SQUIN: A Traversal Based Query Execution System for the Web of Linked Data

Olaf Hartig
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
ohartig@uwaterloo.ca

ABSTRACT

The World Wide Web (WWW) currently evolves into a Web of Linked Data where content providers publish and link their data as they have done with hypertext for the last 20 years. We understand this emerging dataspace as a huge, distributed database which is –at best– partially known to query execution systems. To tap the full potential of the Web, such a system must be able to answer a query using data from initially unknown data sources. For this purpose, traditional query execution paradigms are unsuitable because those assume a fixed set of potentially relevant data sources beforehand.

We demonstrate the query execution system SQUIN which implements a novel query execution approach. The main idea is to integrate the traversal of data links into the result construction process. This approach allows the execution engine to discover potentially relevant data during the query execution.

In our demonstration, attendees can query the Web of Linked Data using SQUIN and, thus, learn about the new query execution approach. Furthermore, attendees can experience the suitability of the approach for Web applications by using a simple, Linked Data based mash-up implemented on top of SQUIN.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*query processing, distributed databases*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software

Keywords

link traversal based query execution; Linked Data; Web of Data

1. INTRODUCTION

During recent years an increasing number of data providers have adopted Berners-Lee's *Linked Data principles* [2] for publishing and linking structured data on the Web. We recall that these principles use: (i) HTTP, as data access protocol, (ii) HTTP scheme based URIs, as identifiers for entities described in the data, and (iii) RDF, as the data model to represent data. An HTTP URI mentioned in an RDF triple may then be understood as a *data link* that enables clients to retrieve more data by looking up the URI on the Web. The adoption of these principles has lead to the creation of a globally distributed dataspace, often called the *Web of Linked Data* [9].

The execution of declarative, SQL-like queries over Linked Data readily available from a large number of sources provides enormous potential. An approach to enable execution of such queries is to populate a centralized repository similar to the collection of Web documents managed by search engines for the Web. Using such a repository it is possible to provide almost instant query results. This capability comes at the cost of setting up and maintaining the centralized repository. Furthermore, users of such an interface for querying Linked Data are restricted to the portion of the Web of Linked Data that has been copied into the repository.

In our work we adopt an alternative view on querying Linked Data. We conceive the Web of Linked Data as a distributed database system. Querying the Web of Linked Data itself opens possibilities not conceivable before; it enables users to benefit from a virtually unbounded set of up-to-date data.

However, the Web of Linked Data is different from traditional distributed database systems: Although there exist approaches to provide source-specific query processing interfaces [3], we cannot assume that all data publishers provide such a service. Hence, the primary approach to access Linked Data on the Web is by looking up URIs. Therefore, in this scenario a query execution system cannot distribute the execution of (sub)queries to remote data sources. Instead, such a system has to retrieve data for local processing.

Further distinguishing characteristics are the unbounded nature of the Web and the lack of a complete database catalog. Due to these characteristics it is impossible to know all data sources that might contribute to the answer of a query. In this context, traditional query execution paradigms are insufficient because those assume that the query system (or the user) has information about the existence of any potentially relevant data source.

In this paper we present SQUIN – a full query execution system that is tailored to the characteristics of the Web of Linked Data. In particular, SQUIN implements a novel query execution approach proposed in our earlier, conceptual work [6, 7]. The general idea of this approach, which we call *link traversal based query execution*, is to intertwine an incremental construction of query results with the traversal of (certain) data links. More precisely, SQUIN evaluates queries over an initially empty, query-local dataset that is continuously augmented with data from the Web. This data is recursively discovered by looking up URIs mentioned in those RDF triples that are used for constructing the query result. Hence, SQUIN discovers and retrieves data that might be relevant for answering a query during the query execution process itself.

The most important feature of SQUIN is the possibility to use data from initially unknown data sources. This possibility allows for serendipitous discovery and, thus, enables applications that tap the full potential of the Web of Linked Data. Another feature is that SQUIN does not require any a-priori information about the queried Web. As a consequence, SQUIN can directly be used without having to wait for the completion of an initial data load phase or any other type of preprocessing. Hence, SQUIN is most suitable for an "on-demand" live querying scenario where freshness and discovery of results is more important than an almost instant answer.

The remainder of this paper is organized as follows: As a foundation of our work, Section 2 provides well-defined semantics for the queries supported by SQUIN. Section 3 describes the general idea of link traversal based query execution. Section 4 introduces the approach that implements this idea in SQUIN. Finally, Section 5 describes what we present during the demonstration session.

2. THEORETICAL FOUNDATIONS

Even if we focus on queries over a "database" that is not completely available to our query system, a precise definition of queries and of expected query results is important to provide soundness and completeness guarantees for query execution approaches and implementations thereof. Queries executed by SQUIN are expressed using SPARQL [10], which has the same expressive power as relational algebra [1] and is the de facto query language for RDF data. Hence, the original SPARQL semantics assumes queries over a-priory defined sets of RDF triples. To use SPARQL as a language for queries over the Web of Linked Data, we have to adjust the semantics by redefining the scope for evaluating SPARQL expressions. As a basis for such an adjustment we need a data model that formally captures the concept of a Web of Linked Data. In this section we introduce these theoretical foundations of our work.

Due to the limited space we focus on conjunctive queries in this paper. Our earlier work covers the complete core fragment of SPARQL 1.0 [5]. Currently, SQUIN supports this fragment except for the OPT operator (which introduces non-monotonicity).

In the following we first recall the basics of RDF and SPARQL. Then, we introduce our data model and define a query semantics for which link traversal based query execution is sound and complete.

Preliminaries

RDF triples are tuples from the set $\mathcal{T} := (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ where $\mathcal{U}$, $\mathcal{B}$, and $\mathcal{L}$ are pairwise disjoint, countably infinite sets of URIs, blank nodes, and literals, respectively. To denote the set of all URIs in an RDF triple $t \in \mathcal{T}$ we write $\mathrm{uris}(t)$.

A conjunctive query in SPARQL is represented via a *basic graph pattern (BGP)*. Such a BGP is a finite set of triple patterns which are defined as follows: Let $\mathcal{V}$ be an infinite set of query variables ($\mathcal{V}$ is disjoint from $\mathcal{U}$, $\mathcal{B}$, and $\mathcal{L}$, respectively), then, a *triple pattern* is a tuple $tp \in (\mathcal{V} \cup \mathcal{U}) \times (\mathcal{V} \cup \mathcal{U}) \times (\mathcal{V} \cup \mathcal{U} \cup \mathcal{L})$. To denote the set of all variables in triple patterns of a BGP B we write $\mathrm{vars}(B)$.

A *valuation* in the context of SPARQL is a partial mapping $\mu : \mathcal{V} \to (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. The *application* of a valuation μ to a triple pattern tp, denoted by $\mu[tp]$, is the triple pattern that we obtain by replacing the variables in tp according to μ. Similarly, a valuation μ is applied to a BGP B by $\mu[B] := \{\mu[tp] \mid tp \in B\}$. It holds that $\mu[B]$ is a set of RDF triples, if $\mathrm{vars}(B) \subseteq \mathrm{dom}(\mu)$.

Data Model

For simplicity, we assume a static view of the Web; i.e., no changes are made to the data on the Web during the execution of a query.

We define a *Web of Linked Data*[1] as a tuple $W = (D, data, adoc)$ that consists of the following elements: D is a set of symbols which we use to formally capture the concept of Web documents that can be obtained by looking up URIs in an implementation of W. Hereafter, we call each $d \in D$ an *LD document*. Mapping $data : D \to 2^{\mathcal{T}}$ associates each LD document with a finite set of RDF triples. Hence, $data$ is a total mapping such that $data(d)$ is finite for all $d \in D$. To denote the set of all RDF triples in W we write $\mathrm{AllData}(W)$; i.e., $\mathrm{AllData}(W) := \bigcup_{d \in D} data(d)$. Finally, $adoc : \mathcal{U} \to D$ is a partial, surjective mapping which models the fact that looking up URI $u \in \mathrm{dom}(adoc)$ (in an implementation of W) results in the retrieval of the document represented by $adoc(u) = d \in D$. We may understand d as the authoritative source of data (in W) about URI u. However, u may also be used in the data of other documents. In fact, the Linked Data principles encourage such a practice. Then, using URI u in the data of LD document $d' \in D$ constitutes a *data link* to LD document $d = adoc(u)$.

We understand a *Linked Data query* as a function over any possible Web of Linked Data (i.e., any 3-tuple that satisfies the definition above). While the three elements, D, $data$, $adoc$, provide the basis for defining semantics of such queries, we emphasize that these elements are not directly available to query execution systems.

Query Semantics

Due to the openness and distributed nature of the WWW, it is impossible to compute query results that are complete w.r.t. all Linked Data on the Web [5]. Thus, to provide a well-defined semantics for Linked Data queries that can be computed completely, we have to limit our understanding of completeness. However, instead of restricting queries to data from a fixed set of sources selected or discovered beforehand, we have proposed a family of reachability based query semantics that allow systems to make use of initially unknown data and sources [5]. Informally, the scope of queries under such a semantics is all data in a certain, reachable part of the queried Web of Linked Data (the notion of reachability is where these semantics differ among one another). In the following we define the reachability based semantics that SQUIN supports.

For the definition, we let $W = (D, data, adoc)$ be a Web of Linked Data; B be a BGP; and $S \subset \mathcal{U}$ be a finite set of seed URIs. Then, we define reachability of LD documents as follows: An LD document $d \in D$ is *B-reachable from S in W* if it holds that:

1. There exists a seed URI $u \in S$ such that $adoc(u) = d$; or
2. there exist $d' \in D$, $t \in data(d')$, $u \in \mathrm{uris}(t)$, and a valuation μ such that (i) d' is B-reachable from S in W, (ii) $t \in \mu[B]$, and (iii) $adoc(u) = d$.

Based on reachability of LD documents we define a reachable part of a Web of Linked Data as a "subweb" that covers all reachable LD documents. Formally: The *(S, B)-reachable part of W* is a Web of Linked Data $W_{\mathfrak{R}}^{(S,B)} = (D_{\mathfrak{R}}, data_{\mathfrak{R}}, adoc_{\mathfrak{R}})$ where:

$$D_{\mathfrak{R}} := \{d \in D \mid d \text{ is } B\text{-reachable from } S \text{ in } W\},$$

$$data_{\mathfrak{R}}(d) := data(d) \text{ for all } d \in D_{\mathfrak{R}}, \text{ and}$$

$$adoc_{\mathfrak{R}}(u) := adoc(u) \text{ for all } u \in \{u \in \mathcal{U} \mid adoc(u) \in D_{\mathfrak{R}}\}.$$

We now use the concept of a reachable part to define a notion of conjunctive Linked Data queries (CLD queries). The *CLD query* that uses B and S, denoted by $\mathcal{Q}^{B,S}$, is a Linked Data query that, for any Web of Linked Data W, is defined by:

$$\mathcal{Q}^{B,S}(W) := \{\mu \mid \mu \text{ is a valuation with } \mathrm{dom}(\mu) = \mathrm{vars}(B) \text{ and } \mu[B] \subseteq \mathrm{AllData}(W_{\mathfrak{R}}^{(S,B)})\}.$$

[1]In this paper we overload the term *Web of Linked Data* to homonymously refer to the mathematical structure that we define as well as to the World Wide Web which presents an actual implementation of such a structure. Nonetheless, it should be clear from the context which meaning each use of the term refers to.

The link traversal based query execution approach introduced in the following section is sound and complete for CLD queries [7].

3. EXAMPLE EXECUTION

In this section we use an example to informally describe the general idea of link traversal based query execution as implemented in SQUIN. For a formally defined execution model we refer to [7].

Let $\mathcal{Q}^{B_{\mathsf{ex}},S_{\mathsf{ex}}}$ be a CLD query. Suppose the BGP of this query is $B_{\mathsf{ex}} = \{tp_1, tp_2, tp_3, tp_4\}$ where:[2]

$$tp_1 = (?p, \mathsf{producedBy}, \mathsf{producer1}), \quad tp_2 = (?p, \mathsf{name}, ?pn),$$
$$tp_3 = (?o, \mathsf{offeredProduct}, ?p), \text{ and } \quad tp_4 = (?o, \mathsf{offeredBy}, \mathsf{vendor1}).$$

The query asks for all products from producer 1 for which we find offers from vendor 1. Let the seed URIs of this query be the URIs of all entities mentioned in B_{ex}; i.e., $S_{\mathsf{ex}} = \{\mathsf{producer1}, \mathsf{vendor1}\}$. We emphasize that URIs in this example are *http://...* URIs; for brevity, however, we denote them using simple symbols such as producer1.

For the sake of the example, suppose the following elements exist in the queried Web of Linked Data $W = (D, data, adoc)$:

$$adoc(\mathsf{producer1}) = d_{\mathsf{Pr1}} \in D, \quad adoc(\mathsf{product2}) = d_{\mathsf{p2}} \in D,$$
$$adoc(\mathsf{vendor1}) = d_{\mathsf{V1}} \in D, \quad adoc(\mathsf{offer1\text{-}1}) = d_{\mathsf{off1.1}} \in D,$$
$$(\mathsf{product2}, \mathsf{producedBy}, \mathsf{producer1}) \in data(d_{\mathsf{Pr1}}),$$
$$(\mathsf{product2}, \mathsf{name}, \text{"Product 2"}) \in data(d_{\mathsf{s}}),$$
$$(\mathsf{offer1\text{-}1}, \mathsf{offeredBy}, \mathsf{vendor1}) \in data(d_{\mathsf{V1}}), \text{ and}$$
$$(\mathsf{offer1\text{-}1}, \mathsf{offeredProduct}, \mathsf{product2}) \in data(d_{\mathsf{off1.1}}).$$

Link traversal based query execution usually starts with an empty, query-local dataset. We obtain some seed data by looking up the seed URIs from the query. For our example query with the seed URIs producer1 and vendor1, we retrieve two sets of RDF triples, $data(d_{\mathsf{Pr1}})$ and $data(d_{\mathsf{V1}})$. We add these triples to the local dataset.

Now, we begin an iterative process which is based on those RDF triples in the query-local dataset that match any of the triple patterns of our query. During each step of the process we use such a triple (i) to construct a valuation and (ii) to retrieve more data by looking up the URIs mentioned in the triple. Looking up these URIs presents a traversal of data links which may result in an augmentation of the query-local dataset. Such an augmentation may allow us to discover more matching triples and, thus, to perform further steps. The valuation that we construct during such a step may either be based solely on the current matching triple and, thus, cover only the corresponding triple pattern; or the valuation extends a previously constructed valuation such that it covers multiple triple patterns from the query. Ultimately, we are interested in those valuations that cover all triple patterns, because these valuations are elements of the query result. The whole process continues until it reaches a fixed point (which may not exist, if we assume the queried Web is infinitely large due to data generating servers [5, 7]).

For triple pattern tp_1 in our example query, the local dataset contains matching triple $(\mathsf{product2}, \mathsf{producedBy}, \mathsf{producer1})$, originating from (seed) $data(d_{\mathsf{Pr1}})$. Thus, we may construct a valuation $\mu_1 = \{?p \rightarrow \mathsf{product2}\}$ and look up the URIs mentioned in the (matching) triple. While a second lookup of URI producer1 is not necessary, we look up producedBy and product2; the lookup of product2 allows us to augment the query-local dataset with $data(d_{\mathsf{p2}})$.

In the augmented dataset we now find a matching triple for tp_2. Using this triple we can extend valuation μ_1 by adding a binding for variable *?pn*; we obtain valuation $\mu_2 = \{?p \rightarrow \mathsf{product2}, ?pn \rightarrow \text{"Product 2"}\}$, which already covers the first two triple patterns of our query. Notice, constructing μ_2 is only possible because we retrieved $data(d_{\mathsf{p2}})$. However, before we discovered the first matching triple and looked up the URI product2, we neither knew about the existence of LD document d_{p2} nor about the matching triple that allows us to construct μ_2. Hence, by traversing data links we answer queries based on data from initially unknown sources.

We may proceed with our execution strategy by using the matching triple for triple pattern tp_4 that we find in the data from the other seed document, d_{V1}. Eventually, we may report the following valuation as an element of the query result: $\{?p \rightarrow \mathsf{product2}, ?pn \rightarrow \text{"Product 2"}, ?o \rightarrow \mathsf{offer1\text{-}1}\} \in \mathcal{Q}^{B_{\mathsf{ex}},S_{\mathsf{ex}}}(W)$.

[2]Symbols with a leading question mark denote variables.

4. OUR IMPLEMENTATION APPROACH

We emphasize that the link traversal based query execution strategy illustrated in the previous section presents a general idea rather than a concrete (implementable) algorithm. A variety of approaches for implementing this idea are conceivable. For SQUIN, we use an implementation approach that adopts the well-known iterator model.

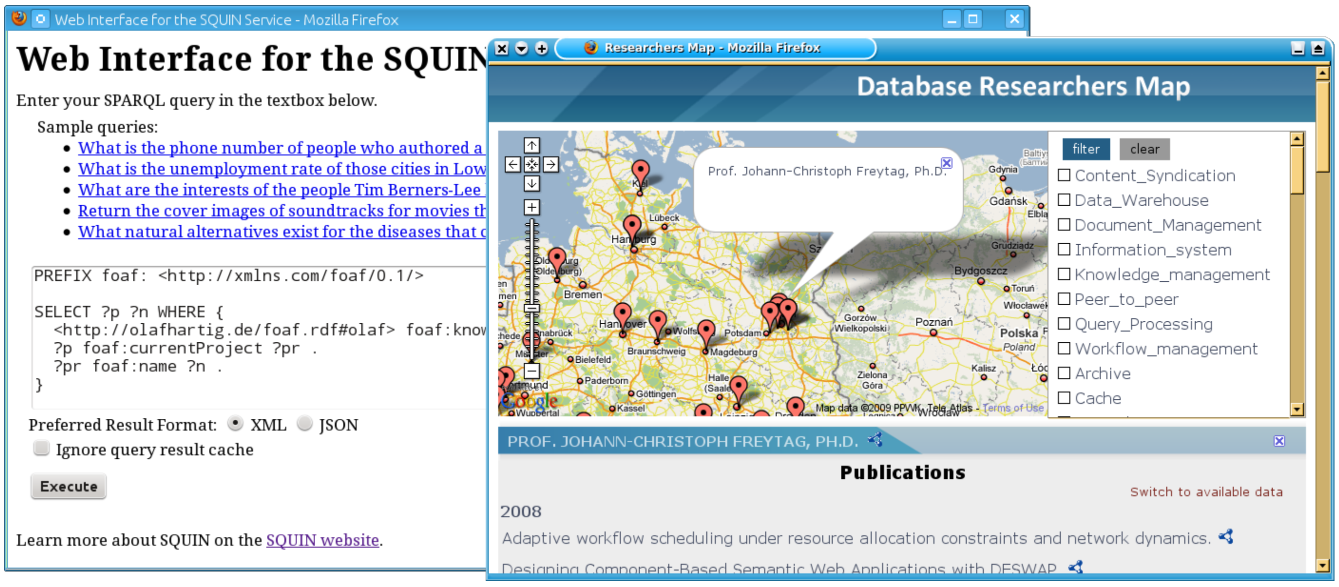
As a basis for the iterator based implementation approach we use logical execution plans that prescribe an order for the triple patterns of a given CLD query. While selecting such a plan is an optimization problem, traditional, cost-based plan selection is unsuitable in our scenario: Estimating the costs of a plan requires information about reachable data and data sources. Such information is not available when we start a link traversal based query execution. As a consequence, in SQUIN we use a heuristic for plan selection [4].

As a physical implementation of our logical plans we use a synchronized pipeline of iterators such that the i-th iterator is responsible for the i-th triple pattern (as given by the selected order). Each iterator I_i reports valuations that cover the first i triple patterns. To produce these valuations, I_i executes three steps repeatedly: First, I_i consumes a valuation μ' from its direct predecessor[3] I_{i-1} and applies this valuation to its triple pattern tp_i, resulting in a triple pattern $tp'_i = \mu'[tp_i]$; second, I_i (tries to) find matching triples for tp'_i in the query-local dataset; third, I_i uses each of these matching triples to (i) retrieve more data and (ii) to (iteratively) report valuations that are extensions of the current input valuation μ' (recall the extension of μ_1 to μ_2 in our example execution in Section 3).

Hence, in contrast to iterators usually used for computing queries over a fixed dataset, calling an iterator in SQUIN may have the side-effect of an augmentation of the query-local dataset (as desired for an implementation of link traversal based query execution).

Even if our iterators support a pipelined execution (i.e. complete materialization of intermediate results is not necessary), the execution may block temporarily when an iterator waits for the completion of certain URI lookups. We investigated concepts to avoid such a temporary blocking [6]. In particular, we studied the possibility to postpone processing of valuations for which necessary URI lookups are still pending. To conceptually integrate this idea into our iterator based implementation approach we extended the iterator model with a `Postpone` function. The general semantics of this function is to treat the element most recently reported by the `GetNext` function as if this element has not yet been reported. Hence, `Postpone` "takes back" the postponed element and keeps it for a later call of `GetNext` (in our particular case, these elements are the valuations computed by our iterators). Our iterators use this extension to temporarily ignore those valuations whose processing would cause blocking. In our experiments this approach results in reducing the overall query execution times to about 50% [6].

[3]We assume that the first iterator, I_1, is chained to a special iterator that provides a single, empty valuation $\mu_\emptyset$ where $\mathrm{dom}(\mu_\emptyset) = \emptyset$.



Web Interface for the SQUIN

Enter your SPARQL query in the textbox below.

Sample queries:

- What is the phone number of people who authored a
- What is the unemployment rate of those cities in Low
- What are the interests of the people Tim Berners-Lee
- Return the cover images of soundtracks for movies th
- What natural alternatives exist for the diseases that c

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?p ?n WHERE {
  <http://olafhartig.de/foaf.rdf#olaf> foaf:know
  ?p foaf:currentProject ?pr .
  ?pr foaf:name ?n .
}
```

Preferred Result Format: [✓ XML] JSON

Ignore query result cache

Learn more about SQUIN on the SQUIN website.

Database Researchers Map

Prof. Johann-Christoph Freytag, Ph.D.

- Content_Syndication
- Data_Warehouse
- Document_Management
- Information_system
- Knowledge_management
- Peer_to_peer
- Query_Processing
- Workflow_management
- Archive
- Cache

PROF. JOHANN-CHRISTOPH FREYTAG, PH.D.

Publications

Switch to available data

2008

Adaptive workflow scheduling under resource allocation constraints and network dynamics.

Designing Component-Based Semantic Web Applications with DESWAP

Figure 1: Screenshots of the Web interface for a SQUIN service (left) and of Researchers Map (right).

5. THE DEMONSTRATION

We will demonstrate our query system SQUIN as well as a mash-up application that queries the Web of Linked Data using SQUIN.

Our Query Execution System

Our Linked Data query execution system SQUIN is available[4] as Free Software; it is portable to any major platform due to its implementation in Java. Multiple options for using SQUIN exist: First, it may be used as a Java library that can be integrated in Web applications. Via its command line interface SQUIN may be used as a standalone tool. However, SQUIN can also be set up as a Web service that provides a machine accessible interface as well as a simple human user Web interface (cf. Figure 1). The machine accessible interface implements the SPARQL protocol [3]. Therefore, publicly available SQUIN services can be accessed like ordinary SPARQL endpoints on the Web. Of course, it is also possible to set up a private SQUIN service dedicated to a specific application.

For the demonstration session, we shall provide a public SQUIN service with a Web interface that enables attendees to issue (monotonic) SPARQL-based Linked Data queries over the live WWW. Thus, attendees can experience link traversal based query execution via this Web interface and an accompanying monitoring component. This monitor logs the URI lookups and triple pattern evaluations that happen during the execution of queries issued at the Web interface.

An Example Application

In addition to providing the attendees direct access to our query system we will demonstrate a simple Web application implemented on top of a dedicated SQUIN service. This application, called Researchers Map [8], provides a map of professors from the German database community; the list of professors in the map can be filtered by research interests; selecting a professor opens a list of her/his publications (cf. Figure 1). Researchers Map is a simple mash-up application that is solely based on the results of queries executed over Linked Data on the WWW.

[4]http://squin.org

We developed Researchers Map to demonstrate that our query execution approach is suitable for applications that consume Linked Data. During the demonstration session, attendees can use the Researchers Map to learn about the German database community and, more importantly, to understand how Web applications can benefit from link traversal based query execution and Linked Data.

6. REFERENCES

[1] R. Angles and C. Gutierrez. The Expressive Power of SPARQL. In *Proc. of the 7th International Semantic Web Conference (ISWC)*, 2008.

[2] T. Berners-Lee. Design Issues: Linked Data. Online at http://www.w3.org/DesignIssues/LinkedData.html, 2006.

[3] K. G. Clark, L. Feigenbaum, and E. Torres. SPARQL Protocol for RDF. W3C Recommendation, Jan. 2008.

[4] O. Hartig. Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. In *Proc. of the 8th Extended Semantic Web Conference (ESWC)*, 2011.

[5] O. Hartig. SPARQL for a Web of Linked Data: Semantics and Computability. In *Proc. of the 9th Extended Semantic Web Conference (ESWC)*, 2012.

[6] O. Hartig, C. Bizer, and J.-C. Freytag. Executing SPARQL Queries over the Web of Linked Data. In *Proc. of the 8th International Semantic Web Conference*, 2009.

[7] O. Hartig and J.-C. Freytag. Foundations of Traversal Based Query Execution over Linked Data. In *Proc. of the 23rd ACM Hypertext Conference (HT), Linked Data track*, 2012.

[8] O. Hartig, H. Mühleisen, and J.-C. Freytag. Linked Data for Building a Map of Researchers. In *Proc. of the 5th Workshop on Scripting and Development for the Semantic Web (SFSW) at ESWC*, 2009.

[9] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition, 2011.

[10] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, Jan. 2008.