A Hybrid Framework for Online Execution of Linked Data Queries

Mohamed M. Sabri
Supervised by: Prof. David Toman and Prof. Grant Weddell
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
mmsabri@uwaterloo.ca

ABSTRACT

Linked Data has been widely adopted over the last few years, with the size of the Linked Data cloud almost doubling every year. However, there is still no well-defined, efficient mechanism for querying such a Web of Data. We propose a framework that incorporates a set of optimizations to tackle various limitations in the state-of-the-art. The framework aims at combining the centralized *query optimization* capabilities of the data warehouse-based approaches with the *result freshness and explorative data source discovery* capabilities of link-traversal approaches. This is achieved by augmenting base-line link-traversal query execution with a set of optimization techniques. The proposed optimizations fall under two categories: *metadata-based* optimizations and *semantics-based* optimizations.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Miscellaneous

Keywords

Linked Data; Semantic Web; Query Execution

1. PROBLEM INTRODUCTION

The World Wide Web is transforming from a medium of interlinked documents to a medium of *interlinked knowledge*, where the sum of human knowledge will eventually be available in a machine-readable and semantic-rich form. This form of an interlinked, globally-distributed knowledge network is becoming known as Linked Data. Linked Data has been widely adopted over the last few years, with the size of the Linked Data cloud almost doubling every year[1]. This fast adoption is realizing the vision of transforming the Web into a globally connected knowledge network, which we can access as a huge, distributed database. However, one of the *key challenges* to enable this vision is *the unavailability of efficient query mechanisms.* There is still no well-defined, efficient mechanism to query such a Web of Data. Most previous research efforts tried to reduce the problem to centralized, data warehouse-based, query processing by requiring that all available Linked Data be pre-crawled and indexed in a local data store to allow user queries to be run later against this *local copy* of online Linked Data. In this research, we focus on the original problem of executing queries against the *online* Linked Data graph. This style of Linked Data query execution is sometimes referred to in the literature as link-traversal query execution. Contrary to the centralized approaches, link-traversal query execution executes the queries against the live Linked Data copy and is able to dynamically discover new data sources and always provide up-to-date answers. However, current proposals for link-traversal query execution suffers from poor query execution performance, incomplete query answers, and redundant data-retrieval. The target of this research is to design and evaluate a hybrid framework that overcomes the current limitations of link-traversal query execution, without sacrificing any of its benefits.

2. STATE OF THE ART

Most of current Linked Data query processing engines follow a *centralized* approach (usually referred to as: search engine-style, data warehouse-based, or materialization-based approaches). The process starts by crawling all available Linked Data and indexing a local copy in one of the available RDF stores. The query engine then uses this local copy to answer user queries. Examples of such systems are: FactForge [1], LOD Cache [2], Sindice[2], YARS2 [3], and SWSE [4]. Centralized approaches can provide fast query answering; however, they suffer from severe limitations: 1. Query answers do not reflect the current status of the web of Linked Data. 2. Extensive *preprocessing* is required. 3. New data sources cannot be discovered at query execution time. 4. It may not be feasible to store a local copy of all available Linked Data in the future. 5. Data providers have to give up (or delegate) access control over their data. 6. There may be legal issues that prevent the engine from storing third–party data locally.

In contrast, *online* link-traversal query execution [5, 6, 7] emerged to provide a remedy for most of the above. It relies only on the Linked Data Principles [8]. The process starts by dereferencing a set of URIs, called the *seed* URIs

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

[2]http://lod.openlinksw.com/sparql/

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2741751.

(usually comes from the query itself); it then retrieves relevant sources, and recursively follows new URIs in the retrieved data. There is no need for preprocessing and results are always up-to-date. Link-traversal query execution does not assume initial complete knowledge of a fixed set of data sources; the execution process integrates data retrieval with the discovery of new relevant sources on-the-fly. That is why it can answer queries based on data from newly-discovered data sources. Furthermore, it does not require the data sources to provide any query processing capabilities.

However, this is achieved at the expense of slower query execution and exhaustive data retrieval. The current limitations of link-traversal query execution approaches can be attributed to one or more of the following points:

1. **The lack of query execution optimization:** Since the query engine learns about data sources at runtime, it cannot perform more elaborate query optimization.
2. **Result incompleteness:** An important source for incompleteness is the lack of support for more expressive *entailment regimes* [9]. Another source of incompleteness is reachability limitations of the initial set of source (seed) URIs. The literature offers several proposals for what can be called *index-based source selection* [10, 11, 12, 13]. However, the way these indexes are structured does not allow reasoning over indexed data.
3. **Redundant data retrieval:** The query execution follows an uninformed, exhaustive, data retrieval process, due to the lack of reasoning about RDFS and OWL semantics and what this might entail about the availability , or *the unavailability*, of additional query answers.

Hybrid Query Execution: Our proposed framework falls under this category. A hybrid query execution engine will not assume full knowledge about available data sources. It can maintain local information about some of the data sources; however, it still follows an explorative, link-traversal approach to compliment this knowledge. Previous work has discussed the potential for such a strategy at a variety of levels [14, 15, 16, 17]. However, to the best of our knowledge, there is no previous work that provides a comprehensive proposal and realization for such a framework.

3. APPROACH AND METHODOLOGY

The ultimate goal of this research is to provide a design, and a reference implementation, for a hybrid framework for online execution of Linked Data queries; a framework that combines the *centralized query optimization* capabilities of the data warehouse-based approaches with the *result freshness and explorative data source discovery* capabilities of link-traversal approaches. What we envision can be described as: a query engine that remembers *how to* answer user queries, instead of *the answers* to user queries. To achieve this we will augment *base-line* link-traversal query execution with two sets of optimizations: 1. Optimizations derived from caching and utilizing statistical *metadata* about Linked Data, referred to below as ***metadata-based Query Optimizations***. 2. Optimizations derived from reasoning about Linked Data *semantics*, referred to below as ***semantics-based Query Optimizations***.

3.1 Query Syntax and Semantics

In previous formal models of link-traversal query execution, queries are evaluated against a **Web of Linked Data** W, defined as a 3-tuple $W = (D, data, adoc)$, such that: D is a (potentially infinite) set $\{d_1, d_2, ...\}$ of Linked Data documents, *data* is a total mapping that associates each document with a finite subset of RDF triples, and *adoc* is a partial, surjective, mapping between a URI and the corresponding LD document [5]. Given the reachability limitations of the Web of Linked Data, link-traversal query approaches follow reachability-based query semantics [5], where the query is evaluated against the subweb of the Web of Linked Data that is recursively *reachable* from the initial set of URIs.

Support for SPARQL *limit* clause: However, given the unbounded nature of the Web of Linked Data, exhaustively complete query answers are not always feasible, nor desirable. In many cases, the user is only interested in a small (limited) number of query results. SPARQL offers a solution modifier for these query cases: the *limit* clause. Nevertheless, to the best of our knowledge, the *limit* clause has not been studied in the context of *live*, link-traversal query execution.

Definition 3.1. Query syntax:
The syntax of an LD query Q is given by: $B \mid Q\ limit\ k$, where B is a BGP and k is a positive integer.

We note here that an *unlimited* Linked Data query (i.e., a query without a *limit* clause) is equivalent to a query with a limit clause of an infinite k. To define the query semantics of queries with *limit* clause, we need to adopt a *possible results semantics*[18], defined as follows (where $Eval(Q, W, S)$ abstracts the standard evaluation of a BGP Q over the S-reachable subweb of W).

Definition 3.2. Query semantics:
Given a query Q, a Web of Linked Data W, and a set S of URIs, we write $Den(Q, W, S)$ to denote the set of all *possible results* $\{R_i\}$ of evaluating Q over the *S-reachable* subweb of W, defined as follows:

1. Where ($Q =$ "B"): $Den(Q, W, S) = \{Eval(Q, W, S)\}$,
2. Where ($Q =$ "$Q'\ limit\ k$"):
   $Den(Q, W, S) = \{R | R \in Den(Q', W, S)\ and\ |R| < k\}$
   $\cup \{R | \exists R' \in Den(Q', W, S) : R \subseteq R', |R| = k\ and\ |R'| \geq k\}$

If a query execution engine computes any element of $Den(Q, W, S)$, we say that the engine is *correct*.

3.2 LD Oracle-based Query Execution

By *Linked Data Oracle*, LD Oracle for short, we refer to a hypothetically perfect oracle that guides the link-traversal process during query execution by continuously prioritizing and revising the list of URIs to be retrieved next *(the retrieval queue)*. Algorithm 1 provides a skeleton for evaluating a query Q, assuming an implementation of LD Oracle which can answer the calls for the *updateAndRevise()* method. The purpose of this method call is to provide a point of entry for the LD Oracle to guide the link-traversal query execution process, and to allow the LD Oracle to incrementally *update* its understanding (statistics) of the Web of Linked Data. The purpose of the subsequent stages of this research is to envision an implementation of such an LD Oracle and to investigate a set of candidate optimizations that could be employed by it.

```
input  : Q: a Linked Data query,
         W: a reference to the Web of Linked Data,
         S: a set of seed URIs,
         LDOracle: an implementation of LD Oracle.
output: Sol a set of solution mappings for evaluating
         Q over W.

D:= ∅ ; /* local storage for intermediate data */
visitedURIs, Sol:= ∅
Ret:= S ;      /* the retrieval queue; a priority
queue of URIs to be retrieved next, initialized
here to S. */
while Ret.size() > 0 do
    Ret:= LDOracle.updateAndRevise(Ret,Q,Sol) ;
    /* to get the LDOracle's revision of the
    retrieval queue. */
    uri:= poll(Ret);
    d:=data(adoc(uri)) ;      /* see section 3.1 */
    visitedURIs:= visitedURIs ∪ uri;
    newURIs:= uris(d) - (visitedURIs ∪ Ret);
    Ret:= Ret ∪ newURIs;
    foreach t ∈ d do
        D:=D ∪ t;
        Sol:= Eval(Q.B,D) ;        /* abstracts the
        incremental evaluation of the query BGP
        over the intermediate data. */
        if |Sol| ≥ Q.k then
            return Sol;
        end
    end
end
return Sol;
```
Algorithm 1: Query Execution Algorithm

4. METADATA-BASED QUERY OPTIMIZATIONS

The intuition here is that, even if the published Linked Data triples change frequently, certain characteristics (*statistics*) of the data source will remain valid for a period of time that *exceeds the validity period of the individual data items.* For example, if a university has a Linked Data page which lists the graduate courses it currently offers, and even if the listed courses change frequently, the facts that this Linked Data page has data about graduate courses and that it has multiple such data items will remain valid for a period of time that far exceeds the validity of the individual course data. The goal is to cache such *statistical metadata* and to use it later to optimize query execution. This is unlike previous efforts that tried to cache *data* [1, 2, 3] and use it to answer Linked Data queries. The cached *metadata* will be used for the sole purpose of query optimization; out-of-date metadata may result in sub-optimal query execution, but *results will always be fresh.*

4.1 Candidate Metadata for Query Optimization

In the following, we elaborate on some candidate metadata items and provide few examples of how they can be used to optimize query execution (the examples, however, are not meant to provide a detailed, step-by-step description of the query execution process). Also, we note here that these metadata items capture the status of the *explicitly mentioned* RDF triples in each Linked Data document; the effect of reasoning (i.e., the *implicitly entailed* RDF triples) will be discussed in Section 5:

1. **List of classes used.**
2. **Total number of instances of each class.**

   Example 4.1. Consider the following query, which asks for the names of all students (i.e., instances of class *ex:student*):

   ```
   SELECT ?n WHERE {
   ?s a ex:student.
   ?s foaf:name ?n.}
   ```

   Listing 1: SPARQL Example

   Suppose that the current retrieval queue has three URIs in the following order *(uri1, uri2, uri3)* and the metadata cache has information that only *uri2* and *uri3* have instances of *ex:student* class. The query execution engine can use this information to give higher retrieval priority to *uri2* and *uri3*. Furthermore, if the cache contains information that *uri3* has more instances of *ex:student* than *uri2*, the engine should give *uri3* higher retrieval priority than *uri2.*

3. **List of predicates used.**

   Example 4.2. Consider the query of Example 4.1 and assume that the intermediate solutions have 10 URIs bound to "?s" (i.e., the query execution engine has found 10 URIs that match the first triple pattern). The engine can now use the cached metadata about which of these URIs have RDF triples with predicate *foaf:name* to give higher retrieval priority to those URIs which are known to have higher potential for matching the second triple pattern.

4. **List of distinct URIs mentioned:** This list can be used to give a higher retrieval priority for the documents which have something to say about the query-relevant URIs (these are URIs which are mentioned in the query or which appear in the intermediate partial solution mappings).

A more comprehensive list of metadata items is expected to be compiled after completing this phase of the proposed research. Then, we plan to propose an alternative to caching, in the form of *an extension* to the current proposals for linked dataset descriptions[19, 20], where Linked Data publishers can automatically extract query optimization-targeted metadata from their datasets and publish it in the same decentralized fashion as Linked Data. The base-line link-traversal query engine will be extended to enable the generation, caching and use of the candidate metadata items. Most of these metadata can easily be generated using the proper SPARQL queries against the target dataset [19]. However, we identified two tools that can be *extended* to programmatically generate the required metadata. One tool is RDFStats [21], which is an extensible RDF statistics generator and the other is make-void[3], which generates statistical information from RDf files as VoID expressions [19].

[3]https://github.com/cygri/make-void

5. SEMANTICS-BASED QUERY OPTIMIZATIONS

Adding reasoning support to link-traversal query execution can improve the following aspects of the query execution process: 1. *Finding more query answers* that only follow implicitly from the data. 2. *Guiding the link-traversal process* into more query relevant data sources. 3. *Query Optimization* (i.e., detecting conditions for early termination).

To achieve efficient query execution at the web scale, *reasoning and search* need to be intertwined into a *hybrid* query framework. As Fensel and van Harmelen [22] stated it: "interweave the reasoning process with the process of establishing the relevant facts and axioms through retrieval ... That way, retrieval and reasoning become two sides of the same coin —a process that aims for useful information derived from data on the Web". Previous work by Umbrich et al. [23], to incorporate *light-weight* reasoning (with RDFS and owl:sameAs) into link-traversal query execution, improved its *recall* by almost double. We propose to extend base-line link-traversal query execution to support more elaborate entailment regimes with a richer subset of OWL axioms. We will start with the most widely used subset of OWL, based on previous empirical analysis by [24].

5.1 Examples:

An example of how RDFS and OWL semantics can be exploited for query optimization is *detecting conditions for early termination.* It is now well-known that the cost of *data retrieval* dominates the cost of link-traversal query execution [25]. That is why the query engine's ability to entail that it does not need to exhaustively explore the entire reachable subweb is essential for efficient query execution. An example of this is exploiting *functional dependencies*, as the following example illustrates.

Example 5.1. Consider the functional dependency between a country and its capital, and the following query, which asks for the capital of the country which currency code is CAD.

```
SELECT ?capital WHERE {
?country rdf:type dbpedia-owl:Country .
?country dbpprop:currencyCode "CAD"@en .
?country  dbpedia-owl:capital ?capital .}
```

Listing 2: SPARQL query

A *semantics-aware* query engine should terminate its data retrieval process once it finds Ottawa as a capital for Canada. On the contrary, in a similar situation, a base-line link-traversal query engine will continue trying to find more capitals for Canada, until it exhaustively *retrieves* all the documents in the reachable subweb.

Reasoning can also help to guide the link-traversal process, by reasoning over the cached metadata (section 4), to prioritize the link traversal process. Consider the following example.

Example 5.2. Consider the query and the retrieval queue of Example 4.1. Assume that the metadata cache (Section 4) has information that *uri2* has 50 instances of class *ex:student* and *uri3* does not have any instances of class *ex:student.* Furthermore, assume that the metadata cache has information that *uri3* contains 150 instances of another class, *ex:gradStudent*, and assume that we know that class *ex:gradStudent* is a subclass of *ex:student* (i.e., we know that *"ex:gradStudent rdfs:subClassOf ex:student"*). A query execution engine that does not consider the effect of reasoning will give low priority to *uri3* and high priority to *uri2*. However, *uri3* has three times the number of students that *uri2* has and should be given the highest priority by a *reasoning-aware* query execution engine.

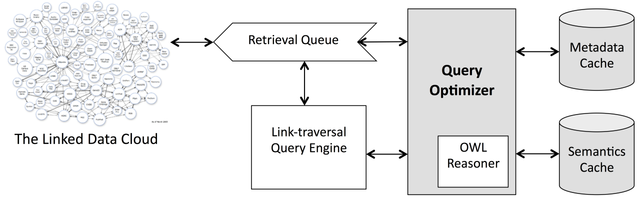

Figure 1: Architecture of a Hybrid Query Execution Framework

To this point, our plan is to start by identifying the situations where semantics can be used for optimizing query execution. Then, we will investigate how a link-traversal query engine can be extended to exploit such situations.

6. ARCHITECTURE OF A HYBRID QUERY EXECUTION FRAMEWORK

The Query Optimizer (Figure 1) is the key component responsible for the realization of the optimizations proposed in this paper. It can be thought of as an implementation of the *LD Oracle* concept (section 3.2). To achieve its optimization goals, the Query Optimizer will need to: *a*) consult a Metadata Cache (section 4), *b*) consult a Semantics Cache (section 5), *c*) incorporate an OWL Reasoner[4], *d*) use a basel-line link-traversal query engine[5], *e*) have access to the URI *Retrieval Queue* to be able to *revise and prioritize* the data retrieval process.

Throughout the link-traversal query execution process, the Query Optimizer's revision of the Retrieval Queue may result in one of the following outcomes:

1. Changing the lookup *priority* of URIs based on the Query optimizer's belief in their respective *potential* to contribute to the query answers.

2. Adding URIs that were not originally part of the queue but the Query Optimizer has reasons to believe that they can contribute answers.

3. Excluding URIs from the queue because the Query Optimizer has reasons to believe they cannot contribute answers to the query at hand (however, to guarantee completeness, this exclusion cannot happen based on *cached* statistics, it can only happen based on reasoning upon the intermediate results).

We will start with a base-line link-traversal query engine[5] and augment it with the proposed optimizations iteratively. At each iteration, we will experimentally validate the performance of the optimized engine against the base-line version of the same engine. For this purpose, we have extended the experimental setup used in [25].

[4]https://jena.apache.org
[5]http://squin.org

7. CONCLUSIONS

To conclude, the following are the key research hypotheses that we are investigating in this research: 1) It is possible to identify a set of *metadata* for online Linked Data which can be cached and utilized later for query optimization purposes. 2) Reasoning about Linked Data *Semantics* can significantly improve the performance of live Linked Data query execution. 3) A *hybrid* Link-traversal query execution framework can be designed to utilize such metadata and semantics for query optimization and can *exhibit* the following two properties: *a*) If the metadata is up-to-date, the framework can provide more efficient query execution and more complete query answers than *base-line* Link-traversal approaches. *b*) Regardless of the freshness of the metadata, the query answers will always be sound and complete.

8. REFERENCES

[1] Barry Bishop, Atanas Kiryakov, Damyan Ognyanov, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. Factforge: A fast track to the web of data. *Semantic Web*, 2(2):157–166, 2011.

[2] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *Int. J. Metadata Semant. Ontologies*, 3(1):37–52, November 2008.

[3] Andreas Harth, Jürgen Umbrich, Aidan Hogan, and Stefan Decker. YARS2: A federated repository for querying graph structured data from the web. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 211–224, 2007.

[4] Aidan Hogan, Andreas Harth, Jürgen Umbrich, Sheila Kinsella, Axel Polleres, and Stefan Decker. Searching and browsing linked data with SWSE: The semantic web search engine. *Web Semant.*, 9(4):365–401, December 2011.

[5] Olaf Hartig and Johann-Christoph Freytag. Foundations of traversal based query execution over linked data. In *HT*, pages 43–52, 2012.

[6] Olaf Hartig, Christian Bizer, and Johann Christoph Freytag. Executing sparql queries over the web of linked data. In *International Semantic Web Conference*, pages 293–309, 2009.

[7] Günter Ladwig and Thanh Tran. Sihjoin: Querying remote and local linked data. In *ESWC (1)*, pages 139–153, 2011.

[8] Tim Berners-Lee. Linked data - design issues. http://www.w3.org/DesignIssues/LinkedData.html. [Online].

[9] Sparql 1.1 entailment regimes. http://www.w3.org/TR/2013/REC-sparql11-entailment-20130321. [Online].

[10] Yingjie Li and Jeff Heflin. Using reformulation trees to optimize queries over distributed heterogeneous sources. In *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, pages 502–517, 2010.

[11] Heiner Stuckenschmidt, Richard Vdovjak, Geert-Jan Houben, and Jeen Broekstra. Index structures and algorithms for querying distributed rdf repositories. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 631–639, New York, NY, USA, 2004. ACM.

[12] Mathias Konrath, Thomas Gottron, Steffen Staab, and Ansgar Scherp. SchemEX - efficient construction of a data catalogue by stream-based indexing of linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 16(5), 2012.

[13] Jürgen Umbrich, Katja Hose, Marcel Karnstedt, Andreas Harth, and Axel Polleres. Comparing data summaries for processing live queries over linked data. *World Wide Web*, 14(5-6):495–544, October 2011.

[14] Günter Ladwig and Thanh Tran. Linked data query processing strategies. In *Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I*, ISWC'10, pages 453–469, Berlin, Heidelberg, 2010. Springer-Verlag.

[15] Olaf Hartig and Andreas Langegger. A database perspective on consuming linked data on the web. *Datenbank-Spektrum*, 10(2):57–66, 2010.

[16] Jürgen Umbrich, Marcel Karnstedt, Aidan Hogan, and Josiane Xavier Parreira. Hybrid sparql queries: Fresh vs. fast results. In *International Semantic Web Conference (1)*, pages 608–624, 2012.

[17] Olaf Hartig and M. Tamer Özsu. Linked data query processing. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 1286–1289, March 2014.

[18] Neil Coburn and Grant E. Weddell. A logic for rule-based query optimization in graph-based data models. In *DOOD*, pages 120–145, 1993.

[19] Michael Hausenblas un Zhao Keith Alexander, Richard Cyganiak. Describing linked datasets with the void vocabulary. http://www.w3.org/TR/void/. [Online].

[20] Thanassis Tiropanis, Wendy Hall, Nigel Shadbolt, David De Roure, Noshir S. Contractor, and Jim Hendler. The web science observatory. *IEEE Intelligent Systems*, 28(2):100–104, 2013.

[21] Andreas Langegger and Wolfram Wöß. Rdfstats - an extensible RDF statistics generator and library. In *Database and Expert Systems Applications, DEXA, International Workshops, Linz, Austria, August 31-September 4, 2009, Proceedings*, pages 79–83, 2009.

[22] Dieter Fensel and Frank van Harmelen. Unifying reasoning and search to web scale. *IEEE Internet Computing*, 11(2):96,94–95, 2007.

[23] Jürgen Umbrich, Aidan Hogan, Axel Polleres, and Stefan Decker. Improving the recall of live linked data querying through reasoning. In *RR*, pages 188–204, 2012.

[24] Birte Glimm, Aidan Hogan, Markus Krötzsch, and Axel Polleres. OWL: yet to arrive on the web of data? In *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012*, 2012.

[25] Olaf Hartig and M. Tamer Özsu. Reachable subwebs for traversal-based query execution. In *WWW (Companion Volume)*, pages 541–546, 2014.