On reflection in linked data management

George Fletcher

Eindhoven University of Technology
The Netherlands

DESWeb 2014

Chicago

31 March 2014

linked data vision

Towards a web of data, use standards in web data management:
HTTP, URIs, RDF, SPARQL, ...

linked data vision

Towards a web of data, use standards in web data management:
HTTP, URIs, RDF, SPARQL, ...

thesis

Towards realizing the full potential of this vision, it is vital that active data be promoted as first class citizens in LD querying.

linked data vision

Towards a web of data, use standards in web data management: HTTP, URIs, RDF, SPARQL, ...

thesis

Towards realizing the full potential of this vision, it is vital that active data be promoted as first class citizens in LD querying.

research challenge

Is there a general theory of reflection for LD query languages?

Active data, namely, query and rule expressions stored as data, provide for the declarative formulation of sophisticated data management policies, for example

- data integration policies,
- security and access control policies,
- data quality and trust policies, and
- provenance reasoning.

Active data, namely, query and rule expressions stored as data, provide for the declarative formulation of sophisticated data management policies, for example

- data integration policies,
- security and access control policies,
- data quality and trust policies, and
- provenance reasoning.

Such data is active in the sense that it can be interpreted dynamically on the current database instance, giving live results, eliminating redundancies and anomalies of maintaining static data.

As data, they 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

- typically accomplished with an "eval" operator

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

- typically accomplished with an “eval” operator

Active data can then be dynamically selected, modified, and evaluated at query-time, as part of query evaluation

- hence, this is a powerful tool for the management of data involving active data

Reflective querying on structured data is an area of successful study

Query and rule data are increasingly finding fundamental applications in many LD areas, e.g.,

- data integration (Correndo et al. 2010, Euzenat et al. 2008, Schenk and Staab 2008, Makris et al. 2010)
- data quality, trust, and provenance (Bizer and Schultz 2010, Dividino et al. 2009, Fürber and Hepp 2010)

Query and rule data are increasingly finding fundamental applications in many LD areas, e.g.,

- data integration (Correndo et al. 2010, Euzenat et al. 2008, Schenk and Staab 2008, Makris et al. 2010)
- data quality, trust, and provenance (Bizer and Schultz 2010, Dividino et al. 2009, Fürber and Hepp 2010)

Rule-based reasoning over LD, e.g.,

- RIF, OWL, SPARQL 1.1 entailment regimes

focuses on support for static rule sets which are not actively available as data for manipulation, modification, and application at query-time

Example. Suppose we have an LD source animalCare concerning pet care, and we want to reason about "officially" allowed pet foods, with the query mayEat:

$$(?animal, \text{mayEat}, ?food) \leftarrow (?animal, \text{eat}, ?food), (?food, \text{type}, ?type), (?type, subClass, animalFood)$$

Example. Suppose we have an LD source animalCare concerning pet care, and we want to reason about "officially" allowed pet foods, with the query mayEat:

$$(?animal, \text{mayEat}, ?food) \leftarrow (?animal, \text{eat}, ?food), (?food, \text{type}, ?type), (?type, subClass, animalFood)$$

To maintain mayEat, we could materialize the results at animalCare. However, as the graph evolves, e.g., with new feeding guidelines, the results and the query become outdated.

Example. Suppose we have an LD source animalCare concerning pet care, and we want to reason about "officially" allowed pet foods, with the query mayEat:

$$(?animal, \text{mayEat}, ?food) \leftarrow (?animal, \text{eat}, ?food), (?food, \text{type}, ?type), (?type, subClass, animalFood)$$

To maintain mayEat, we could materialize the results at animalCare. However, as the graph evolves, e.g., with new feeding guidelines, the results and the query become outdated.

Alternatively, we could store mayEat as a piece of active data (i.e., "reify" the query), and select and compute it as part of (reflective) query processing.

Current LD query languages continue to maintain a divide between active and static data, with no reflective mechanisms introduced to date

Hence, the many applications of active data in LD management rely on ad-hoc purpose-built solutions, thereby limiting their scientific and practical impact

Current LD query languages continue to maintain a divide between active and static data, with no reflective mechanisms introduced to date

Hence, the many applications of active data in LD management rely on ad-hoc purpose-built solutions, thereby limiting their scientific and practical impact

Research challenges here include

- study of RDF reification strategies, building on work in the community on rule and query vocabularies
- design of reflective extensions to LD languages
- effective implementation strategies over massive graphs

two example applications

Example, cont. Suppose we buy a pet turtle, and, finding that animalCare lacks feeding information for turtles, we explore linked sources to resolve our query.

Example, cont. Suppose we buy a pet turtle, and, finding that animalCare lacks feeding information for turtles, we explore linked sources to resolve our query.

What if we had active data, providing up-to-date information on the domain authority and user-ratings of other LD sources?

A reflective language could rewrite and execute the distribution of our query to incorporate this live information.

Example, cont. Suppose we buy a pet turtle, and, finding that animalCare lacks feeding information for turtles, we explore linked sources to resolve our query.

What if we had active data, providing up-to-date information on the domain authority and user-ratings of other LD sources?

A reflective language could rewrite and execute the distribution of our query to incorporate this live information.

Other issues which could likewise be handled include

- real-time caching/indexing and query optimization, and
- source discovery and query distribution

using dynamically derived data regarding, e.g., source and connection quality.

The general research goal here is the development of a framework promoting the independence of query formulation from declaratively orchestrated distributed query evaluation over LD sources.

reflective LD distributed query processing

The general research goal here is the development of a framework promoting the independence of query formulation from declaratively orchestrated distributed query evaluation over LD sources.

Major challenges here include:

- study general strategies for dynamic federation and orchestration using reflective querying
- develop optimization methodologies for reflective LD queries
- investigate implementation solutions for reflective LD queries

Example, cont. Suppose that our search for pet food spans providers in the UK, Australia, and the USA.

Although all English speaking areas, subtle distinctions are made regarding the word “turtle”

- fresh water turtle (US, Australia) = terrapin (UK)
- land turtle (US) = tortoise (UK, Australia)
- turtle = chelonian (veterinarians and animal societies)

Example, cont. Suppose that our search for pet food spans providers in the UK, Australia, and the USA.

Although all English speaking areas, subtle distinctions are made regarding the word "turtle"

- fresh water turtle (US, Australia) = terrapin (UK)
- land turtle (US) = tortoise (UK, Australia)
- turtle = chelonian (veterinarians and animal societies)

Hence, sources must be integrated before our query can be successfully resolved.

On-the-fly, using reflection, our query could be resolved by

- locating (perhaps from a mapping authority source) and applying appropriate animal terminology mappings,
- constructing an appropriately rewritten query for each LD source, and
- finally, mapping the retrieved results back into our local vocabulary.

On-the-fly, using reflection, our query could be resolved by

- locating (perhaps from a mapping authority source) and applying appropriate animal terminology mappings,
- constructing an appropriately rewritten query for each LD source, and
- finally, mapping the retrieved results back into our local vocabulary.

In this vision, data integration is not restricted to some fixed static set of mapping policies, but rather reflects the current state of integration policies at the time of query evaluation.

The general research goal is to develop a theoretical foundation and practical toolset supporting independence of clients of LD information systems from the internal workings of data integration processes.

reflective linked data integration

The general research goal is to develop a theoretical foundation and practical toolset supporting independence of clients of LD information systems from the internal workings of data integration processes.

Major challenges here include:

- study of RDF representations of integration policies
- reflective methodologies for locating and resolving appropriate mappings amongst LD providers
- solutions for efficient execution of mapping policies

recap

thesis

Towards realizing the full potential of the linked data vision, it is vital that active data be promoted as first class citizens in linked data querying.

research challenge

Is there a general theory of reflection for linked data query languages?

Thank you!

Questions?

On reflection in linked data management

George Fletcher

Eindhoven University of Technology