On Reflection in Linked Data Management

George H. L. Fletcher

Eindhoven University of Technology
The Netherlands
g.h.l.fletcher@tue.nl

***Abstract*—The Linked Data (LD) initiative promotes the use of international standards for data sharing on the web. The flexible Resource Description Framework (RDF) graph data model is a central LD technology. Much of RDF's flexibility is due to its blurring of the traditional distinction between data and metadata.**

A particularly powerful form of metadata is query expressions. A query language is called *reflective* if it can interpret expressions of the language stored as data both as "active" data and as regular "static" data. Fundamental applications of reflection are found in diverse domains such as security, information integration, and data quality. Consequently, a variety of RDF representations of active data have been proposed. However, RDF query languages, while bridging the data-metadata barrier, continue to maintain a divide between active and static data. Hence, applications of RDF active data rely on ad-hoc special-purpose solutions, thereby limiting their broader use and impact.

This paper argues that active data must be promoted as first class citizens in RDF querying, if we are to realize the full potential of LD. The aim is to stimulate the study of general frameworks and solutions for reasoning about and applying reflection in the web of data, towards addressing this fundamental research challenge.

I. Motivation and Vision

Data management relies heavily on the effective use of metadata such as database schemas and statistics. A particularly powerful type of metadata is query and rule expressions. Such data is "active" in the sense that it can be interpreted dynamically on the current database instance.

Active data (i.e., query expressions stored as data) play an important role in a wide range of basic applications, for example in the specification and implementation of

- dynamic data integration rules [1]–[6];
- security and access control policies [6], [7];
- web reasoning and information dissemination policies [8]–[12];
- data quality and trust policies [6], [13]–[16]; and,
- web data provenance reasoning [6], [14].

In these applications, active data is preferable to static metadata because it provides for up-to-date "live" results, thereby eliminating possible redundancies and the anomalies of maintaining static data. In general, active data provides for the declarative formulation of sophisticated data management policies, which themselves, as data, can be pushed and pulled as the environment evolves.

A query language is called *reflective* if it can alternate between interpreting active data statically, as regular data, and actively, as expressions in the language itself. Active interpretation is typically accomplished with an "eval" operator, akin to the eval operator found in functional languages such as Lisp and Python. In a reflective language, active data can be dynamically selected and evaluated at query-time, as part of query evaluation. Hence, reflective querying is a powerful approach to the management of data involving active data, such as in the variety of applications indicated above. Indeed, reflective querying on structured data is an area of successful study (e.g., [6], [11], [12], [17], [18]).

As data has evolved from centralized structured databases to loosely structured data distributed on the web, the distinction between data and metadata has become increasingly blurry. The RDF data model, a W3C standard, was designed in part to support this blurring [19]. RDF data sets are not required to abide by a pre-defined schematic structure. This flexibility of RDF makes it particularly well suited to dynamic data creation and sharing on the web, where, indeed, it is the standard data model under the W3C's Linked Data (LD) initiative to promote the use of standards in web data management [20], [21]. RDF and the LD standards have experienced rapid broad adoption in the public (e.g., open government data across the globe), commercial (e.g., Google, BBC, Yahoo), and science sectors (e.g., Uniprot and Bio2RDF).

Active RDF data and its representations (e.g., [13], [15], [22]) are increasingly finding fundamental applications in many of the areas listed above (e.g., [1], [2], [5], [14]–[16], [23]). Current RDF query languages, however, continue to maintain a divide between active and static data. In particular, no reflective mechanisms have been introduced in RDF query languages.

Indeed, while there exists a rich body of work on rule-based entailment for LD which manifests itself in mature standards such as RIF, OWL, and entailment regimes in SPARQL 1.1 (cf. [22], [24]), to date this work focuses on support for static rule sets which are not "actively" available as data for manipulation, modification, and (re)application in the SPARQL language at run-time, during query evaluation.

Hence, the many applications of active RDF data rely on ad-hoc purpose-built solutions, thereby limiting their scientific and practical impact. This situation presents a critical barrier to progress in the field. To overcome this, *it is vital that active data be promoted as first class citizens in RDF querying.* Motivated by these observations, I propose the following

research challenge to the LD management community: *Is there a general theory of reflection for LD query languages?* A coherent unified understanding of effective approaches to reflective RDF querying will provide the solid foundations necessary to advance the state of the art in the many basic applications of active data in LD, towards realizing the full potential of the LD vision. Due to the novel blurring of data and metadata in RDF and its query languages, and the heterogeneous distributed nature of LD sources, the study of reflective mechanisms will require fresh insights and solutions beyond those found for reflection in traditional structured data management.

I next discuss reflective querying in more detail, and sketch two fundamental applications of reflection in LD management.

II. Reflection in RDF query languages

I assume familiarity with RDF, SPARQL, and the conjunctive "basic graph patterns" (BGPs) which form the core of SPARQL queries. Suppose as a simple running example that we have an RDF graph animalCare concerning pet care, with the predicate "eat", among others (e.g., ⟨dbpedia:Rabbits, ex:eat, dbpedia:Carrots⟩). An example BGP on our graph is allowedFoods $=$ $(?animals, \text{ex:eat}, ?food)$, $(?food, \text{rdf:type}, ?type)$, $(?type, \text{rdfs:subClassOf}, \text{ex:animal-food})$. Here, the variable $?food$ binds to all recognized animal foods which $?animals$ are known to eat. A BGP *query* is an expression of the form $Q \leftarrow P$, where Q and P are both BGPs such that all variables occurring in Q also occur in P. For example, we can define a query mayEat as $(?animals, \text{ex:may-eat}, ?food) \leftarrow$ allowedFoods.

What would reflection mean in RDF data? Suppose we also want to keep information about allowed animal foods in the database. One approach is to materialize the results of mayEat in the animalCare graph. However, as animalCare evolves (e.g., new feeding guidelines appear, or the definition of mayEat itself changes) this materialized data can easily become out of date and, hence, a source of inconsistency. Alternatively, we can store the definition of mayEat itself as a piece of (active) data, known as *reifying* the query, and then extract and compute the results of mayEat on demand during query processing using BGPs extended with *reflective evaluation* capabilities.

Two basic approaches have been proposed in the literature for extending query languages with reflection: (i) reify queries by decomposing and tagging their parse trees, explicitly storing this data in the database, and then include in the language an operator to execute such reified queries on the fly [25], [26]; or, (ii) reify queries as strings, and extend the language with operators to evaluate and, possibly, to manipulate such string representations [3], [4], [6], [26], [27]. In their most general form, both approaches are powerful, permitting the definition of recursive queries and dynamic run-time query construction and reification in the query language itself.

To my knowledge, no general study of reflection in RDF has been undertaken. Basic research challenges here include: (1) RDF reification strategies, building on work in the community on rule and query vocabularies (e.g., [15]); (2) reflective extensions to LD languages (e.g., from full reflection to more modest forms disallowing recursion); and, (3) computationally effective implementation strategies over massive RDF graphs.

III. Two example applications

A. Reflective linked data distributed query processing

Continuing our example, suppose that the animalCare graph is linked to other graphs using the LD standards. Suppose now that we have recently purchased a turtle, and would like to know what we may safely feed our new pet. So, we pose a query against animalCare in which we evaluate the reified mayEat to look for turtle foods, and, finding no local results, proceed to distribute our query. During this distribution, suppose we would only like to consider providers which are animal food authorities with good reputations (e.g., with high ratings from users). To orchestrate and execute the distribution of our query, we can transform the input query using reflective query construction to incorporate additional reified queries which provide us, for example, with dynamically constructed domain authority and quality rating information, using the up-to-date information available in animalCare.

Many fundamental aspects of dynamic on-the-fly LD query processing can likewise be effectively expressed and implemented using active data and reflective querying. For example, this methodology can be applied towards issues such as run-time caching/indexing and real-time query optimization, distribution, and source discovery strategies [20], [28], using dynamically derived data (perhaps also polling from other linked data sets) regarding source and connection quality. In general, adaptive query processing policies and strategies can be uniformly and declaratively expressed as queries themselves. Reflection would then serve as a fundamental unifying mechanism for the design and effective implementation of flexible dynamic LD query processing solutions.

To my knowledge, reflection for LD querying has never before been explored. The introduction of a framework promoting the independence of query formulation from declaratively orchestrated distributed query evaluation over LD sources would present a fresh breakthrough in our understanding of querying the web of data.

Basic research challenges here include: (1) dynamic federation [20] and orchestration of LD sources using reflective querying; (2) general optimization methodologies for reflective LD queries (e.g., deciding when and where to construct, store, and/or evaluate reified queries); and, (3) implementation strategies for reflective LD queries (e.g., reflection-aware LD source indexing, and caching mechanisms for reified queries).

B. Reflective linked data source integration

Suppose next that our search for pet food spans providers in the UK, Australia, and the USA. Fortunately, these regions share the English language. Unfortunately, in local varieties of English, subtle distinctions are made regarding the word "turtle". For example, if our pet is a fresh water turtle (in

American and Australian English), then in the UK it might be referred to as a terrapin. If our pet is a land turtle (in American English), then it might be called a tortoise in the UK and Australia. Furthermore, veterinarians and animal societies might refer to our pet as a chelonian. Clearly, these differences in terminology must be bridged during query evaluation.

Mapping between terminologies is an example of the general problem of data/information/ontology integration, a common challenge in web information systems [29]. In the context of LD, the use of SPARQL queries for expressing and processing data integration policies has been successfully studied [1], [2], [5], [13], [23].

These mappings, as active data, can be stored and managed in RDF, and dynamically selected and applied during data integration using reflective querying. In our example, this would involve, at query time, (1) locating (perhaps from other trusted LD providers) and applying appropriate animal terminology mappings in order to (2) construct an appropriately rewritten query for each data source and, finally, (3) mapping the retrieved results back into our local vocabulary. Each of these three steps is performed on the fly using reflection.

In this way, data integration processing is not restricted to some fixed static set of mapping policies, as is the case in traditional data integration, but rather reflects the current state of integration policies at the time of query evaluation. In this view, data integration is treated as a dynamic aspect of LD query compilation, optimization, and evaluation. In addition to the flexibility of declarative expression of adaptive data integration policies, this approach also promotes the full use of the query processing tool-chain in performing data integration.

To my knowledge, this reflective approach to integration of linked data is novel. The introduction of a coherent theory and set of tools supporting independence of clients of LD information systems from the internal workings of data integration processes would open a new perspective in our understanding of data integration on the web.

Basic research challenges here include: (1) the study of RDF representations for integration policies; (2) reflective methodologies for locating and resolving appropriate mappings amongst LD providers (e.g., strategies for chaining mappings); and, (3) approaches to efficient execution of mapping policies (e.g., in an approximate, pay-as-you-go fashion).

IV. Other related work

Perhaps closest to the proposed vision of reflective querying for LD management are the notions of "integration independence" [30], [31] and "intensional associations" [3], [4] which have been recently proposed and studied. The research vision is synergistic with this work. More generally, the proposed research program contributes to and builds on the study of views in web data management, e.g., [32], [33], and work on adaptive query processing and active components in web data management, e.g., [12], [34].

Acknowledgments: I thank the reviewers for their helpful critical feedback.

References

[1] G. Correndo et al, "SPARQL query rewriting for implementing data integration over linked data," in *DataSem*, Lausanne, 2010.
[2] J. Euzenat, A. Polleres, and F. Scharffe, "Processing ontology alignments with SPARQL," in *CISIS*, Barcelona, 2008, pp. 913–917.
[3] A. Presa et al, "Modeling associations through intensional attributes," in *ER*, Gramado, Brazil, 2009, pp. 315–330.
[4] M. A. V. Salles, J. Dittrich, and L. Blunschi, "Intensional associations in dataspaces," in *ICDE*, Long Beach, CA, USA, 2010, pp. 984–987.
[5] S. Schenk and S. Staab, "Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web," in *WWW*, Beijing, 2008, pp. 585–594.
[6] D. Srivastava and Y. Velegrakis, "Intensional associations between data and metadata," in *SIGMOD*, Beijing, 2007, pp. 401–412.
[7] B. Carminati et al, "Enforcing access control in web-based social networks," *ACM Trans. Inf. Syst. Secur.*, vol. 13, pp. 6:1–6:38, 2009.
[8] R. Chirkova and G. H. L. Fletcher, "Towards well-behaved schema evolution," in *WebDB*, Providence, 2009.
[9] S. Hagemann and G. Vossen, "Web page augmentation with client-side mashups as meta-querying," in *ACIIDS*, Hue City, Vietnam, 2010.
[10] G. Tummarello et al, "Sig.ma: Live views on the web of data," *J. Web Sem.*, vol. 8, no. 4, pp. 355–364, 2010.
[11] D. Gawlick et al, "Applications for expression data in relational database systems," in *ICDE*, Boston, 2004, pp. 609–620.
[12] A. Kemper et al, "Building scalable electronic market places using hyperquery-based distributed query processing," *WWW*, vol. 8, 2005.
[13] C. Bizer and A. Schultz, "The R2R framework: Publishing and discovering mappings on the web," in *COLD*, Shanghai, 2010.
[14] R. Q. Dividino et al, "Querying for provenance, trust, uncertainty and other meta knowledge in RDF," *J. Web Sem.*, vol. 7, pp. 204–219, 2009.
[15] C. Fürber and M. Hepp, "Using SPARQL and SPIN for data quality management on the semantic web," in *BIS*, Berlin, 2010, pp. 35–46.
[16] D. J. Weitzner et al, "Creating a policy-aware web: Discretionary, rules-based access for the world wide web," in *Ferrari & Thuraisingham, (eds.), Web and Information Security*. Idea Group, 2005.
[17] L. E. Olson, C. A. Gunter, W. R. Cook, and M. Winslett, "Implementing reflective access control in SQL," in *DBSec*, Montreal, 2009, pp. 17–32.
[18] J. Van den Bussche, D. Van Gucht, and S. Vansummeren, "A crash course on database queries," in *PODS*, Beijing, 2007, pp. 143–154.
[19] G. Klyne and J. J. Carroll, "Resource Description Framework (RDF): Concepts and Abstract Syntax," W3C Recommendation, 2004.
[20] O. Hartig and A. Langegger, "A database perspective on consuming linked data on the web," *Datenbank Spektrum*, vol. 10, pp. 57–66, 2010.
[21] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.
[22] M. Kifer and H. Boley, "RIF overview (second edition)," W3C Working Group Note, 2013.
[23] K. Makris et al, "Ontology mapping and SPARQL rewriting for querying federated RDF data sources," in *ODBASE*, Hersonissos, 2010.
[24] B. Glimm and C. Ogbuji, "SPARQL 1.1 entailment regimes," W3C Rec., 2013.
[25] J. Van den Bussche et al, "Reflective programming in the relational algebra," *J. Comput. Syst. Sci.*, vol. 52, no. 3, pp. 537–549, 1996.
[26] J. Van den Bussche, S. Vansummeren, and G. Vossen, "Towards practical meta-querying," *Information Systems*, vol. 30, no. 4, pp. 317–332, 2005.
[27] F. Neven et al, "Typed query languages for databases containing queries," *Information Systems*, vol. 24, no. 7, pp. 569–595, 1999.
[28] O. Hartig, "An overview on execution strategies for linked data queries," *Datenbank-Spektrum*, vol. 13, no. 2, pp. 89–99, 2013.
[29] G. H. L. Fletcher and C. Wyss, "Towards a general framework for effective solutions to the data mapping problem," *J. Data Semantics*, vol. XIV, pp. 37–73, 2009.
[30] L. M. Haas et al, "A first step towards integration independence," in *NTII*, Long Beach, CA, USA, 2010, pp. 147–150.
[31] R. J. Miller, "Information integration: a vision for integration independence and linking open data," in *AMW*, Buenos Aires, 2010.
[32] A. Magkanaraki, V. Tannen, V. Christophides, and D. Plexousakis, "Viewing the semantic web through RVL lenses," *J. Web Sem.*, vol. 1, no. 4, pp. 359–375, 2004.
[33] M. Shaw et al, "vSPARQL: A view definition language for the semantic web," *J Biomed Inform*, vol. 44, no. 1, pp. 102–117, 2011.
[34] S. Abiteboul, O. Benjelloun, and T. Milo, "The active XML project: an overview," *VLDB J.*, vol. 17, no. 5, pp. 1019–1040, 2008.