Linked Data Management

Olaf Hartig, Katja Hose, and Juan Sequeda

Synonyms

- Linked Data Query Processing
- Web of Data

Definitions

The term *Linked Data* refers to data that is made available on the Web in a structured, directly machine-processable form via the following four principles:

1. *"Use URIs as names for things*
2. *Use HTTP URIs so that people [and machines] can look up those names.*
3. *When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)*
4. *Include links to other URIs, so that they can discover more things."*
   (Berners-Lee 2006; Bizer et al 2009)

Per these principles, given a URI that names some thing (e.g., a person, a book, a concept, an abstract idea) [Principle 1], it is possible to look up this URI using the HTTP protocol [Principle 2] and retrieve a document with RDF data describing the thing [Principle 3]. This data may include other URIs to refer to other things, and these URIs can be looked up in the same manner [Principle 4].

Overview

When applying the above mentioned principles, data becomes interlinked on the Web. These links may not only point to additional data from the same website but also to other websites. This results in the emergence of a decentralized, globally distributed Web of Linked Data.

Since their introduction, these Linked Data principles have been adopted by various publishers (Bizer et al 2009). Today, there are significant amounts of Linked Data accessible on the Web (Mika and Potter 2012; Schmachtenberg et al 2014).

In addition to publishing data on the Web via a Linked Data interface, many data publishers also support other RDF-based mechanisms to access their datasets. One approach is in the form of *RDF dumps* that provide the complete dataset as a single downloadable file serialized in an RDF format. Another approach is through *SPARQL endpoints*, which are Web services that can be called to obtain the result of executing any given SPARQL query over the

Olaf Hartig
Linköping University, e-mail: olaf.hartig@liu.se

Katja Hose
Aalborg University, e-mail: khose@cs.aau.dk

Juan Sequeda
Capsenta, e-mail: juan@capsenta.com

dataset (Feigenbaum et al 2013). Alternative query interfaces have become available that are deliberately limited in the expressiveness of the queries they support; the purpose of these limitations is to achieve a restricted, more predictable server load per request and, thus, to increase the overall availability of the query-based interfaces. The most prevalent of these alternative query interfaces is the *Triple Pattern Fragment* (*TPF*) interface (Verborgh et al 2016).

While all these data publishing approaches are based on the RDF data model, it has to be emphasized that an isolated set of RDF triples (e.g., in an RDF dump or behind a SPARQL endpoint) is not Linked Data per se; what would make it Linked Data is the possibility to discover it on the Web by the aforementioned principles. Hence the fourth principle of including links between different RDF datasets is vital.

The remainder of this article first provides a general overview of different data management options to make use of Linked Data in applications and, thereafter, focuses on approaches to query Linked Data live on the Web by relying solely on the Linked Data publishing principles (i.e., URI lookups).

Key Research Findings

Various approaches towards linked data management and querying have been proposed in the literature. To provide an overview, we have categorized them into general approaches, foundations, source selection, source ranking strategies, query execution, and query optimization.

General Approaches to make use of Linked Data

Centralization. A first option to use Linked Data in an application is to obtain the application-relevant data (e.g., by crawling or by downloading the respective RDF dumps) and load this data into a database management system (DBMS) for RDF data that is set up for the application. DBMSs for RDF data, often referred to as *triple stores*, enable applications to query the data using SPARQL, the RDF query language. There are a number of commercial triple stores such as AllegroGraph, Amazon Neptune, Blazegraph, GraphDB, MarkLogic, Stardog, and Virtuoso. For an overview of research on techniques to build highly scalable triple stores refer to other entries in this encyclopedia (see the cross-references section below).

In contrast to the following options, copying data into a dedicated triple store set up for the sole purpose of supporting a particular application is perhaps the most efficient option in terms of query performance. A possible downside is that the triple store has to be maintained and the copied data may have to be kept in sync with the data in the original data sources on the Web.

Distributed Client-Server Processing. If the application uses a single dataset only (or multiple in isolation) and the maintainer of this dataset provides a Web interface to query the dataset, the application can be built based on queries to be executed via this interface.

Such an interface may be a *SPARQL endpoint* (see above). However, Buil-Aranda et al (2013) have shown empirically that public SPARQL endpoints on the Web may have availability issues.

The aforementioned *Triple Pattern Fragment interface* is a more reliable option (Verborgh et al 2016). In this case however, to execute SPARQL queries over the dataset exposed via this TPF interface, the application requires a client-side query execution engine that is capable of processing such queries based on TPF requests. A number of TPF-based query execution approaches have been proposed (Verborgh et al 2016; Van Herwegen et al 2015) with the state of the art being an adaptive approach by Acosta and Vidal (2015) that adopts the idea of using Eddy operators (Avnur and Hellerstein 2000).

In addition to using pure TPF interfaces, some authors propose extended versions of this interface with the goal of reducing the shortcomings of TPF (e.g., significant network load) by retaining (or even increasing) the overall query throughput of TPF-based client-server systems. Examples of such proposals are to augment *TPF responses with membership metadata* such as Bloom filters (Vander Sande et al 2015), and the *bindings-restricted Triple Pattern Fragment* (*brTPF*) interface that allows client-side query engines to use variations of the semijoin algorithm (Hartig and Buil-Aranda 2016).

Federated Processing. Federated query processing engines go beyond the aforementioned client-server setting by supporting the execution of queries over a federation of multiple servers. Each of these servers typically provides access to a different dataset. Therefore, given a SPARQL query, the goal of a federated query execution is to return a query result that is the same as if the query was executed directly over the union of the datasets in the federation.

Existing approaches to federated query processing in the Linked Data context can be categorized into three classes.

The first class consists of approaches that assume Linked Data can be accessed via query-based Web interfaces (e.g., SPARQL endpoints, TPF). These approaches are discussed in more detail in the "*Federated RDF query processing*" entry in this encyclopedia.

Approaches in the second class rely only on the Linked Data publishing principles (i.e., URI lookups) and, thus, do not require the additional availability of query-based interfaces to access the Linked Data. Such approaches, called *Linked Data query processing*, are the main focus of this article and will be discussed in the next sections.

The third class comprises *hybrid query approaches* which aim to leverage query-based interfaces to access Linked Data in combination with Linked Data query processing techniques. To date, the only proposals in this direction are by Umbrich et al (2012) which combines Linked Data query processing with access to a SPARQL endpoint that serves as a query-able cache of Linked Data. Additionally, Lynden et al (2013) proposes to integrate data retrieval from a SPARQL endpoint into a Linked Data query execution process.

Foundations of Linked Data Queries

A *Linked Data query* is a query over a Web of documents of RDF data that can be retrieved by URI lookups as per the Linked Data publishing principles. Such queries may be not only

about the topology of the interlinked RDF documents but also, and perhaps more interesting, about the data in and across these documents. In particular, existing approaches to process Linked Data queries (as discussed in the next sections) focus on queries that are expressed using the conjunctive fragment of the SPARQL query language and that are to be evaluated in terms of the union of the data in the interlinked documents. An initial approach to providing a formal semantics for these queries has been proposed by Bouquet et al (2009). More recently, Hartig (2012) has improved this work in three ways: i) covering the whole core fragment of SPARQL (i.e., including OPTIONAL, FILTER, UNION), ii) showing formally that completeness of query results w.r.t. all Linked Data cannot be guaranteed, and iii) introducing a more restrictive formal query semantics for which completeness can be guaranteed.

As an alternative to adopting SPARQL expressions as a language for Linked Data queries, a number of dedicated Linked Data query languages have been proposed, namely, LDPath (Schaffert et al 2012), NautiLOD (Fionda et al 2015), and LDQL (Hartig and Pérez 2016). However, with the exception of the swget approach (Fionda et al 2014) for NautiLOD, techniques for processing Linked Data queries have focused on (conjunctive) SPARQL expressions.

Source Selection Strategies for Linked Data Queries

Any execution of Linked Data queries involves the retrieval of data by looking up URIs. Existing approaches for selecting the URIs to be looked up during the execution of a given query can be classified into the following three classes:

Index-based approaches determine the URIs to look up during query execution by using a pre-populated index (Harth et al 2010; Umbrich et al 2011). A typical example of such an index uses patterns of RDF triples as index keys (Ladwig and Tran 2010). Given such a pattern, the corresponding index entry is a set of URIs where each of these URIs, when looked up, allows the query engine to retrieve some data that contains an RDF triple that matches the pattern. Further index structures for selecting URIs in Linked Data query processing have been studied by Harth et al (2010), Umbrich et al (2011), Tian et al (2011), and Paret et al (2011).

After such an index has been populated and can be used for query processing, the index has to be maintained. Maintenance may comprise indexing additionally discovered URIs and ensuring that the index is up to date (Umbrich et al 2011). The latter is necessary because the data to be retrieved from indexed URIs may change over time. While such an index maintenance is similar to maintaining materialized views in a data warehouse, no work exists that explicitly studies approaches to maintain source selection indexes for Linked Data queries.

Live exploration approaches leverage the characteristics of Linked Data, in particular, the existence of data links. That is, to execute a given Linked Data query, live exploration systems perform a URI lookup process during which they incrementally discover further URIs that may be scheduled for lookup (Hartig et al 2009; Hartig 2011b; Ladwig and

Tran 2011; Schmedding 2011; Miranker et al 2012). The data retrieved during this process is used not only to discover more URIs, but it also provides the basis for constructing the query result. Assuming the queries are expressed using SPARQL, live exploration approaches support naturally the aforementioned restrictive semantics for such queries as introduced by Hartig (2012), which also gives a well-defined boundary for the URI lookup process.

In comparison to index-based source selection approaches, live exploration approaches have the interesting characteristic of being able to discover initially unknown data sources *at query execution time*. Additionally, live exploration systems can readily be used without having to wait for the completion of an initial data load phase, index creation, or any other type of preprocessing.

However, given that the general ideas of index-based source selection and of live exploration may be implemented in various ways, respectively, comparing both types of approaches in a fair manner is a nontrivial open problem.

Hybrid source selection approaches combine the two aforementioned ideas. So far, only one hybrid approach has been proposed in the literature. This approach uses an index to create a (ranked) list of URIs to look up; additional URIs discovered during the query execution process are then added into this list (Ladwig and Tran 2010).

Source Ranking Strategies for Linked Data Queries

In addition to selecting URIs that are to be looked up for a given Linked Data query, the selected URIs may be ranked in terms of the order in which these URIs have to be looked up. The goal of such a data source ranking may be to maximize the subset of the query result that can be found in a given amount of time.

Harth et al (2010) and Ladwig and Tran (2010) have proposed data source ranking approaches for index-based source selection. These approaches rely on additional information about the data sources; the assumption is that this information is available in the index.

For selecting URIs using a live exploration approach, Hartig and Özsu (2016) have studied various heuristics to prioritize URI lookups; none of these heuristics uses any a priori information.

Linked Data Query Execution

Existing approaches to execute Linked Data queries as proposed in the literature can be classified into two categories:

Two-phase execution approaches separate the query execution process into two phases: During the first phase, all required data is retrieved from the Web (by looking up URIs, which may be selected using any of the aforementioned source selection approaches). Then, during the second phase, the query result is produced from the data retrieved in the first phase. Harth et al (2010), Umbrich et al (2011), and Paret et al (2011) have adopted such a two-phase execution process to study their index-based source selection approaches. While such a two-

phase process is straightforward to implement, having to wait for the completion of the data retrieval phase may not be practical; it may even exceed the resources of the query execution system.

Integrated execution approaches integrate the retrieval of data with the result construction process. This allows a query engine to return query results incrementally; i.e., before data retrieval has been completed. As for the two-phase execution approaches it is possible to use any type of source selection strategy as a basis for an integrated execution approach. Integrated execution approaches that use some form of live exploration are also referred to as *traversal-based query execution* approaches. A number of such approaches have been proposed where each of them is based on different implementation techniques: The first approach made use of the well-known iterator model (Hartig et al 2009; Hartig 2011b, 2013). In contrast to such a pull-based implementation, Ladwig and Tran (2010, 2011) proposed a push-based implementation using the symmetric hash join operator. Other push-based implementations adapt the Rete match algorithm (Miranker et al 2012) and the idea of an Eddy-style tuple routing operator (Hartig and Özsu 2016).

Query Optimization for Linked Data Queries

Besides the aforementioned source ranking techniques, which can be understood as a form of query optimization, the topic of query optimization for Linked Data queries is largely unexplored. In fact, there does not yet exist any work that investigates cost-based query optimization techniques in this context.

For traversal-based query execution, cost-based query optimization is particularly challenging because information to assess query plans may only be obtained incrementally during query execution. Hence, selecting an initial plan requires heuristics. Hartig (2011b) has proposed such plan selection heuristics for the aforementioned, iterator-based approach to implement traversal-based query execution. For the same implementation approach, Hartig et al (2009) introduce an extension of the iterator model to support a form of adaptive query optimization. This extension avoids a blocking behavior of iterators that may otherwise happen when waiting for the completion of particular URI lookups.

Future Directions for Research

Query optimization. As mentioned in the previous section, query optimization in the context of Linked Data query processing is a largely unexplored area. Possible topics for contributions are: Cost-based query optimization; multi-objective query optimization (i.e., taking into account not only the overall execution time but also, e.g., the time until the first output when returning query results incrementally, or network traffic); techniques for adaptive query processing; optimization for subsequent queries (e.g., collecting statistics as a side-product of traversal-based query executions and leveraging these statistics to optimize future queries).

Experimental comparisons. While a number of approaches for the different

aspects of Linked Data query processing have been proposed, there exists little research that compares these approaches in a comprehensive and fair manner.

Benchmarks. Related to the previous point, there are no well-defined and well-understood benchmarks to test Linked Data query processing systems (neither for specific aspects of Linked Data query processing such as source selection, nor for the process of Linked Data query execution in general).

Techniques for more expressive query languages. All existing techniques for Linked Data query processing focus on queries expressed using the conjunctive fragment of SPARQL. However, the foundations of using more expressive fragments of SPARQL are well-understood (Hartig 2012), and there even exist more fitting languages that have been specifically designed for Linked Data queries (Fionda et al 2015; Hartig and Pérez 2016). It is an open question what are effective techniques for executing queries in these languages.

Hybrid query approaches. While there exists initial work on combining Linked Data query processing with other query paradigms such as query federation or queries over a centralized collection of data (Ladwig and Tran 2011; Umbrich et al 2012; Hartig 2011a; Lynden et al 2013), more research on such combinations is necessary. In particular, contributions can be made in terms of both establishing and understanding the foundations of such hybrid approaches as well as techniques for implementing hybrid approaches.

Cross-References

- Federated RDF query processing
- Graph data models
- Graph query languages
- Graph data management systems
- Framework-based scale-out RDF systems
- Native distributed RDF systems
- Semantic interlinking

References

Acosta M, Vidal M (2015) Networks of linked data eddies: An adaptive web query processing engine for RDF data. In: The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I, pp 111–127, DOI 10.1007/978-3-319-25007-6_7, URL https://doi.org/10.1007/978-3-319-25007-6_7

Avnur R, Hellerstein JM (2000) Eddies: Continuously adaptive query processing. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA., pp 261–272, DOI 10.1145/342009.335420, URL http://doi.acm.org/10.1145/342009.335420

Berners-Lee T (2006) Design Issues: Linked Data. Online at http://www.w3.org/DesignIssues/LinkedData.html

Bizer C, Heath T, Berners-Lee T (2009) Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS) 5(3):1–22

Bouquet P, Ghidini C, Serafini L (2009) Querying the web of data: A formal approach. In: The Semantic Web, Fourth Asian Conference, ASWC 2009, Shanghai, China, December 6-9, 2009. Proceedings, pp 291–305, DOI 10.1007/978-3-642-10871-6_20, URL https://doi.org/10.1007/978-3-642-10871-6_20

Buil-Aranda C, Hogan A, Umbrich J, Vandenbussche P (2013) SPARQL web-querying infrastructure: Ready for action? In: The

Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II, pp 277–293, DOI 10.1007/978-3-642-41338-4_18, URL https://doi.org/10.1007/978-3-642-41338-4_18

Feigenbaum L, Williams GT, Clark KG, Torres E (2013) SPARQL 1.1 Protocol. W3C Recommendation, Online at https://www.w3.org/TR/sparql11-protocol/

Fionda V, Gutierrez C, Pirrò G (2014) The swget portal: Navigating and acting on the web of linked data. J Web Sem 26:29–35, DOI 10.1016/j.websem.2014.04.003, URL https://doi.org/10.1016/j.websem.2014.04.003

Fionda V, Pirrò G, Gutierrez C (2015) Nautilod: A formal language for the web of data graph. TWEB 9(1):5:1–5:43, DOI 10.1145/2697393, URL http://doi.acm.org/10.1145/2697393

Harth A, Hose K, Karnstedt M, Polleres A, Sattler K, Umbrich J (2010) Data summaries for on-demand queries over linked data. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, pp 411–420, DOI 10.1145/1772690.1772733, URL http://doi.acm.org/10.1145/1772690.1772733

Hartig O (2011a) How caching improves efficiency and result completeness for querying linked data. In: WWW2011 Workshop on Linked Data on the Web, Hyderabad, India, March 29, 2011, URL http://ceur-ws.org/Vol-813/ldow2011-paper05.pdf

Hartig O (2011b) Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In: The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I, pp 154–169, DOI 10.1007/978-3-642-21034-1_11, URL https://doi.org/10.1007/978-3-642-21034-1_11

Hartig O (2012) SPARQL for a web of linked data: Semantics and computability. In: The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings, pp 8–23, DOI 10.1007/978-3-642-30284-8_8, URL https://doi.org/10.1007/978-3-642-30284-8_8

Hartig O (2013) SQUIN: a traversal based query execution system for the web of linked data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013, pp 1081–1084, DOI 10.1145/2463676.2465231, URL http://doi.acm.org/10.1145/2463676.2465231

Hartig O, Buil-Aranda C (2016) Bindings-restricted triple pattern fragments. In: On the Move to Meaningful Internet Systems: OTM 2016 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, October 24-28, 2016, Proceedings, pp 762–779, DOI 10.1007/978-3-319-48472-3_48, URL https://doi.org/10.1007/978-3-319-48472-3_48

Hartig O, Özsu MT (2016) Walking without a map: Ranking-based traversal for querying linked data. In: The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I, pp 305–324, DOI 10.1007/978-3-319-46523-4_19, URL https://doi.org/10.1007/978-3-319-46523-4_19

Hartig O, Pérez J (2016) LDQL: A query language for the web of linked data. J Web Sem 41:9–29, DOI 10.1016/j.websem.2016.10.001, URL https://doi.org/10.1016/j.websem.2016.10.001

Hartig O, Bizer C, Freytag JC (2009) Executing SPARQL queries over the web of linked data. In: The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings, pp 293–309, DOI 10.1007/978-3-642-04930-9_19, URL https://doi.org/10.1007/978-3-642-04930-9_19

Ladwig G, Tran T (2010) Linked data query processing strategies. In: The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I, pp 453–469, DOI 10.1007/978-3-642-17746-0_29,

URL https://doi.org/10.1007/978-3-642-17746-0_29

Ladwig G, Tran T (2011) Sihjoin: Querying remote and local linked data. In: The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I, pp 139–153, DOI 10.1007/978-3-642-21034-1_10, URL https://doi.org/10.1007/978-3-642-21034-1_10

Lynden SJ, Kojima I, Matono A, Nakamura A, Yui M (2013) A hybrid approach to linked data query processing with time constraints. In: Proceedings of the WWW2013 Workshop on Linked Data on the Web, Rio de Janeiro, Brazil, 14 May, 2013, URL http://ceur-ws.org/Vol-996/papers/ldow2013-paper-07.pdf

Mika P, Potter T (2012) Metadata Statistics for a Large Web Corpus. In: Proceedings of the 5th Linked Data on the Web Workshop (LDOW)

Miranker DP, Depena RK, Jung H, Sequeda JF, Reyna C (2012) Diamond: A SPARQL Query Engine for Linked Data Based on the Rete Match. In: Proceedings of the Workshop on Artificial Intelligence meets the Web of Data (AImWD)

Paret E, Van Woensel W, Casteleyn S, Signer B, De Troyer O (2011) Efficient querying of distributed RDF sources in mobile settings based on a source index model. In: Proceedings of the 2nd International Conference on Ambient Systems, Networks and Technologies (ANT 2011), the 8th International Conference on Mobile Web Information Systems (MobiWIS-2011), Niagara Falls, Ontario, Canada, September 19-21, 2011, pp 554–561, DOI 10.1016/j.procs.2011.07.072, URL https://doi.org/10.1016/j.procs.2011.07.072

Schaffert S, Bauer C, Kurz T, Dorschel F, Glachs D, Fernandez M (2012) The linked media framework: integrating and interlinking enterprise media content and data. In: I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012, pp 25–32, DOI 10.1145/2362499.2362504, URL http://doi.acm.org/10.1145/2362499.2362504

Schmachtenberg M, Bizer C, Paulheim H (2014) Adoption of the Linked Data Best Practices in Different Topical Domains. In: Proceedings of the 13th International Semantic Web Conference (ISWC)

Schmedding F (2011) Incremental SPARQL evaluation for query answering on linked data. In: Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011, URL http://ceur-ws.org/Vol-782/Schmedding_COLD2011.pdf

Tian Y, Umbrich J, Yu Y (2011) Enhancing source selection for live queries over linked data via query log mining. In: The Semantic Web - Joint International Semantic Technology Conference, JIST 2011, Hangzhou, China, December 4-7, 2011. Proceedings, pp 176–191, DOI 10.1007/978-3-642-29923-0_12, URL https://doi.org/10.1007/978-3-642-29923-0_12

Umbrich J, Hose K, Karnstedt M, Harth A, Polleres A (2011) Comparing data summaries for processing live queries over linked data. World Wide Web 14(5-6):495–544, DOI 10.1007/s11280-010-0107-z, URL https://doi.org/10.1007/s11280-010-0107-z

Umbrich J, Karnstedt M, Hogan A, Parreira JX (2012) Hybrid SPARQL queries: Fresh vs. fast results. In: The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I, pp 608–624, DOI 10.1007/978-3-642-35176-1_38, URL https://doi.org/10.1007/978-3-642-35176-1_38

Van Herwegen J, Verborgh R, Mannens E, Van de Walle R (2015) Query execution optimization for clients of triple pattern fragments. In: The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 - June 4, 2015. Proceedings, pp 302–318, DOI 10.1007/978-3-319-18818-8_19, URL https://doi.org/10.1007/978-3-319-18818-8_19

Vander Sande M, Verborgh R, Van Herwegen J, Mannens E, Van de Walle R (2015) Opportunistic linked data querying through approximate membership meta-

data. In: The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I, pp 92–110, DOI 10.1007/978-3-319-25007-6_6, URL https://doi.org/10.1007/978-3-319-25007-6_6

Verborgh R, Vander Sande M, Hartig O, Van Herwegen J, De Vocht L, De Meester B, Haesendonck G, Colpaert P (2016) Triple pattern fragments: A low-cost knowledge graph interface for the web. J Web Sem 37-38:184–206, DOI 10.1016/j.websem.2016.03.003, URL https://doi.org/10.1016/j.websem.2016.03.003

PDF | Linked Data refers to data published in accordance with a number of principles rooted in web standards. In the past few years we have witnessed a | Find, read and cite all the research you need on ResearchGate.Â Federated Data Management. and Query Optimization for Linked Open Data. In. New Directions in Web Data Management, chapter 5, pages 109â€“137. Springer, 2011. Various approaches toward Linked Data management and querying have been proposed in the literature. To provide an overview, we have categorized them into general approaches, foundations, source selection, source ranking strategies, query execution, and query optimization. General Approaches to Make Use of Linked Data. Centralization. A first option to use Linked Data in an application is to obtain the application-relevant data (e.g., by In computing, linked data (often capitalized as Linked Data) is structured data which is interlinked with other data so it becomes more useful through semantic queries. It builds upon standard Web technologies such as HTTP, RDF and URIs, but rather than using them to serve web pages only for human readers, it extends them to share information in a way that can be read automatically by computers. Part of the vision of linked data is for the Internet to become a global database. In computing linkedï»¿ data describes a method of publishing and linking data coming from heterogeneous data sources that can be interlinked and shared.Â Linked data is a method for publishing structured data using vocabularies like schema.org that can be connected together and interpreted by machines. Using linked data, statements encoded in triples can be spread across different websites. Linked Data Examples. How does Linked Data works? ðŸ“˜ FIWARE 603: Traversing Linked Data Programmatically. www.fiware.org/developers/. MIT License.Â Stock Management Frontend. All the code Node.js Express for the demo can be found within the ngsi-ld folder within the GitHub repository. Stock Management example. The application runs on the following URLs: http://localhost:3000/app/store/urn:ngsi-ld:Building:store001.