Understanding Linked Open Data as a Web-Scale Database

Michael Hausenblas, Marcel Karnstedt
Digital Enterprise Research Institute (DERI), NUI Galway
IDA Business Park, Lower Dangan, Galway, Ireland
Email: michael.hausenblas@deri.org, marcel.karnstedt@deri.org

***Abstract*—While Linked Open Data (LOD) has gained much attention in the recent years, requirements and the challenges concerning its usage from a database perspective are lacking. We argue that such a perspective is crucial for increasing acceptance of LOD. In this paper, we compare the characteristics and constraints of relational databases with LOD, trying to understand the latter as a Web-scale database. We propose LOD-specific requirements beyond the established database rules and highlight research challenges, aiming to combine future efforts of the database research community and the Linked Data research community in this area.**

***Keywords*-linked data; Web-scale database; applications.**

I. Introduction

The Web of Data, more specifically *Linked Open Data* [1] (LOD), has gained tremendous momentum over the past couple of years. A significant number of large-scale datasets[1] have been published, adhering to the *Linked Data principles* [2]:

1) All items in a dataset should be identified using *URIs*;
2) All URIs should be *dereferenceable*: using HTTP URIs [3] allows looking up an item identified through an URI;
3) When looking up an URI, it leads to more data (typically represented in RDF [4]);
4) Links to URIs in other datasets should be included in order to enable the discovery of more data.

The Linking Open Data community project[2] was kicked-off in 2007 and has—at time of writing— produced over 100 data sets, providing over 6.7 billion RDF triples, interlinked by approximately 160 million RDF links [5] (cf. the LOD cloud in Fig. 1).

In contrast to the full-fledged Semantic Web vision, Linked Data is mainly about publishing structured data in RDF using HTTP URIs, hence lowering the entry barrier for data providers and data consumers. As much as the current Web (of documents) is mainly targeting human users, a particular strength of Linked Data is that applications can use it straightforward. Essentially, one can understand the LOD cloud as a single data-space, or simply put, as a *single, Web-scale database*.

[1]For example, DBpedia (http://dbpedia.org/), BBC music (http://www.bbc.co.uk/music/), LinkedGeoData (http://linkedgeodata.org/), and only recently by the New York Times (http://data.nytimes.com/)

[2]http://linkeddata.org

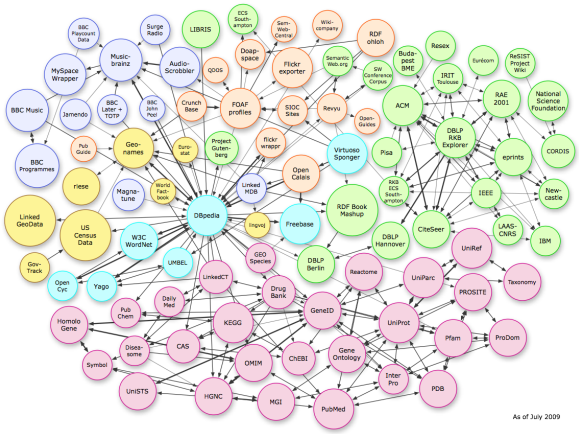

Figure 1. The LOD cloud in mid 2009, courtesy of Cyganiak and Jentzsch.

Given that an increasing number of applications uses LOD [6], one has to wonder what it takes to migrate (parts of) the relational database to the LOD setup. We hence argue that a *database perspective* for LOD is required to ensure its acceptance and realise a low-barrier adoption process. Although research into mapping relational data to RDF [7] and distributed query processing over the Web of Data [8] is known, a systematic analysis of LOD from a database perspective is, to the best of our knowledge, not available. With our work we aim to provide requirements and challenges concerning "LOD as a database"; our main contributions are: (i) to provide a comparison of relational databases and LOD from an application perspective, (ii) to highlight challenges in understanding LOD as a database and (iii) to identify requirements for specialised architectures for LOD to enable the essential principles of relational database management systems (RDBMS) as well as distinct features. Eventually, we want to motivate both the database research community and the Linked Data research community to mutually benefit from the issues we raise here and potentially use them as a starting point for further, joint research and development.

The remainder of this paper is structured as follows: we review related work in Section II and contrast application development in a relational setup with LOD-based development in Section III. We then compare relational databases and LOD in Section IV and discuss challenges concerning the usage of LOD from a database perspective in Section V. Section VI concludes our work.

II. Related Work

Recently, an interesting discussion about "One size does not fit all" was started by Stonebraker et al. [9], [10]. Essentially, the authors observe that relational databases and their underlying models do not fit current application needs. due to their broad support of features and the implicit performance overheads. In the context of Web databases and RDF data, several specialised data stores have developed, for example, graph databases such as Neo4J[3], array databases like Rasdaman[4], column stores like C-Store [11], or document-based systems like CouchDB[5]. A lot of research is also aiming at simplified key/value stores, such as Amazon's Dynamo [12]. However, the number of RDF stores that use RDBMS as back-ends—such as the relational column store [13]—indicates the popularity of the relational model in the RDF-based world. All these systems represent specialised solutions for specific application domains, but none of them is particularly designed for Linked Data.

Systems that are highly related to our work are mediator systems [14] and certain RDF stores, such as [15], [16]. In contrast, both types of systems were designed for relatively few endpoints with probably huge amounts of data, whereas Linked Data is rather formed by a huge number of small Web resources. Additionally, current RDF stores lack several features we discuss in Section IV, as they are usually not build for Linked Data in particular.

Linked Data can be understood as a read-only RESTful [17] Web service with representations restricted to RDF serialisations. Though Linked Data is conceptually "read-only", there exist attempts to extend it with "write" capabilities [18], [19]. We will, however, in the following operate under the assumption of the original, read-only case.

III. Application Development

Application development with relational databases is typically processed along the following steps:

1) **Design**: create the general design of the database using modelling tools like Entity-Relationship diagrams;
2) **Create**: transform the ER diagram into a relational schema, minimise and normalise, create the resulting relations and integrity constraints;
3) **Fill**: fill the relations with tuples;
4) **Access** and update: query and modify the data using a query and update language (such as SQL).

In partitioned DBMS, the partitioning scheme is additionally defined in step 2. Further, in distributed DBMS, the chain above is either processed once and deployed at all participating instances, or run and deployed separately for each instance. The latter requires a matching between the different schemata to agree on a globally used schema. In P2P approaches, each peer (i.e., instance) develops and fills its local database autonomously. Integration is usually achieved on the fly during query processing, although some systems revert to a schema matching approach using centralised components similar to distributed databases.

The *Linked Open Data publishing process* [20] usually starts off with existing, structured data in various formats (CSV, relational data, XML, etc.), which is converted to RDF. The domain is modelled either by creating a new local schema (in RDF Schema, OWL, etc.) or by reusing existing, wide-spread vocabularies (such as Dublin Core, FOAF, SIOC)[6] etc. with the emphasis on the latter, that is reusing rather than re-inventing. Typically, one then assigns URIs to the items in the data set on the instance level [21]. A key step after this is the interlinking with other datasets. There are typically two types of (RDF) links one deals with:

- `owl:sameAs`, establishing an identity between items in different data sets[7] for which generic interlinking frameworks are available [22], [23], and
- other, domain-specific RDF links such as `foaf:knows`, `dc:author`, where typically customised code is used to establish links between items.

IV. Comparison

A. Baseline Comparison

Current RDBMS can be characterised by a set of features they support. Table I lists these features, based on the *Codd's rules* [24], [25]. In the following, we discuss the applicability of the relational rules to Linked Open Data. Where applicable, we discuss how they manifest in LOD; note also that we not discuss the rows *Security*, *Transactions*, *Synchronisation* and *Safety* from Table I, as they do not apply to the read-only case of LOD).

Integration: Integration of data is a major requirement for Linked Data. From the application perspective, one expects to query [8] physically distributed data comparable to what RDBMS provide locally. However, with Linked Data, typically an additional step—the entity consolidation [28]—has to be performed in order to provide a consistent view on the data.

Operations and Language: Operations are the essence of structured queries and thus a crucial requirement for Linked Data as well. The *SPARQL Query Language for RDF* [26], a W3C standard, corresponds to the read-only part of SQL's Data Manipulation Language. Ongoing standardisation efforts[8] focus on the extension of SPARQL with update capabilities.

[3]http://neo4j.org
[4]http://www.rasdaman.org
[5]http://couchdb.apache.org

[6]For an overview see http://semanticweb.org/wiki/Ontology
[7]For example, http://dbpedia.org/resource/London and http://geonames.org/12345 identify the same real-world entity, that is, the city of London.
[8]http://www.w3.org/TR/sparql11-update/

Principle	RDBMS		LOD	
	Meaning	Instantiation	Meaning	Instantiation
Integration	Data are managed homogeneously and without any redundancies	Tables, rows, keys	With RDF, a homogeneous model is provided	Linked Data principles [2]
Operations and Language	The DBMS provides operations of accessing, creating, modifying, and deleting data	DQL, DML, DDL; SQL	A standard structured query language (for read-only operations) is required	SPARQL [26]
Catalogue	The database catalogue describes the data and is accessible as part of the database itself	Catalogue relations & views	RDF is self-descriptive; vocabularies should be described appropriately	Domain vocabularies
User Views	Each user can have a different and individual view on the data	`create view as select ...`	Views are not directly supported, but become mandatory for the database point of view	SPARQL `CONSTRUCT`
Integrity	Consistency and integrity of data are granted by the DBMS	Keys, constraints, normal-forms	Mitigated in the read-only environment	entity recognition
Security	Access to data is granted to only authorised users	Privileges, authorisation, authentic.	N/A	
Transactions	Transactions group multiple operations into a logical unit; integrity is granted on the transaction level	`commit, abort`	N/A	
Synchronisation	Concurrent accesses are strictly handled by the DBMS	ACID guarantees, locks	N/A	
Safety	Long-term accessibility to data is granted even in the case of failures	Logging & recovery	N/A (delegated to each data provider)	
Null Values	Missing information is systematically represented and handled	`NULL` semantic	Open World Assumption	schema-dependent [27]
Physical/Logical Data Independence	Changes to the physical/logical level require no changes to applications	Five-layer architecture	Guaranteed	Linked Data principles [2]
Distribution Independence	A distributed database should provide the user with a view as on a single centralised database	Distributed/federated DBMS	The inherent distribution of Linked Data should be transparent to the user	see Section V

Table I
APPLICABILITY OF RELATIONAL DATABASE RULES TO LOD.

Catalogue: Based on the RDF model and semantics [4] all data inherently follows a common schema, such as the notion of objects and data types. In order to represent domain-specific schema information, one uses RDF vocabulary languages such as RDF-S or OWL [29]. However, concerning schema consolidation, additional efforts are necessary in the LOD case, which will be discussed below.

User Views: SPARQL, the de facto standard query language for Linked Data, does not define views explicitly as they are known from RDBMS. With the SPARQL `CONSTRUCT` operation a basic support for views is available. However, what is currently missing are definitions about the up-to-date character of the so generated RDF graphs. This is mandatory for the support of views as known from the RDBMS world.

Integrity: Enabling a database-like view on Linked Data requires actions for enabling integrity as a fundamental prerequisite. In RDBMS, the main building blocks to enable integrity guarantees by the system are primary keys, foreign keys, integrity constraints and normal-forms of relations. These building blocks have to be defined independently from the application and have to be stored in the database catalogue. Primary and foreign keys refer to the problem of entity recognition on URI basis [22], [28]. Besides this, in the Linked Data scenario the problem of integrity is mitigated, as updates and deletions are not supported.

Null Values: In contrast to relational databases, LOD operates under the *Open World Assumption* [29][9]. It typically requires schema knowledge to decide how null values from RDBMS are exposed in RDF [27].

Physical/Logical Data Independence: Logical data independence is a non-issue in LOD, as the third Linked Data principle [2] ensures that the underlying data model in any case is RDF. Physical data independence is guaranteed through the fact that processing of LOD data is independent of the used representation (such as RDF/XML, RDFa, Turtle, etc.) [30].

Distribution Independence: This is a very crucial principle in the context of Linked Data, as it is inherently distributed. We will discuss according architectural approaches in Section V.

B. LOD-specific Features

After our discussion of the applicability of RDBMS rules to LOD, we now have a closer look at requirements that are specific for LOD as a Web-scale database. Table II summarises the identified special features, which are discussed in the following.

Web-scale Consumption: Linked Open Data spans the entire Web; datasets are provided by various publisher from all around the world [31]. This fact requires any solution

[9]In contrast to the *Closed World Assumption*, which essentially says that a statement, which is not known to be true is explicitly assumed to be false.

Principle	Meaning
Web-scale Consumption	Discovery, entity consolidation, (distributed) query, etc. have to scale to the size of the Web
Schema Integration	Employment and tight integration of schema matching and schema mapping techniques
Ranking	Combinations of IR and DB approaches are required
Reusable Identifiers	Public identifiers of well-known entities should be reused
Provenance	Knowing the origin of the data is essential
Mapping Keywords to Resources	Support for keyword-based search and mapping to URIs is required

Table II
SPECIAL REQUIREMENTS FOR A LOD DATABASE

in the Linked Data consumption process, such as discovery, entity consolidation, (distributed) query, etc. to potentially scale to the size of the Web.

Schema Integration: Beside a small set of widely-known vocabularies such as Dublin Core, FOAF, SIOC, etc. [20], typically one has to deal with various, sometimes overlapping, domain vocabularies [32]. From an application point-of-view this means that almost always schema matching and schema mapping techniques have to be employed in order to use the data. There is a large body of work in the databases community on schema matching, for example [33], that can and should be applied to LOD.

Ranking: LOD in a sense depends on both techniques from the Database community (for example, structured queries) and the *Information Retrieval* domain, such as ranking. Ranking results in a Web of Data indexer [34] is a mandatory requirement from an application perspective. Different ranking approaches, ranging from document-centric ones such as PageRank to dataset-centric ones [35] are needed to address different granularity levels.

Reusable Identifiers: Public identifiers of well-known entities such as places, people, products, etc. should be reused [36]. This will considerably improve the effectiveness, as well as the efficiency, of data and information integration processes. It requires the support of searching for and using already existing identifiers.

Provenance: Provenance can be understood as a specific form of context information necessary to process the data in order to: (i) assess the credibility of a piece of data (trust), (ii) disambiguate data[10] and (iii) for supporting other, non-technical decision making processes, such as licensing. As RDF has no built-in notion of provenance, one has to depend on external mechanisms such as *Named Graphs* (NG)[11], supported in the RDF query language SPARQL and by most RDF stores. Provenance has recently been identified as a pressing issue yielding pre-standardisation activities in W3C[12].

Mapping Keywords to Resources: Support of keyword-based search is a crucial requirement for at least two reasons: (i) enabling entry points into LOD data sets, serving as a disambiguation facility (input: keywords, output: set of URIs), such as provided by DBpedia[13], and (ii) for general-purpose free-text search. This text support has to be extended by advanced features known from IR and NLP in order to handle heterogeneity on instance level. While this is often achieved by according extensions in RDBMS, it has to be implemented as an integral part of LOD.

V. CHALLENGES AND FUTURE RESEARCH DIRECTIONS

Several of the requirements discussed in Section IV lead to one main conclusion: LOD as a database has not to provide ACID guarantees for transactions. This leads to significant alleviations in the design of such a system, and finally in its performance (cf. also [10]). However, the LOD-specific requirements pose a new set of challenges, as already highlighted above. In the following, we discuss two of the main challenges in more detail, namely suitable architectures and data descriptions.

A suitable architecture for managing LOD data is one major challenge. Certain information in the LOD cloud is understood to be static (e.g., archives), while others are rather dynamic, potentially updated in periodic time intervals. Thus, data freshness has to be guaranteed in general, particularly concerning data that is likely to change. Several proposals to address this issue are on the table[14]. However, little practical experience has been gathered so far. Based on existing works and application development experiences, we have identified the following general approaches:

1) Centralised repositories of Linked Data;
2) Live look-ups (combined with caches);
3) Total distribution in a P2P setup.

Option 1 assumes that all LOD data is maintained in a central repository (on a single machine or in a cloud setup[15]). Obviously, issues with this option are the time-consuming pre-processing, freshness and maintenance of the data, as well as scalability issues.

Approaches of the second category (option 2) apply live look-ups and can easily integrate new sources, while requiring less storage and processing resources. A drawback of this option is the dependency of the availability of the sources. One such approach is to try to leverage the

[10]For example, querying for the affiliation of a person should typically result in different affiliations for different time periods

[11]http://www.w3.org/2004/03/trix/

[12]http://www.w3.org/2005/Incubator/prov/

[13]http://lookup.dbpedia.org/

[14]http://esw.w3.org/topic/DatasetDynamics

[15]For example, http://lod.openlinksw.com/

correspondence between source addresses and identifiers contained in the sources to answer queries [8]. Recent approaches try to leverage lightweight data summaries (based on only schema information [37] or on schema and instance information [38]) that can be kept in memory to enable source selection for parallel look-ups. Live look-ups potentially can be used in combination with caches: LOD has—at least theoretically[16]—built-in support for caching through HTTP [3, Sec. 13]. Option 2 can further benefit from research on mediator systems [14] and federated databases, although these systems were developed for much smaller scales. See [38] for a deeper discussion of the different approaches from options 1 and 2.

Total distribution in a P2P manner (option 3) is a promising alternative due to its inherent scalability, decentralisation, guaranteed freshness and self-management [39]. However, research on P2P data management systems revealed other issues like efficiency of query processing, robustness and result completeness. The read-only character of LOD can be exploited in this context, as P2P systems usually replace ACID guarantees by the BASE (**BA**sically available, **S**oft state, **E**ventually consistent [40]) principle and are bound to the CAP theorem [41]. Further, the Open World Assumption used in the LOD suits P2P systems perfectly.

Data descriptions are mandatory for enabling users to meaningfully query LOD. Statistics can support that, but they are even more important for planning and optimising queries internally. Catalogue and statistics become a problem as soon as one applies a decentralised setup. Due to the characteristics of RDF, live queries could be used to query the according information. But, such queries will be usually expensive (many unbound patterns, i.e., querying large fractions of the Linked Data cloud) and introduce inappropriate delays. Querying and collecting description and statistics beforehand (i.e., off-line) are an option, which increasingly gain support [42].

VI. Conclusion

In this paper we have first argued that a database perspective for Linked Open Data (LOD) is beneficial and then discussed the applicability of the relational database rules to LOD. We have further discussed LOD-specific issues for a database perspective and highlighted challenges stemming from this view. We call out for rethinking current approaches for Linked Data management from both the database and Web science community, along the lines of current discussions in database research. This should result in specialised Web-scale database engines tailor-made for the requirements of the Linked Open Data cloud.

[16]http://webofdata.wordpress.com/2009/11/23/linked-open-data-http-caching/

Acknowledgment

Our work has partly been supported by the European Commission under Grant No. 217031, FP7/ICT-2007.1.2, Romulus project, under Grant No. 231335, FP7/ICT-2007.4.4 iMP project and by the Science Foundation Ireland under Grant No. 08/SRC/I1407 (Clique: Graph & Network Analysis Cluster).

References

[1] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data—The Story So Far," *IJSWIS, Special Issue on Linked Data*, 2009.

[2] T. Berners-Lee, "Linked Data, Design Issues," http://www.w3.org/DesignIssues/LinkedData.html, 2009.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," http://tools.ietf.org/html/rfc2616, IETF Network Working Group, Request for Comments: 2616, June 1999, 1999.

[4] G. Klyne, J. J. Carroll, and B. McBride, "Resource Description Framework (RDF): Concepts and Abstract Syntax," http://www.w3.org/TR/rdf-concepts/, RDF Core Working Group, W3C Recommendation 10 February 2004, 2004.

[5] C. Bizer, "The Emerging Web of Linked Data," *IEEE Intelligent Systems*, vol. 24, no. 5, pp. 87–92, 2009.

[6] M. Hausenblas, "Exploiting Linked Data to Build Web Applications," *IEEE Internet Computing*, vol. 13, no. 4, pp. 68–73, 2009.

[7] S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. Thibodeau, S. Auer, J. Sequeda, and A. Ezzat, "A Survey of Current Approaches for Mapping of Relational Databases to RDF," http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf, W3C RDB2RDF Incubator Group, Incubator Group Report January 08 2009, 2009.

[8] O. Hartig, C. Bizer, and J.-C. Freytag, "Executing sparql queries over the web of linked data," in *ISWC '09*, 2009, pp. 293–309.

[9] M. Stonebraker and U. Cetintemel, "One size fits all: an idea whose time has come and gone," in *ICDE '05*, 2005, pp. 2–11.

[10] 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 *VLDB '07*, 2007, pp. 1150–1160.

[11] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik, "C-Store: A Column-oriented DBMS." in *VLDB '05*, 2005, pp. 553–564.

[12] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *SIGOPS Operation Systems Review*, vol. 41, no. 6, pp. 205–220, 2007.

[13] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach, "Scalable semantic web data management using vertical partitioning," in *VLDB '07*, 2007, pp. 411–422.

[14] G. Wiederhold, "Mediators in the architecture of future information systems," *Computer*, vol. 25, no. 3, pp. 38–49, 1992.

[15] M. Cai and M. Frank, "RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network," in *WWW'04*, 2004, pp. 650–657.

[16] A. Harth and S. Decker, "Optimized index structures for querying rdf from the web," in *3rd Latin American Web Congress*, 2005, pp. 71–80.

[17] R. Fielding and R. Taylor, "Principled design of the modern Web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, 2002.

[18] T. Berners-Lee, J. Hollenbach, K. Lu, J. Presbrey, E. Prud'hommeaux, and mc schraefel, "Tabulator Redux: Browsing and Writing Linked Data," in *WWW 2008 Workshop: Linked Data on the Web (LDOW2008)*, Beijing, China, 2008.

[19] O. Ureche, A. Iqbal, R. Cyganiak, and M. Hausenblas, "Accessing Site-Specific APIs Through Write-Wrappers From The Web of Data," in *Semantics for the Rest of Us Workshop (SemRUs) at ISWC09*, 2009.

[20] C. Bizer, R. Cyganiak, and T. Heath, "How to Publish Linked Data on the Web," http://linkeddata.org/docs/how-to-publish, 2007.

[21] L. Sauermann and R. Cyganiak, "Cool URIs for the Semantic Web," http://www.w3.org/TR/cooluris/, W3C Semantic Web Education and Outreach Interest Group, W3C Interest Group Note 31 March 2008, 2008.

[22] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Silk A Link Discovery Framework for the Web of Data," in *WWW '09 Workshop: Linked Data on the Web (LDOW2009)*, 2009.

[23] O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang, "Semantic link discovery over relational data," *CIKM '09*, 2009.

[24] E. F. Codd, "Is Your DBMS Really Relational?" *ComputerWorld*, vol. 14, 1985.

[25] ——, "Does Your DBMS Run by the rules?" *ComputerWorld*, vol. 21, 1985.

[26] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," http://www.w3.org/TR/rdf-sparql-query/, RDF Data Access Working Group, W3C Recommendation 15 January 2008, 2008.

[27] K. Byrne, "Populating the Semantic Web - Combining Text and Relational Databases as RDF Graphs," Ph.D. dissertation, Edinburgh University, 2009.

[28] A. Harth, A. Hogan, R. Delbru, J. Umbrich, S. O'Riain, and S. Decker, "SWSE: Answers Before Links!" in *Semantic Web Challenge at ISWC07*, ser. CEUR Workshop Proceedings, vol. 295, 2007.

[29] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, 2008.

[30] M. Hausenblas, W. Slany, and D. Ayers, "A Performance and Scalability Metric for Virtual RDF Graphs," in *3^{rd} Workshop on Scripting for the Semantic Web (SFSW07)*, Innsbruck, Austria, 2007.

[31] M. Hausenblas, W. Halb, Y. Raimond, and T. Heath, "What is the Size of the Semantic Web?" in *I-Semantics '08*, 2008.

[32] L. Ding, P. Kolari, Z. Ding, and S. Avancha, "Using Ontologies in the Semantic Web: A Survey," in *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*. Springer, 2006.

[33] H. H. Do, *Schema Matching and Mapping-based Data Integration: Architecture, Approaches and Evaluation*. Saarbrücken, Germany, Germany: VDM Verlag, 2007.

[34] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello, "Sindice.com: a document-oriented lookup index for open linked data," *IJMSO*, vol. 3, no. 1, pp. 37–52, 2008.

[35] N. Toupikov, J. Umbrich, R. Delbru, M. Hausenblas, and G. Tummarello, "DING! Dataset Ranking using Formal Descriptions," in *WWW '09 Workshop: Linked Data on the Web (LDOW2009)*, 2009.

[36] P. Bouquet, H. Stoermer, W. Barczynski, and S. Bocconi, "Entity-centric Semantic Interoperability," in *Cases on Semantic Interoperability for Information Systems Integration: Practices and Applications*. IGI Global, 2007.

[37] H. Stuckenschmidt, R. Vdovjak, G.-J. Houben, and J. Broekstra, "Index structures and algorithms for querying distributed rdf repositories," in *WWW'04*, 2004, pp. 631–639.

[38] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K. Sattler, and J. Umbrich, "On lightweight data summaries for optimised query processing over linked data," DERI, NUI Galway, Tech. Rep., 2009.

[39] M. Karnstedt, *Query Processing in a DHT-Based Universal Storage*. AVM-Verlag, 2009, phD thesis, to appear.

[40] E. A. Brewer, "Towards robust distributed systems (abstract)," in *PODC'00*, 2000, p. 7, key note talk, slides available at http://www.cs.berkeley.edu/∼brewer/cs262b-2004/PODC-keynote.pdf.

[41] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002.

[42] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao, "Describing Linked Datasets - On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets'," in *WWW '09 Workshop: Linked Data on the Web (LDOW2009)*, 2009.