of up to several TB, *e.g.*, Publication Office of the European Union, French ministry of culture, institutes dealing with sensus in France and Italy. We found out that these companies are not updating their databases through high transaction rates but are rather bulk loading data in manner reminiscent to data warehouses, *i.e.*, at a per day or per week frequency rate. The main reasons for this update pattern is mainly due to the cost of live reasoning over incoming triples. Moreover, the companies that were not using any form of reasoning on RDF data, while their ontologies would permit to, were not using high transaction rates.

2.2 RDF Stores Should Support Analytical Operations

We now provide some arguments toward enriching RDF stores with analytical operations. These arguments come from the companies and end-users of RDF stores.

Use Cases Emphasized by RDF Sellers. We have collected the list of the most commonly encountered use cases of Table 1's RDF stores. Since, these systems have all commercial editions (Blazegraph is now AWS Neptune), we can believe that the use cases correspond to the needs of theirs customers. Among the top nine entries, we are finding the following: Advanced search and discovery, analytics/Business Intelligence, fraud detection and recommendations. These tasks are clearly relevant in analytical oriented DBMS. It is interesting to note that none of the other common use cases are really requiring ACID transaction guarantees but are rather related to data integration or smart metadata management.

End-User Point of Views. VLDB 2017's best paper [5] provides a nice survey on the usage of graph processing and graph data management system (for both LPG and RDF data models). It highlights that analytics is the task where end-users are spending the most hours (more than testing, debugging, maintenance, ETL and cleaning). Moreover, the top graph computations performed on these systems, DBMS including, are finding connected components, neighborhood queries, finding shortest paths, subgraph matching (*i.e.*, SPARQL), ranking

and centrality scores, reachability queries. These operations are quite useful in queries performing some kind of recommendations, *e.g.*, in medicine to identify a certain molecule or in culture to find a popular artist or art form.

3 The Road to Analytics in RDF Stores

3.1 Kind of Analytical Operations

Two forms of analytical operations can be considered for RDF stores. In the former, the idea is to adapt the multidimensional aspects that we can find in relational OLAP systems (ROLAP). They enable the management of information cubes and are generally extending the SQL query language with new operations such as roll-up, drill-down, pivot, slice and dice. In order to facilitate this management, they rely on a specific kinds of schemata that organize database tables in a certain manner. These schemata correspond to so-called star, snowflake or constellation. The adaptation of this approach to the RDF data model is not straightforward due to the schemaless nature of RDF, and graph models in general (*e.g.*, LPG). Moreover, in these schemata, the fact table(s) is (are) supposed to store data originating from OLTP systems. Although these data can be bulk loaded in the RDF stores, we consider that it would be better to leave them in the relational DBMS and organize some linkage solution with the RDF stores in the style of virtual KG [7]. The low frequency of update operations on ROLAP systems favors the use of materialized views (as opposed to virtual views of OLTP). Such views can be useful in RDF stores where it can impact query processing. They can be created via the SPARQL CONSTRUCT query form and hence enable RDF compositional queries. The work presented in [1] is an attempt to integrate these cube operations in SPARQL and RDF data management. The same researchers have implemented the SPADE [2] system on top of this approach but the code is not available and the approach does not seem to have been adopted in existing systems.

The second form of analytical operations correspond to graph algorithms. Until recently, they were generally present in large graph processing systems, *i.e.*, GraphX, Giraph, Gelly, but definitely make sense in a graph oriented DBMS where the data management is given much more attention. These graphs algorithms correspond to finding connected components and shortest paths, getting centrality and ranking scores, counting and enumerating connected components. Again, these algorithms are quite relevant for RDF stores since they meet the high expectations of typical end-users.

3.2 Emerging Systems

One DBMS adopting the RDF data model has in fact started to consider the analytical direction proposed in this paper. It corresponds to AnzoGraph DB which happens to be the most recent store among our list of production-ready system (only RDFox is more recent). AnzoGraph DB proposes a complete support of

SPARQL 1.1 with RDFS+ and OWLRL inferencing (via triple materialization), OLAP style analytics with windowed aggregates, cube, rollup, grouping sets and large set of functions. Moreover, it supports graph algorithms such as page rank, betweenness centrality, connected components, triangle enumeration, shortest and all paths. To the best of our knowledge, AnzoGraph DB is the only DBMS to support materialized views.

Among other production-ready RDF stores, we see the first movement toward analytics. For instance, Stardog in its 7.5 release (january 2021) provides a beta graph analytics component which contains operations such as page rank, label propagation, (strongly) connected components. Finally, outside of pure RDF store players, a similar trend is catching up with SANSA [3]. It is defined by its creators as a "big data engine for scalable processing of large-scale RDF data" and takes the form of a set of libraries. It is able to perform reasoning, querying and analytic operations. Considering analytics, it relies on either Apache Spark[2] or Apache Flink[3] distributed computing frameworks. Hence, it benefits from their respective GraphX and Gelly graph components to compute graph algorithms. Nevertheless, this can only be specified by mixing some SPARQL queries with some programs compiling over a Java environment.

4 Conclusion

In this vision paper, we have motivated the fact that the playground of RDF stores consists more of analytical than transactional processing. The limitation toward processing high rates of transaction mainly lies in the cost of inferences, the schemaless characteristic of RDF data considering multidimensional-based analytics. We believe that analytical operations based on standard graph processing are the most relevant for end-users. Potentially, some very interesting features should emerge for RDF analytical DBMS, *e.g.*, mixing reasoning with analytical operations and analytic-based data integration. Obtaining such services will come at the price of providing a seamless collaboration between relational DBMS for managing transactional data and RDF stores for computing graph analytics.

Finally, a similar trend is occurring in systems based on the LPG data model. Among the plethora of production-ready systems, *e.g.*, Neo4J, JanusGraph, RedisGraph, currently only TigerGraph seems to consider graph analytics as a promising market.

[2] https://spark.apache.org/.
[3] https://flink.apache.org/.

References

1. Colazzo, D., Goasdoué, F., Manolescu, I., Roatis, A.: RDF analytics: lenses over semantic graphs. In: Chung, C., Broder, A.Z., Shim, K., Suel, T. (eds.) 23rd International World Wide Web Conference, WWW 2014, Seoul, Republic of Korea, 7–11 April, 2014, pp. 467–478. ACM (2014)
2. Diao, Y., Guzewicz, P., Manolescu, I., Mazuran, M.: Spade: a modular framework for analytical exploration of RDF graphs. Proc. VLDB Endow. **12**(12), 1926–1929 (2019)
3. Lehmann, J., et al.: Distributed semantic analytics using the SANSA stack. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 147–155. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_15
4. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: RDFox: a highly-scalable RDF store. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 3–20. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25010-6_1
5. Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J., Özsu, M.T.: The ubiquity of large graphs and surprising challenges of graph processing. Proc. VLDB Endow. **11**(4), 420–431 (2017)
6. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the owl vocabulary. Web Semant. **3**(2–3), 79–115 (2005)
7. Xiao, G., Ding, L., Cogrel, B., Calvanese, D.: Virtual knowledge graphs: an overview of systems and use cases. Data Intell. **1**(3), 201–223 (2019)