Towards an RDF Analytics Language: Learning from Successful Experiences

Fadi Maali and Stefan Decker

Digital Enterprise Research Institute, NUI Galway, Ireland
{fadi.maali,stefan.decker}@deri.org

Abstract. SPARQL, the W3C standard querying language for RDF, provides rich capabilities for slicing and dicing RDF data. The latest version, SPARQL 1.1, added support for aggregation, nested and distributed queries among others. Nevertheless, the purely declarative nature of SPARQL and the lack of support for common programming patterns, such as recursion and iteration, make it challenging to perform complex data processing and analysis in SPARQL. In the database community, similar limitations of SQL resulted in a surge of proposals of analytics languages and frameworks. These languages are carefully designed to run on top of distributed computation platforms. In this paper, we review these efforts of the database community, identify a number of common themes they bear and discuss their applicability in the Semantic Web and Linked Data realm. In particular, design decisions related to the data model, schema restrictions, data transformation and the programming paradigm are examined and a number of related challenges for defining an RDF analytics language are outlined.

1 Introduction

The cost of acquiring and storing data has dropped dramatically in the last few years. Consequently, petabytes and terabytes datasets are becoming commonplace, especially in industries such as telecom, health care, retail, pharmaceutical and financial services. This collected data is playing a crucial role in societies, governments and enterprises. For instance, data science is increasingly utilised in supporting data-driven decisions and in delivering data products [16, 20]. Furthermore, scientific fields such as bioinformatics, astronomy and oceanography are going through a shift from "querying the world" to "querying the data" in what commonly referred to as e-science [12]. The main challenge nowadays is analysing the data and extracting useful insights from it.

In order to process the available massive amount of data, a number of frameworks were built on top of distributed cluster of commodity machines. In 2004, Google introduced the MapReduce framework [9] and its open source implementation, Hadoop[1], came out in 2007. Microsoft also introduced Dryad [14], its own distributed computation engine. Furthermore, there has also been a surge of activity on layering distributed and declarative programming languages on top of these platforms. Examples include PIG Latin from Yahoo [19], DryadLINQ from Microsoft [29], Jaql from IBM [2] and Meteor/Sopremo [11].

While analytics languages aim to utilise the high scalability of distributed platforms, they also aim to increase developer productivity by facilitating flexible adhoc data analysis and exploration. It is increasingly recognised that SQL, the main database query language, has a number of limitations that restrict its utility in analytics and complex data processing scenarios [9, 19, 29, 24]. SQL limitations include: (i) a very restrictive type system (ii) common programming patterns such as iteration and recursion can not be expressed directly in SQL (iii) processing data requires importing and formatting the data into a normalised relational format (iv) programmers often find it unnatural to analyse data by writing pure declarative queries in SQL instead of writing imperative scripts.

A close parallel can be drawn in the Semantic Web and Linked Data realm. The size of available RDF data is increasing and massive datasets are becoming commonplace. The 1.2 billion triple of Freebase can be freely downloaded[2] and the LOD Cloud grew to 31 billion RDF triples as of September 2011 [3]. Furthermore, distributed execution platforms are being utilised to process RDF data particularly for building query engines that support (part of) SPARQL [18, 8, 22] and for reasoning [28, 27, 15]. However, there has not been much activity in introducing high-level languages to support RDF analytics and processing. While general-purpose languages, such as PIG Latin and HiveQL, can be used; they are not tailored to address the peculiarities of the RDF data model and do not utilise its strength

[1] `http://hadoop.apache.org/core/`
[2] `https://developers.google.com/freebase/data`
[3] `http://lod-cloud.net/state/`

points. We contend that SPARQL alone is also not sufficient as it suffers from the same restrictions that SQL has.

In this paper, we present lessons that can be learned from existing efforts towards building an analytics language on top of RDF. We review a number of high-level analytics languages proposed in the big data and database communities. We identify five common themes they bear. For each of these themes, we present our observation, discuss corresponding efforts in the Semantic Web community and present pertinent challenges (section 2). We also discuss some further characteristics of RDF and Linked Data that can prove useful for analytics language (section 3).

2 Common Themes of High-level Languages

2.1 Data model[4]

Observation: Adoption of "relaxed" versions of the relational data model.

A large number of data models that relax the constraints of the relational data model has been proposed and adopted recently, particularly in the context of big data. MapReduce [9] uses a simple key-value pair data model. PIG [19] and HiveQL [25] support tuples but allow richer data types such as arrays, lists and maps. Jaql [2] uses a nested data model very similar to JSON. Cattell presented a survey of data models used in SQL and NoSQL data stores [6].

RDF is the data model underlying Semantic Web data. RDF is a graph-based data model that consists of a set of of triples. There has been a number of proposals for a more abstract view of RDF data. Ding et al. proposed the notion of RDF molecule [10] to decompose RDF graphs into components more coarse granular than triples. Carroll et al. introduced Named graphs [5] as a way to group and describe a set of RDF triples. Ravindra et al. introduced Nested Triple Group to refer to a set of RDF triples sharing the same subject or object [21].

[4] A data model consists of a notation to describe data and a set of operations used to manipulate that data [26]. We address the operators separately in the next subsection.

We argue that an RDF analytics language requires defining a data model that abstracts the data at a higher level than individual triples. This introduces a number of challenges and design choices of whether to support a notion of records or tuples, whether to support nesting data structures and whether to support collection types such as sets and arrays. Nested data structures simplify data manipulation and processing. Additionally, a collection of nested data is easier to be partitioned and processed in parallel. On the other hand, adopting a nested data structure on top of RDF requires enforcing the RDF graph into a set of trees. This approach was adopted by Tabulator for intuitive presentation of RDF graphs [1] and by RAPID+ [21] to enhance the performance of processing RDF on top of Hadoop. JSON-LD[5] also encodes RDF into tree but additionally extends the semantic of JSON by allowing referencing other objects in a manner similar to the RDF/XML serialisation use of `rdf:resource` (i.e. nesting with references). It remains to be seen which option of nesting, nesting with referencing or pure referencing would prove best in the context of a data model for an analytics language.

Challenge: Define a data model on top of RDF that simplifies manipulating data and works at a higher level than individual triples.

2.2 Data processing operators

Observation: Supporting only a subset of relational algebra, focusing on operators that can be easily executed in a distributed architecture.

There has been a number of proposals to support SQL on top of MapReduce framework. However, many of those proposals chose not to support the full relational algebra underlying SQL. HiveQL for instance supports only equality predicates in a join. Similar restrictions are included in SCOPE [7] and PIG Latin.

Challenge: define a subset of SPARQL algebra that is sufficient to address most common needs and is amenable to distributed execution.

2.3 Programming paradigm

Observation: A shift from pure declarative languages towards hybrid ones.

[5] `http://www.w3.org/TR/json-ld/`

Declarative languages are abstract, concise, easier for domain experts and provide opportunities for optimisation. Nevertheless, they are not always the preferred way by programmers. It is often hard to fit complex needs in a single query. Imperative scripts also allow programmers to pin down an execution plan to exploit optimisation opportunities that automatic optimisers might miss.

Increasingly, declarative languages are enriched with features from other paradigms such as imperative, functional and logic-based. PIG Latin adopts a hybrid imperative-declarative programming paradigm. Jaql and Cascalog[6] adopt features from functional programming.

On the other hand, most of the languages utilised in the Semantic Web and Linked Data realms are pure declarative languages. Examples include R2RML[7] for mapping relational databases to RDF, SPIN[8] to define rules in SPARQL and the languages defined as part of the Linked Data Integration Platform (R2R for schema mapping [3], Silk LSL for data interlinking [4] and Sieve configuration for data fusion and quality definition [17]). Another strand of related work is embedding RDF manipulation in common object-oriented languages such as ActiveRDF[9] for Ruby and SuRF[10] for Python. These approaches inherit the expressivity of the general-purpose programming language they are embedded in, but they still handle RDF data in a detailed low-level manner.

Challenge: Adopt a hybrid declarative-imperative programming paradigm for RDF processing.

2.4 Schema

Observation: From rigid schemas to partial or no schemas.

Relational databases require the schema to be designed before any data can be added to the database (a.k.a. schema first). It is generally not easy to change the schema and data that does not strictly adhere to the schema cannot be added to the database. There is a

[6] `https://github.com/nathanmarz/cascalog`
[7] `http://www.w3.org/TR/r2rml/`
[8] `http://www.w3.org/Submission/spin-overview/`
[9] `http://activerdf.org/`
[10] `https://code.google.com/p/surfrdf/`

number of advantages to schema specification, including data validation, transactional consistency guarantees, static type checking and optimisation. Nevertheless, requiring a predefined rigid schema can be an overkill particularly for ill-defined ad-hoc analytics tasks. In this context, users want to start working with the data right away in an exploratory read-only manner. Consequently, schema-on-read is increasingly adopted. PIG and Jaql support partial schema definition and allow schema definition to evolve as users are interacting with the data.

RDF data is self-describing in the sense that (a significant part of) the schema is explicitly encoded in the data and can be extracted. However, an essential task in consuming RDF data from different sources is schema mapping [23, 3]. Schema mapping exposes a homogeneous model to facilitate efficient consumption and analysis of the data. The current practice of schema mapping is similar to that of schema-first approach of relational databases (i.e. full schema mapping needs to be defined, executed before data consumption might start). We argue for supporting partial and evolving schema mapping while interacting with RDF data.

Challenge: support partial and evolving schema mapping while interacting with RDF data.

2.5 In-situ data processing

Observation: in-place processing has become an important tool for dealing with data.

The increasing volume of data generated by applications has added constraints on how easily and efficiently it can be processed. Requiring data to be moved before it can be processed, especially with read-only analytics tasks, is not a viable mechanism at extreme scale. Therefore, processing data in-place is more and more supported. In-situ data processing is also reflected by processing data coming from different locations and in different formats. It is common for analytics language to support plain text, HDFS files, JSON and databases. Most RDF tools require full transformation and materialisation of data into RDF before it can be processed (with R2RML being a notable positive exception).

Challenge: Support in-situ RDF data processing and in-place transformation of non-RDF data.

3 Further characteristics of RDF and Linked Data

Linked Data has an additional number of characteristics that should be utilised in the design of an analytics language. In the following, we go through some of these characteristics.

HTTP accessibility deploying RDF data as Linked Data makes it available via the Web and interlinked to related information. Support for retrieving data over the Web and following links should be employed by RDF processing languages.

Graph-based nature this introduces opportunity to support graph traversal and graph algorithms on top of the RDF data. SPARQL 1.1 property path provides first support in this regards. However, other languages such as Gremlin[11] and Green-Marl [13] provide richer graph traversal capabilities and support for breadth-first and depth-first traversal. These features can be embedded in an RDF analytics language.

Inference RDF has a formally defined semantics that can be used for inferencing. Inferencing allows enriching the data and makes implicit relations and facts explicit. An RDF processing language should include some support for basic inference tasks. However, a trade-off between inference capabilities and performance is inevitable.

4 Conclusion and Future Work

The Linked Data community has been very successful in publishing large amounts of useful data as evidenced by the growth of the LOD Cloud. Further emphasis is being put on building applications that utilise this data. The interlinked nature of RDF data along with its clearly defined semantics form a great basis to enable rich analysis and distilling valuable insights from this data.

[11] `https://github.com/tinkerpop/gremlin/wiki`

In this paper we focused on a number of lessons that can be learned from existing efforts on designing analytics language and on identifying some of the challenges ahead. Our current work focuses on defining use cases where SPARQL and other existing approaches for processing RDF data fall short. These use cases, along with the design clues outlined in this paper, will be used to inform the design, the implementation and the evaluation of an RDF analytics language.

Acknowledgements. Fadi Maali is funded by the Irish Research Council, Embark Postgraduate Scholarship Scheme. The ideas in this paper benefited from valuable discussions with Aidan Hogan and Marcel Karnstedt and from the material of the "Introduction to Data Science" course on Coursera by Bill Howe.

References

1. T. Berners-lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and Analyzing Linked Data on the Semantic Web. In *In Procedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06)*, 2006.
2. K. S. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Y. Eltabakh, C.-C. Kanne, F. zcan, and E. J. Shekita. Jaql: A Scripting Language for Large Scale Semistructured Data Analysis. *PVLDB*, 4(12), 2011.
3. C. Bizer and A. Schultz. The R2R Framework: Publishing and Discovering Mappings on the Web. In O. Hartig, A. Harth, and J. Sequeda, editors, *COLD*, volume 665 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
4. C. Bizer, J. Volz, G. Kobilarov, and M. Gaedke. Silk - A Link Discovery Framework for the Web of Data. In *18th International WWW Conference*, April 2009.
5. J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named Graphs. *Journal of Web Semantics*, 3(3), 2005.
6. R. Cattell. Scalable SQL and NoSQL Data Stores. *SIGMOD Rec.*, 39(4), May 2011.
7. R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. *Proc. VLDB Endow.*, 1(2), Aug. 2008.
8. H. Choi, J. Son, Y. Cho, M. K. Sung, and Y. D. Chung. SPIDER: A System for Scalable, Parallel / Distributed Evaluation of Large-scale RDF Data. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09. ACM, 2009.
9. J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1), Jan. 2008.

10. L. Ding, T. Finin, Y. Peng, P. P. da Silva, and D. L. McGuinness. Tracking RDF Graph Provenance using RDF Molecules. In *Proceedings of the 4th International Semantic Web Conference*, November 2005.
11. A. Heise, A. Rheinländer, M. Leich, U. Leser, and F. Naumann. Meteor/Sopremo: An Extensible Query Language and Operator Model. In *BigData Workshop*, 2012.
12. T. Hey, S. Tansley, and K. Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.
13. S. Hong, H. Chafi, E. Sedlar, and K. Olukotun. Green-Marl: a DSL for Easy and Efficient Graph Analysis. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII. ACM, 2012.
14. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07. ACM, 2007.
15. C. Liu, G. Qi, and Y. Yu. Large Scale Temporal RDFS Reasoning Using MapReduce. In *AAAI*, 2012.
16. M. Loukides. What is Data Science? *O'Reilly radar*, 6 2010.
17. P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: Linked Data Quality Assessment and Fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12. ACM, 2012.
18. J. Myung, J. Yeon, and S.-g. Lee. SPARQL Basic Graph Pattern Processing with Iterative MapReduce. In *Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*, MDAC '10. ACM, 2010.
19. C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: a Not-so-foreign Language for Data Processing. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08. ACM, 2008.
20. F. Provost and T. Fawcett. Data Science and its Relationship to Big Data and Data-Driven Decision Making. *Big Data*, 1(1), Mar. 2013.
21. P. Ravindra, H. Kim, and K. Anyanwu. An Intermediate Algebra for Optimizing RDF Graph Pattern Matching on MapReduce. In *Proceedings of the 8th extended semantic web conference on The semanic web: research and applications - Volume Part II*, ESWC'11. Springer-Verlag, 2011.
22. A. Schätzle, M. Przyjaciel-Zablocki, and G. Lausen. PigSPARQL: Mapping SPARQL to Pig Latin. In *Proceedings of the International Workshop on Semantic Web Information Management*, SWIM '11. ACM, 2011.
23. P. Shvaiko and J. Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1), 2013.
24. M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The End of an Architectural Era: (It's Time for a Complete Rewrite). In *Proceedings of the 33rd international conference on Very large data bases*, 2007.
25. A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Z. 0002, S. Anthony, H. Liu, and R. Murthy. Hive - a Petabyte Scale Data Warehouse Using Hadoop. In *ICDE*. IEEE, 2010.
26. J. Ullman. *Principles of Database and Knowledge-base Systems*, chapter 2. Computer Science Press, Rockville, 1988.
27. J. Urbani, S. Kotoulas, J. Maassen, F. van Harmelen, and H. Bal. OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples. In *Proceedings of the 7th international conference on The Semantic Web: research and Applications - Volume Part I*, ESWC'10. Springer-Verlag, 2010.

28. J. Urbani, S. Kotoulas, E. Oren, and F. Harmelen. Scalable Distributed Reasoning Using MapReduce. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09. Springer-Verlag, 2009.
29. Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey. DryadLINQ: a System for General-purpose Distributed Data-parallel Computing Using a High-level Language. In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*. USENIX Association, 2008.