Federated RDF Query Processing

Maribel Acosta, Olaf Hartig, and Juan Sequeda

Synonyms

- SPARQL Federation
- Federation of SPARQL Endpoints

Definitions

Federated RDF Query Processing is concerned with querying a federation of RDF data sources where the queries are expressed using a declarative query language (typically, the RDF query language SPARQL) and the data sources are autonomous and heterogeneous. The current literature in this context assumes that the data and the data sources are semantically homogeneous, while heterogeneity occurs at the level of data formats and access protocols.

Maribel Acosta
Karlsruhe Institute of Technology, e-mail: maribel.acosta@kit.edu

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

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

Formalization of SPARQL over Federated RDF Data

In its initial version, the SPARQL query language did not have features to explicitly express queries over a federation of RDF data sources. To support querying such a federation without requiring the usage of specific language features the following assumption has been made throughout the literature: The result of executing any given SPARQL query over the federation should be the same as if the query was executed over the union of all the RDF data available in all the federation members.

To capture this assumption formally (Stolpe 2015), it is necessary to introduce a few basic concepts first: The core component of each SPARQL query is its query pattern. In the simplest case, such a query pattern is a so-called *triple pattern*, that is, a tuple $tp = (s, p, o)$ where s and o may be either a URI, an RDF literal, or a variable, respectively, and p may be either a URI or a variable. Multiple triple patterns can be combined into a set, which is called a *basic graph pattern* (*BGP*) and resembles the notion of a conjunctive query. In addition to such BGPs, several other features are possible in SPARQL query patterns (e.g., UNION, FILTER). For the complete syntax refer to the SPARQL specification (Harris et al 2013).

The expected result of any given SPARQL query is defined based on an evaluation function $[\![\cdot]\!]_G$ that, given an RDF graph G, takes a SPARQL query pattern and returns a set (or multiset) of *solution mappings*, that is, partial functions that associate variables with RDF terms (i.e., URIs, literals, or blank nodes). For instance, for a triple pat-

tern tp, the result $[\![tp]\!]_G$ contains every solution mapping whose domain is the set of all variables in tp and replacing the variables in tp according to the mapping produces an RDF triple that is in G. The evaluation function $[\![\cdot]\!]_G$ is defined recursively over the structure of all query patterns (Pérez et al 2009).

To capture the aforementioned assumption and, thus, define SPARQL over a federation of RDF data sources (rather than over an RDF graph), it is possible to introduce another evaluation function $[\![\cdot]\!]_F$ where F is a given federation. Formally, such a federation may be captured as a tuple $F = (E,d)$ where E is the set of all data sources in the federation and d is a function that associates every source $e \in E$ with the RDF graph that can be accessed via e. Then, the new evaluation function $[\![\cdot]\!]_F$ can be defined in terms of the standard evaluation function $[\![\cdot]\!]_G$ as follows: For every SPARQL query pattern P, $[\![P]\!]_F$ is a set of mappings such that $[\![P]\!]_F = [\![P]\!]_G$ where G is the RDF graph that is the (virtual) union of the graphs of all federation members; i.e., $G = \bigcup_{e \in E} d(e)$.

Types of RDF Data Sources

Note that the aforementioned definition is deliberately generic in what it considers to be possible federation members. Most of the current solutions for federated query processing over RDF data have focused on federations of SPARQL endpoints. Nonetheless, federations of RDF data sources may be composed of other types of sources including RDF documents or Triple Pattern Fragments.

Federation of SPARQL Endpoints. SPARQL endpoints are web services that allow for querying RDF datasets online using the SPARQL protocol (Feigenbaum et al 2013). Such an endpoint has a URI as its address and is, in theory, capable of evaluating any given SPARQL query over its internal RDF dataset. Therefore, SPARQL endpoints are considered RDF querying interfaces with a high expressive power. The concept of querying SPARQL endpoints has even been integrated into the latest version of the SPARQL language. That is, SPARQL 1.1 comes with an extension (Prud'hommeaux and Buil-Aranda 2013) that introduces the notion of a SERVICE clause that can be used as part of a SPARQL query pattern. This new clause has the form SERVICE $c\,P$ where c is a URI and P is a (sub-)pattern (note that, by the SPARQL 1.1 standard, c may even be a variable, but the existing formal definition of this case (Aranda et al 2013) is not part of the standard). Then, assuming URI c is an address of a SPARQL endpoint, the SERVICE clause means that the evaluation of P has to be delegated to this endpoint. The resulting solution mappings returned from the endpoint have to be combined with the result of the rest of the query pattern in which the SERVICE clause is contained.

Federation of RDF Documents. This type of federation composes a Web of (Linked) Data in which the federation members are RDF documents that can be retrieved by URI dereferencing. In URI dereferencing, clients use URIs (typically found in the data) to perform HTTP requests which result in retrieving documents that contain a set of RDF triples. The process of evaluating queries over a federation of RDF documents is referred to as *Linked*

Data query execution (Hartig 2013) and URI dereferencing is a core task of this process. Hartig (2012) has shown that by using this process, producing the complete query result as defined by the aforementioned evaluation function $[\![\cdot]\!]_F$ cannot be guaranteed, and that there exist more restrictive approaches to define an evaluation function that can be computed completely over a federation of interlinked RDF documents.

Federation of Triple Pattern Fragments. Triple Pattern Fragments (TPFs) is an alternative interface for providing access to RDF data (Verborgh et al 2016). TPF sources are able to evaluate SPARQL triple patterns over their internal RDF dataset. The result of such an evaluation is a sequence of RDF triples that match the given triple pattern; this is called a fragment. Given that some fragments may contain a large number of triples, fragments are partitioned into fragment pages that contain a fixed maximum number of triples. Then, to retrieve an entire fragment, clients must traverse the TPF pages. The expressive power of TPFs is higher than URI dereferencing, but notably lower than SPARQL endpoints. To execute an arbitrary SPARQL query using TPFs, the query engine has to decompose the query into a series of TPF requests.

Query Processing

Conceptually, the processing of queries in a federated setting can be separated into three tasks: (i) query decomposition and source selection, (ii) query optimization, and (iii) query execution. This section provides an overview of existing approaches to tackle these tasks. To this end, the tasks are considered separately, one after another, which is also the most common approach to perform them in many RDF federation engines. However, some engines intertwine the tasks (at least partially) to account for changing data sources and run-time conditions.

Query Decomposition and Source Selection

To execute a query over a federation, the query first has to be decomposed into subqueries that are associated with data sources (federation members) selected for executing the subqueries. Essentially, the result of this step resembles a rewriting of the query into a SPARQL query with SERVICE clauses, and the objective is to achieve a complete query result with a minimum number of SERVICE clauses. An important notion in this context is the relevance of data sources for triple patterns; that is, a data source d is *relevant* for a triple pattern tp if the evaluation of tp over the data of d is nonempty; i.e., $[\![tp]\!]_{ep(d)} \neq \emptyset$.

An initial, not necessarily optimal approach to decompose a given SPARQL query is to treat each triple pattern in the query as a separate subquery and to associate each such subquery with all the data sources that are relevant for the respective triple pattern. A common improvement of such an initial decomposition is to merge subqueries that are associated with the same single data source; such merged subqueries are called *exclusive groups* (Schwarte et al 2011).

Note that the results of the different subqueries have to be combined to produce the result of the given SPARQL query. It is possible, however, that

for any of the (relevant) data sources associated with a subquery, the subquery result obtained from that source may give an empty result when combined with the results of some of the other subqueries. Hence, that subquery result does not contribute to the overall query result produced in the end. Based on this observation it is possible to prune some of the data sources associated with subqueries if there is evidence that the subquery results from these data sources can be safely ignored without affecting the completeness of the overall query result. Source selection approaches that apply such a pruning are called *join-aware* (Saleem and Ngomo 2014).

Table 1 lists the approaches for query decomposition / source selection as proposed in the SPARQL federation literature. The main difference between these approaches (in addition to join awareness) is in the information they require about the federation members and the point at which they obtain this information. There exist approaches that rely solely on some type of *pre-populated data catalog* with (approach-specific) metadata about the federation members. Other approaches do not assume such a catalog and, instead, identify relevant federation members by querying all federation members *during the source selection* process. A third type of approaches are *hybrids* that employ a pre-populated data catalog in combination with runtime requests to the federation members. The approaches that employ a catalog (with or without additional runtime requests) can be distinguished further based on what type of metadata and statistics they capture in their catalog. Both Görlitz and Staab (2011a, Sec.5.3.2) and Oguz et al (2015, Sec.3.1–3.2) provide a more detailed overview of existing approaches.

Table 1 Properties of query decomposition / source selection approaches proposed in the literature, where DC approaches rely solely on a pre-populated data catalog, RT approaches rely only on runtime requests, and HY are hybrids.

Approach	join-aware	class
Federation engine		
DARQ (Quilitz and Leser 2008)	✓	DC
ADERIS (Lynden et al 2010, 2011)	✗	DC
ANAPSID (Acosta et al 2011)	✓	HY
SPLENDID (Görlitz and Staab 2011b)	✗	HY
FedX (Schwarte et al 2011)	✗	RT
Prasser et al (2012)	✓	DC
WoDQA (Akar et al 2012)	✓	HY
LHD (Wang et al 2013)	✗	HY
SemaGrow (Charalambidis et al 2015)	✗	HY
Lusail (Mansour et al 2017)	✓	RT
Source selection extension		
DAW (Saleem et al 2013)	✗	HY
HiBISCuS (Saleem and Ngomo 2014)	✓	HY
Fed-DSATUR (Vidal et al 2016)	✓	RT

Query Optimization

The optimizer devises a plan that encodes how the execution of the query is carried out. The objective of the optimizer is to devise an execution plan that minimizes the cost of processing the query decompositions. In federated scenarios, both the number of intermediate results as well as the communication costs impact on the query performance.

Cost Estimation for Plans. To estimate the cost of plans in federated SPARQL query processing, Görlitz and Staab (2011b) propose a cost model that accounts for communication costs and

estimates the cardinality of SPARQL expressions. Regarding the communication costs, this model assumes that all the responses from the sources require the same number of packages to be transmitted over the network, and that the solution mappings have the same average size. Furthermore, this cost model relies on exact cardinalities for triple patterns with bound predicates and estimations for other variations of triple patterns, graph patterns, and joins. Nonetheless, in federated query processing where sources are autonomous, it is not always possible to collect sufficient statistics from the data sources to estimate cardinalities accurately. Therefore, some SPARQL federation engines (Acosta et al 2011; Schwarte et al 2011) implement heuristic-based optimization techniques that estimate the cost of subqueries based on the number of triple patterns and the number of variables in each triple pattern.

Plan Enumeration. To explore the space of query plans, optimizers of current SPARQL federation engines implement different search strategies that devise different tree plan shapes: (left-) deep plans and bushy plans. In contrast to deep plans, bushy trees are able to reduce the size of intermediate results in SPARQL queries (Vidal et al 2010) and enable parallel execution of independent sub-plans. One common strategy used in traditional query processing is Dynamic Programming, which is able to enumerate all join plans while pruning inferior plans based on the estimated cost. This strategy is implemented by SPLENDID (Görlitz and Staab 2011b) and LHD (Wang et al 2014) to devise bushy tree plans. The main advantage of Dynamic Programming is that it produces the best possible plans under the assumption that that cost model is accurate. However, the time and space complexity of Dynamic Programming is exponential, and thus this strategy does not scale for queries with a large number of sub-queries. To overcome this limitation, the federated engines FedX (Schwarte et al 2011) and ANAPSID (Acosta et al 2011) implement heuristics to devise reasonable effective plans quickly. The heuristics implemented by FedX enumerate left-deep plans, while ANAPSID heuristics are able to devise bushy tree plans.

Further Optimization Techniques. In traditional query processing, the optimizer considers pushing down filters in the query plan which, in turn, allows for reducing the amount of intermediate results produced during query execution. Following this technique, ANAPSID (Acosta et al 2011) implements heuristics to: i) push down filter expressions in the query plan, and ii) decide whether the filter is evaluated at the query engine or at the endpoint. In case that the filter expression contains variables that are resolved by the same source, ANAPSID ships the evaluation of the filter to the endpoint thus reducing the amount of data transmitted over the network. Other optimization techniques split composed filter expressions into sub-expressions such that each sub-expressions can be pushed down in the query plan (Görlitz and Staab 2011b).

Query Execution Techniques

During query execution, the engines evaluate the plan devised by the optimizer. In this task, the engine contacts the members of the federation identified

as relevant and combines the retrieved data to produce the final query result. In federated SPARQL query processing, physical operators and adaptive query processing techniques have been proposed to efficiently combine the intermediate results.

Physical Operators. In terms of Nested Loops Join, Görlitz and Staab (2011a) and Schwarte et al (2011) have proposed join operators that reduce the number of requests submitted to the sources by grouping several bindings into a single request. The *distributed semi-join* (Görlitz and Staab 2011a) buffers the solution mappings in FILTER expressions using the logical connectors $\wedge$ (when mappings contain several variables) and $\vee$ (for grouping several mappings). In contrast, the *bound join* (Schwarte et al 2011) groups a set of solution mappings in a single subquery using SPARQL UNION constructs. Furthermore, ANAPSID (Acosta et al 2011) provides non-blocking join operators *agjoin* and *adjoin* that efficiently combine solution mappings able to produce results as soon as the data arrives from the endpoints.

Adaptive Query Processing. Most of the current federated SPARQL query engines follow the optimize-then-execute paradigm. In this paradigm the plan devised by the optimizer is fixed. Nonetheless, the optimize-then-execute paradigm is insufficient in federated scenarios in which the optimizer is not able to devise optimal plans due to mis-estimated statistics or unpredictable costs resulting from network delays or source availability. To overcome the limitations of the optimize-then-execute paradigm, some SPARQL federation engines implement adaptive query processing techniques (Deshpande et al 2007) to adjust their execution schedulers according to the monitored runtime conditions. ANAPSID (Acosta et al 2011) is an adaptive engine that flushes solution mappings that have matched a resource to secondary memory, in cases when the main memory becomes full. Solution mappings in secondary memory are re-considered to produce new solutions when the endpoints become blocked. The query execution techniques implemented by ANAPSID are tailored to cope with network delays and to handle a large number of intermediate results produced by arbitrary data distributions. Another federated engine that implements adaptivity during query execution is ADERIS (Lynden et al 2010, 2011). ADERIS is able to re-invoke the query optimizer at execution time after all the data is retrieved from the endpoints. The optimizer then changes the join ordering to devise a new efficient query plan given the current runtime conditions.

Open Research Challenges

The open research challenges can be organized in two parts. The first set of challenges are in the context of the Semantic Web. The second set of challenges extend known issues in federated query processing on relational databases.

Challenges of Using the Semantic Web as a Federation

Reasoning: RDF and SPARQL are part of a larger set of Semantic Web

technologies standardized by the W3C. An important standard is the Web Ontology Language (OWL), which is a knowledge representation and reasoning language based on Description Logics. RDF systems may implement reasoners which enables inferring new data based on the input data and ontologies. A federated query processing system on RDF data coupled with OWL ontologies must be extended to support the following: i) understanding different entailment regimes (Glimm and Ogbuji 2013) that a federation member is able to support (e.g., RDF entailment, RDFS entailment, OWL2 direct semantics entailment) and ii) reasoning with alignments between different OWL ontologies used across different federation members. Early approaches that have taken first steps to integrate ontology alignments into federated query processing are ALOQUS (Joshi et al 2012) and LHD-d (Wang et al 2014).

Unboundedness: Existing approaches to federated RDF query processing assume that the complete federation is known beforehand. However, in the context of the Semantic Web, this may not be the case. An initial set of data sources (federation members) may be known, but it is possible that all the sources are not known initially. Hence, the total number of sources that can contribute to the answer of a given query may be unbound. This unboundedness has implications in query processing, namely in the query decomposition and source selection step.

Completeness: Recall that the result of executing any given SPARQL query over the federation should be the same as if the query was executed over the union of all the RDF data available in all the federation members. Therefore, completeness of a SPARQL query over the federation depends on knowing all the federation members. Per the unboundedness challenge (see above), if the sources are not known, then this notion of completeness should be relaxed. The implication of this challenge is to understand a tradeoff between soundness and completeness versus precision and recall of answering a query in a federation.

Dynamicity: The data of some federation members may change frequently (e.g., streams, social media). It has to be studied how such dynamicity affects the results of queries over a federation and how the dynamicity can be taken into account during federated query processing.

Challenges of Extending Federated Query Processing

Access Control: Within federated database systems, access control is a mechanism to check whether a query issued by a user is allowed or not. Given that each federation member is autonomous and has its own access control, conflicts may arise when data sources are combined in a federation. This challenge is further increased with the presence of ontologies and reasoning capabilities at the federation members. The inferred data needs to be checked with respect to the access control.

Heterogeneity: Although all the federation members are assumed to use a common data model (RDF), the members can be heterogeneous at different levels: i) members may be using different ontologies to model data, ii) different ver-

sions of SPARQL and different query capabilities (e.g., SPARQL endpoints vs. TPF) may have an impact on the type of queries that members are able to answer, iii) depending on which entailment regimes are supported, a member can infer different types of answers.

Source Selection: The challenge of decomposing a query into subqueries that are associated with data sources is increased when the following additional constraints are considered. Given that data sources are autonomous, they can: i) have different timeouts, ii) put a limit on the number of results a query can return, iii) support different version of SPARQL or even support subsets of SPARQL only, iv) be of different types, such as SPARQL endpoints, RDF documents, or TPF interfaces, v) support different entailment regimes. Therefore, a federated query processing system should take into account these constraints during the source selection.

Query Results: An important challenge in federated query processing is to associate provenance with the query results in order to understand where an answer is coming from. When sources that support reasoning such that they are able to infer new data, this adds to further explanations in the provenance of the answers.

Cross-References

- Graph data models
- Graph query languages
- Linked Data management

References

Acosta M, Vidal M, Lampo T, Castillo J, Ruckhaus E (2011) ANAPSID: an adaptive query processing engine for SPARQL endpoints. In: The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I, pp 18–34, DOI 10.1007/978-3-642-25073-6_2, URL https://doi.org/10.1007/978-3-642-25073-6_2

Akar Z, Halaç TG, Ekinci EE, Dikenelli O (2012) Querying the web of interlinked datasets using VOID descriptions. In: WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012, URL http://ceur-ws.org/Vol-937/ldow2012-paper-06.pdf

Aranda CB, Arenas M, Corcho Ó, Polleres A (2013) Federating queries in SPARQL 1.1: Syntax, semantics and evaluation. J Web Sem 18(1):1–17, DOI 10.1016/j.websem.2012.10.001, URL https://doi.org/10.1016/j.websem.2012.10.001

Charalambidis A, Troumpoukis A, Konstantopoulos S (2015) Semagrow: optimizing federated SPARQL queries. In: Proceedings of the 11th International Conference on Semantic Systems, SEMANTICS 2015, Vienna, Austria, September 15-17, 2015, pp 121–128, DOI 10.1145/2814864.2814886, URL http://doi.acm.org/10.1145/2814864.2814886

Deshpande A, Ives ZG, Raman V (2007) Adaptive query processing. Foundations and Trends in Databases 1(1):1–140, DOI 10.1561/1900000001, URL https://doi.org/10.1561/1900000001

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

Glimm B, Ogbuji C (2013) SPARQL 1.1 Entailment Regimes. W3C Recommendation, Online at https://www.w3.org/TR/sparql11-entailment/

Görlitz O, Staab S (2011a) Federated data management and query optimization for linked open data. In: New Directions in Web Data Management 1, Springer, pp 109–137, DOI 10.1007/978-3-642-17551-0_5,

URL https://doi.org/10.1007/978-3-642-17551-0_5

Görlitz O, Staab S (2011b) SPLENDID: SPARQL endpoint federation exploiting VOID descriptions. In: Proceedings of the Second International Workshop on Consuming Linked Data (COLD2011), Bonn, Germany, October 23, 2011, URL http://ceur-ws.org/Vol-782/GoerlitzAndStaab_COLD2011.pdf

Harris S, Seaborne A, Prud'hommeaux E (2013) SPARQL 1.1 Query Language. W3C Recommendation, Online at http://www.w3.org/TR/sparql11-query/

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) An overview on execution strategies for linked data queries. Datenbank-Spektrum 13(2):89–99, DOI 10.1007/s13222-013-0122-1, URL https://doi.org/10.1007/s13222-013-0122-1

Joshi AK, Jain P, Hitzler P, Yeh PZ, Verma K, Sheth AP, Damova M (2012) Alignment-based querying of linked open data. In: On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part II, pp 807–824, DOI 10.1007/978-3-642-33615-7_25, URL https://doi.org/10.1007/978-3-642-33615-7_25

Lynden SJ, Kojima I, Matono A, Tanimura Y (2010) Adaptive integration of distributed semantic web data. In: Databases in Networked Information Systems, 6th International Workshop, DNIS 2010, Aizu-Wakamatsu, Japan, March 29-31, 2010. Proceedings, pp 174–193, DOI 10.1007/978-3-642-12038-1_12, URL https://doi.org/10.1007/978-3-642-12038-1_12

Lynden SJ, Kojima I, Matono A, Tanimura Y (2011) ADERIS: an adaptive query processor for joining federated SPARQL endpoints. In: On the Move to Meaningful Internet Systems: OTM 2011 - Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Hersonissos, Crete, Greece, October 17-21, 2011, Proceedings, Part II, pp 808–817, DOI 10.1007/978-3-642-25106-1_28, URL https://doi.org/10.1007/978-3-642-25106-1_28

Mansour E, Abdelaziz I, Ouzzani M, Aboulnaga A, Kalnis P (2017) A demonstration of lusail: Querying linked data at scale. In: Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017, pp 1603–1606, DOI 10.1145/3035918.3058731, URL http://doi.acm.org/10.1145/3035918.3058731

Oguz D, Ergenc B, Yin S, Dikenelli O, Hameurlain A (2015) Federated query processing on linked data: a qualitative survey and open challenges. Knowledge Eng Review 30(5):545–563, DOI 10.1017/S0269888915000107, URL https://doi.org/10.1017/S0269888915000107

Pérez J, Arenas M, Gutierrez C (2009) Semantics and complexity of SPARQL. ACM Trans Database Syst 34(3):16:1–16:45, DOI 10.1145/1567274.1567278, URL http://doi.acm.org/10.1145/1567274.1567278

Prasser F, Kemper A, Kuhn KA (2012) Efficient distributed query processing for autonomous RDF databases. In: 15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings, pp 372–383, DOI 10.1145/2247596.2247640, URL http://doi.acm.org/10.1145/2247596.2247640

Prud'hommeaux E, Buil-Aranda C (2013) SPARQL 1.1 Federated Query. W3C Recommendation, Online at https://www.w3.org/TR/sparql11-federated-query/

Quilitz B, Leser U (2008) Querying distributed RDF data sources with SPARQL. In: The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings, pp 524–538, DOI 10.1007/978-3-540-68234-9_39,

URL https://doi.org/10.1007/978-3-540-68234-9_39

Saleem M, Ngomo AN (2014) Hibiscus: Hypergraph-based source selection for SPARQL endpoint federation. In: The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings, pp 176–191, DOI 10.1007/978-3-319-07443-6_13, URL https://doi.org/10.1007/978-3-319-07443-6_13

Saleem M, Ngomo AN, Parreira JX, Deus HF, Hauswirth M (2013) DAW: duplicate-aware federated query processing over the web of data. In: The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I, pp 574–590, DOI 10.1007/978-3-642-41335-3_36, URL https://doi.org/10.1007/978-3-642-41335-3_36

Schwarte A, Haase P, Hose K, Schenkel R, Schmidt M (2011) Fedx: Optimization techniques for federated query processing on linked data. In: The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I, pp 601–616, DOI 10.1007/978-3-642-25073-6_38, URL https://doi.org/10.1007/978-3-642-25073-6_38

Stolpe A (2015) A logical characterisation of SPARQL federation. Semantic Web 6(6):565–584, DOI 10.3233/SW-140160, URL https://doi.org/10.3233/SW-140160

Verborgh R, Sande MV, Hartig O, Herwegen JV, Vocht LD, Meester BD, 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

Vidal M, Ruckhaus E, Lampo T, Martínez A, Sierra J, Polleres A (2010) Efficiently joining group patterns in SPARQL queries. In: The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part I, pp 228–242, DOI 10.1007/978-3-642-13486-9_16, URL https://doi.org/10.1007/978-3-642-13486-9_16

Vidal M, Castillo S, Acosta M, Montoya G, Palma G (2016) On the selection of SPARQL endpoints to efficiently execute federated SPARQL queries. Trans Large-Scale Data- and Knowledge-Centered Systems 25:109–149, DOI 10.1007/978-3-662-49534-6_4, URL https://doi.org/10.1007/978-3-662-49534-6_4

Wang X, Tiropanis T, Davis HC (2013) LHD: optimising linked data query processing using parallelisation. 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-06.pdf

Wang X, Tiropanis T, Davis HC (2014) Optimising linked data queries in the presence of co-reference. In: The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings, pp 442–456, DOI 10.1007/978-3-319-07443-6_30, URL https://doi.org/10.1007/978-3-319-07443-6_30