Fedra: Query Processing for SPARQL Federations with Divergence

Gabriela Montoya, Hala Skaf-Molli, Pascal Molli, Maria-Esther Vidal

Fedra: Query Processing for SPARQL Federations with Divergence

Gabriela Montoya[12], Hala Skaf-Molli[1], and Pascal Molli[1]
Maria-Esther Vidal[3]

[1] LINA– Nantes University, France
{gabriela.montoya,hala.skaf,pascal.molli}@univ-nantes.fr
[2] CNRS Unit UMR6241, France
[3] Universidad Simón Bolívar, Venezuela
mvidal@ldc.usb.ve

Abstract. Data replication and deployment of local SPARQL endpoints improve scalability and availability of public SPARQL endpoints, making the consumption of Linked Data a reality. This solution requires synchronization and specific query processing strategies to take advantage of replication. However, existing replication aware techniques in federations of SPARQL endpoints do not consider data dynamicity. We propose FEDRA, an approach for querying federations of endpoints that benefits from replication. Participants in FEDRA federations can copy fragments of data from several datasets, and describe them using provenance and views. These descriptions enable FEDRA to reduce the number of selected endpoints while satisfying user divergence requirements. Experiments on real-world datasets suggest savings of up to three orders of magnitude. Keywords: Source Selection, SPARQL Endpoints, Data Replication.

1 Introduction

Linked Open Data makes millions of triples available for processing SPARQL queries through federation of SPARQL endpoints. However, infrastructure of SPARQL endpoints have intrinsics limitations in terms of scalability and availability. According to [2], on 427 public endpoints of federation only one third have an availability rate above 99% .

Traditional approaches for improving data availability involves fragmentation [14] and replications [12]. According to incoming queries, data publishers compute fragments, replicate and smartly place them on their managed clusters to improve data availability. Unfortunately, for these approaches, scalability and availability only relies on resources of data publishers. We advocate for a vision where data scalability and availability costs should be supported by the whole federation of linked data relying on existing standards.

Fragmentation and replication can occur opportunistically among federation members. One participant can materialize a fragment of another one for its own purpose and make it available for the whole federation. A federated query engine that take advantage of such opportunistic fragmentation and replication can improve general scalability and availability of linked open data.

Listing 1: DBpedia Query Q

```
SELECT DISTINCT *
WHERE {
  ?city <http://dbpedia.org/
      ontology/country> ?c .
  ?city <http://dbpedia.org/
      ontology/department> ?d .
  ?city <http://dbpedia.org/
      ontology/postalCode> ?pc
}
```

Unfortunately, existing federated query engines do not exploit such opportunities. To illustrate, consider a DBpedia dataset d_1, and a federation that only accesses a public SPARQL endpoint of DBpedia. Using FedX [13], the execution of the three-triple pattern query in Listing1 is quite simple because the query can be exclusively executed in one endpoint. If the same query were executed in a federation with a mirror of DBpedia (d_2), each triple pattern should be executed against both endpoints, thus query performance deteriorates. Nevertheless, using the knowledge that d_2 is just a mirror of d_1 leads to a simple query strategy that surely will save execution time. On the other hand, using mirrors raise the issue of consistency of different fragment and dynamicity of data. Performing queries in presence of divergent data leads to stale answers [7].

Recently, Saleem et al. [12] propose DAW, a framework able to detect data duplication between datasets and reduce the number of selected sources. DAW uses data present in the federation at a given time to build indexes for each dataset and is able to detect quickly data duplication between two datasets. In the previous example, FedX with DAW would contact only one DBpedia endpoint. However, DAW does not consider fragments nor source dynamicity, i.e., DAW summaries has to be recomputed after each change.

In this paper we describe FEDRA, a source selection strategy that can be used in conjunction with existing federated query engine in order to improve efficiency of queries and data availability. FEDRA considers a set of endpoints S exposing fragments. A fragment is defined as a SPARQL construct query, the original data source, a value expressing freshness of the fragment according to original source. Given a query Q, and divergence threshold, FEDRA computes the minimum set of endpoints to resolve the query. Divergence threshold allow to control how much stale values can be retrieved in query results.

On experimental setup based on DBpedia, we observe that FEDRA takes advantage of opportunistic replication to significantly reduce execution time of federated queries while bounding stale values even with simple divergence metrics.

The main contributions of this paper are: *i*) Endpoint descriptions in terms of SPARQL views, containment relationships between views, data provenance, and timestamps; *ii*) FEDRA source selection algorithm that reduces the number of selected sources while selected sources divergence is controlled; and *iii*) an ex-

perimental study that reveals the benefits of both exploiting knowledge encoded in endpoint descriptions and controlling divergence to avoid obsolete results.

The paper is organized as follows: Section 2 presents FEDRA and the source selection algorithm. Section 3 reports our experimental study. Section 4 summarizes related work. Finally, conclusions and future work are outlined in Section 5.

2 Fedra Source Selection for SPARQL Federations with Replication

The lack of reliability of SPARQL endpoints is currently forcing intensive linked data consumers, or linked data application developers to rely on dataset dumps and local re-installation to fulfill their own needs. Some data producers such as DBpedia or MusicBrainz are also providing live update feeds to keep local mirrors up-to-date and leverage the load on their own SPARQ endpoints.
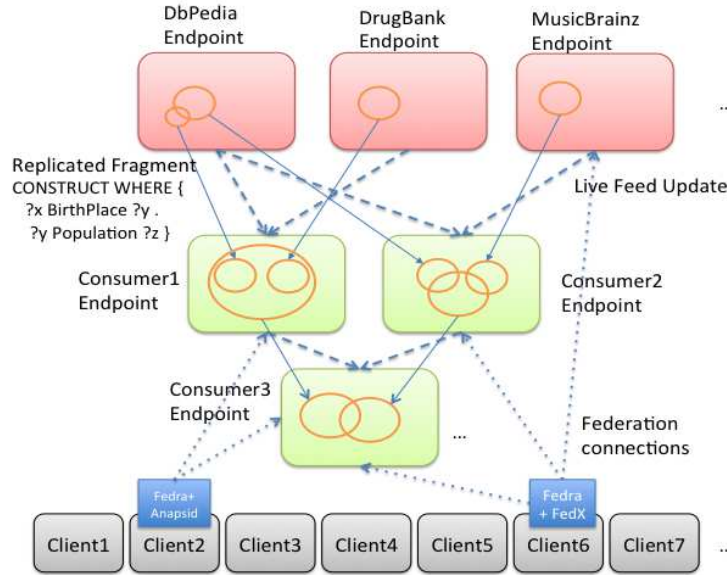

Fig. 1: Fedra general approach

In FEDRA, we generalize this approach by defining and replicating fragments in an opportunistic way. For example, in figure 1, one opportunistic federation may be composed by public endpoints for datasets Musicbrainz[4], DBpedia[5],

---

[4] http://dbtune.org/musicbrainz/sparql

[5] http://dbpedia.org/sparql

and Drugbank[6], and 3 intensive data consumers. Such deployment allows data producers to share the load on their own endpoints and data consumers to access to the knowledge of federated queries issued by the community concerning these fragments.

Fragments are defined by regular CONSTRUCT SPARQL queries that can be disjoint, overlapping, or contained with other fragments. Live Update Feeds as in DBpedia Live, SPARQL Push publish-subscribe [11], or just re-executing fragments queries allow to maintain fragments up-to-date but at unpredictable rates.

Finally, clients declare original and consumers endpoints in their federated queries. FEDRA compute the minimal set of endpoints where a given query has to be executed taking into account the freshness on fragments and delegate the execution of the query to an existing federated query engine such as FedX or Anapsid [1].

FEDRA will take advantage of locality properties introduced by opportunistic presence of fragments on the same endpoints to solve the source selection problem and consequently improve the query execution time, and data availability. Divergence metrics allow to control synchronization issues between fragments. Selected aged fragments can introduce stale values in the query result. In FE-DRA, clients can express the divergence tolerance between fragments to impact number of expected stale values in query results.

Before presenting the problem statement for the source selection problem FEDRA solves, we introduce some definitions:

Definition 1 (Data fragment). *A data fragment is a subset of the triples that constitute a dataset. It can be described using a query that gives the pattern satisfied by the fragment members, or using the triples contained in it at a given time.*

The first description is given in terms of the schema the fragment triples are compliant to, and the second one is instance dependent. A recent study [9] has shown that the schema of datasets is generally static and have a lot less changes instances, hence using queries to describe fragments provide a more reliable description.

Definition 2 (Replica). *Replicas are fragments that do not represent new data, but that was taken from an existing dataset.*

Definition 3 (Replica description). *A replica may be uniquely described by the source from where the replica was made, the query that when performed in that source returns the replica fragment, the date when the replica was made, and the SPARQL endpoint that provides access to it.*

Definition 4 (Containment relation among fragments). *A fragment F1 is contained in a fragment F2, if all the possible triples that may belong to F1 may also belong to F2.*

---

[6] http://drugbank.bio2rdf.org/SPARQL

Definition 5 (Divergence). *Divergence is the different between a fragment and its replicas.*
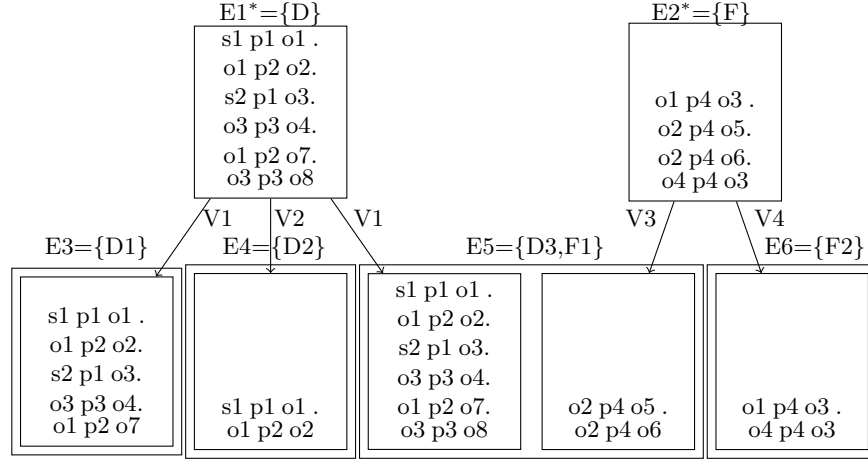
This difference may be measured in different ways, some examples are elapsed time and in number of performed operations.

Definition 6 (Fragment relevance). *A data fragment f is relevant for answering a query Q if the query that defines f and Q have some compatible triple patterns, and it is denoted as $relevant(f,Q)$. Two triple patterns $t1$ and $t2$ are compatible if there exists at least one possible binding of values to variables in $t1$, $t2$ such that the resulting triples after applying the binding are equal.*

Source Selection Problem (SSP). Given a set of SPARQL endpoints E, a subset P of E that correspond to the public endpoints, the set of fragments contained in each endpoint as a function $fragments : Endpoint \rightarrow set\,of\,Fragment$, a containment relation among endpoints for each $frag \in fragments(e_i) \wedge frag \in fragments(e_j)$, $e_i \subseteq_{frag} e_j$, the age or divergence of each fragment $frag$ replicated in e_i from e_j, $e_i\,div_{frag}\,e_j$, a query Q and an age limit or divergence tolerance dt, find a map D, such that for each triple pattern t in Q $D(t) \subseteq (E)$ and:

- answer(Q, D) $=_{dt}$ answer(Q, F), where F is a map such that for each triple pattern t in Q has value E.
- $(\forall t : t \in triplePatterns(Q) : (\forall e, f : e \in E{-}D(t), f \in fragments(e),$ $relevant(f,Q) : (\exists e' : e' \in D(t) : e \subseteq_f e')))$
- $D(t)$ contains as few public endpoints as possible.
- $\bigcup_{t\,in\,triplePatterns(Q)} D(t)$ is a minimal set that satisfies these conditions

Figure (2a) illustrates an opportunistic federation with fragments from datasets E and F. D fragments ($D1$, $D2$, and $D3$) are defined by views $V1$ and $V2$, and F fragments ($F1$ and $F2$) are defined by views $V3$ and $V4$. $V1$ defines a fragment that consists of the whole dataset, while $V2$, $V3$, and $V4$ define fragments that consist of a subset of the corresponding dataset. There is an explicit containment relationship among these fragments, for example fragment $D1$ data in endpoint $E3$ comes from endpoint fragment D in endpoint $E1$, *i.e.*, $D \sqsubseteq_{V1} D1$. There is also an implicit containment relationship, for example fragment D2 data is contained in fragment D1 data. The public SPARQL endpoint $E1$ provides access to D, while local SPARQL endpoints $E3$, $E4$, and $E5$ access $D1$, $D2$, and $D3$, respectively. In addition, a replica endpoint may provide access to fragments from different datasets like $E5$ that combines data from $D3$ and $F1$. At the moment T+2, fragments $D3$ and $F1$ are up to date because their last included update was at time T+2, while fragments $D1$ and $F2$ have age 1, because their last included update was at time T+1, and fragment $D2$ has age 2 because its last included update was at time T. All the SPARQL endpoints can contribute during federated query processing. Therefore, workload on public endpoints can be reduced, and replica SPARQL endpoints can replace the public ones whenever they are not available.

E1*={D}

s1 p1 o1 .
o1 p2 o2.
s2 p1 o3.
o3 p3 o4.
o1 p2 o7.
o3 p3 o8

E2*={F}

o1 p4 o3 .
o2 p4 o5.
o2 p4 o6.
o4 p4 o3

V1   V2   V1   V3   V4

E3={D1}   E4={D2}   E5={D3,F1}   E6={F2}

E3={D1}:
s1 p1 o1 .
o1 p2 o2.
s2 p1 o3.
o3 p3 o4.
o1 p2 o7

E4={D2}:
s1 p1 o1 .
o1 p2 o2

E5={D3,F1}:
s1 p1 o1 .
o1 p2 o2.
s2 p1 o3.
o3 p3 o4.
o1 p2 o7.
o3 p3 o8

(E5 second):
o2 p4 o5 .
o2 p4 o6

E6={F2}:
o1 p4 o3 .
o4 p4 o3

(a) Fragments of Datasets D and F

Date	Type	Operation
T	I	s1 p1 o1
	I	o1 p2 o2
	I	s2 p1 o3
	I	o3 p3 o4
T+1	I	o1 p2 o7
T+2	I	o3 p3 o8

(b) D last operations

View	Definition
V1	CONSTRUCT WHERE { ?x p1 ?y . ?y ?p ?z }
V2	CONSTRUCT WHERE { ?x p1 ?y . ?y p2 ?z }
V3	CONSTRUCT WHERE { o2 p4 ?x }
V4	CONSTRUCT WHERE { ?x p4 o3 }

(c) Views definitions

Endpoint[Fragment]	E3[D1]	E4[D2]	E5[D3]	E5[F1]	E6[F2]
Age	1	2	0	0	1

(d) Fragments age

Fig. 2: Example of Opportunistic Federation, for each endpoint on the top it is indicated the contained fragments, endpoints marked with * are public endpoints. Updates for the last three time unit, fragments view definitions and fragment age.

<div align="center">Listing 2: Query Q1</div>

```
SELECT DISTINCT ?s ?o ?r
WHERE {
    ?s p1 ?o .
    ?o p4 ?r
}
```

Consider the query given in Listing 2 with triple patterns k1=$(?s\,p1\,?o)$ and k2=$(?o\,p4\,?r)$, and an age limit of 1. A solution to the SSP is { (k1, { E5 }), (k2, { E5, E6})}. For k1 endpoints E1, E3, E4, and E5 are relevant, and from these E1, E3 and E5 are equivalent, i.e., there is a containment in both senses, and E4 is strictly contained in them. Then, choosing E5 for k1 satisfies that all the other endpoints that provide relevant data for k1 are "covered" by E5. For k2 endpoints E2, E5 and E6 are relevant, and from these E2 provides as much information as both E5 and E6 combined, but E2 is a public endpoint, then E5 and E6 should be selected. Notice that the choice of E5 for k1 is necessary to reduce the total number of selected sources, since it is also relevant for k2. Selected endpoints, $E5$ and $E6$, satisfy the age limit of 1.

2.1 Source Selection Algorithm

Algorithm 1 Source Selection algorithm

Require: Q: SPARQL Query
Require: E: set of Endpoints
Require: $fragments\ :\ Endpoint \rightarrow set\ of\ fragment$ ▷ fragments offered by each endpoint
Require: $P\ :\ set\ of\ Endpoint$ ▷ endpoints that should not be selected
Require: $\subseteq_{frag}:\ Endpoint \times Endpoint$ ▷ containment relation given by fragments
Require: $div_{frag}\ :\ Endpoint \times Endpoint \rightarrow Integer$ ▷ Fragment age
Require: $dt\ :\ Integer$ ▷ Age limit
Ensure: D: Dictionary From Triple Pattern to List of Endpoints.
 1: **function** SOURCESELECTION(Q,E, $fragments$, P, $\subseteq_{frag}$, div_{frag}, dt)
 2: triplePatterns ← get triple patterns in Q
 3: **for** each $k \in triplePatterns$ **do**
 4:   G(k) ← get candidates c from E, such that c div$_f$rag frag.origin ≤ dt
 5:   split G(k) into set of endpoints that provide the same data fragment
 6:   simplify G(k) using containment among views
 7:   for the sets in G(k) that are not singleton of publicEndpoint, remove publicEndpoint.
 8: **end for**
 9: (S, C) ← get instance of minimal set covering problem using G
10:   C' ← minimalSetCovering(S, C)
11: **for** each $k \in domain(G)$ **do**
12:    G(k) ← filter G(k) according to C'
13:    D(k) ← for each set in G(k) include one of its elements
14: **end for**
15:   **return** D
16: **end function**

Algorithm 1 presents the source selection pseudocode. First, this algorithm pre-select for each triple pattern in the query the sources that can be used to evaluate it, and satisfies the divergence threshold provided by the user. The capability of

a source to provide data for a triple pattern can be implemented using an `ASK` query for dynamic data, or relying on fragment definition for more stable data. In the example given above, G = { (k1, {E1, E3, E4, E5 }), (k2, { E2, E5, E6} }. Second, these pre-selected sources are grouped according to the containment relation between them, some sources can provide the same fragment of data or the data provide by one source may be contained in the data provided by another source. At the end of the first for loop (lines3-8), a list whose elements are list of equivalent endpoints is produced for each triple pattern in the query. This means that they offer the same fragment given their view definition, then during execution only one of them needs to be contacted. And different elements of this resulting list correspond to different fragments that should be considered in order to obtain an answer as complete as possible, modulo the considered endpoints and the allowed divergence threshold. In the example given above, E1, E3 and E5 are grouped together since they provide access to the same fragment, then at the end of this second step G = { (k1, { {E1, E3, E5}, { E4} }), (k2, { {E2, E5}, { E2, E6}} }. Third, these pre-selected sources are simplified according to the containment relation among them. Then the source { E4 } is removed from G(k1), because it provides a fragment already covered by {E1, E3, E5}. Fourth, the public endpoints are removed when possible. In the example, endpoint E2 is removed since its contents can be accessed using E5 and E6. Fifth, a general selection takes place, considering the pre-selected sources for each triple pattern in the query. This last part can be reduce to the well-known set covering problem, and an existing heuristic like the one given in [6] may be used to perform the procedure indicated in line 10. To illustrate this reduction, in the example $G = \{(k1, \{\{E1, E3, E5\}\}), (k2, \{\{E5\}, \{E6\}\})\}$, corresponds to $S = \{k1_1, k2_1, k2_2\}$ and $C = \{E1, E3, E5, E6\}$, $E1 = E3 = \{k1_1\}$, $E5 = \{k1_1, k2_1\}$, $E6 = \{k2_2\}$. Here ki_j represent the different fragments of ki that should be covered to produce a complete answer. In the case of $k1$, only one fragment should be covered, and in the case of $k2$ two different fragments should be covered $k2_1$ and $k2_2$. Then a minimal set covering solution $C' = \{E5, E6\}$ covers all set S, and has minimal size. This solution is used in line 12 to filter $G(k)$ for each k, then $G = \{(k1, \{\{E5\}\}), (k2, \{\{E5\}, \{E6\}\}\}$. In this case, each element of $G(k)$ is a singleton, but otherwise a last step may be performed to choose among endpoints that provide the same fragment and ensure a better query processing by existing federated query engines. Nevertheless, these alternative sources could be used to speed up query processing, for example by getting a part of the answer from each endpoint and combining them.

Proposition 1. *Let n be the number of triple pattern in the query, s be the number of available sources, and v be the maximal number of views that describe a source, the time complexity of Algorithm 1 is $O(n \times s \times Max(s, v))$*

3 Experiments

The aim of this section is to give evidence of FEDRA benefits over existing approaches. We compare FEDRA to DAW source selection algorithm as it is the

8

closest approach for federations of SPARQL endpoints. The selected sources are used to annotate SPARQL queries with the SERVICE clause. Execution of the annotated queries is performed using FedX, additionally direct FedX execution is used as baseline.

Table 1: Queries caracteristics

Queries	# Answers	# Triple Patterns	Distinct	Union	Filter	Optional	Regex	Bound
q1	3	1	1	0	0	0	0	0
q2	1	1	1	0	0	0	0	0
q3	720	5	0	0	0	1	0	0
q4	39	5	0	2	0	0	0	0
q5	9310	4	0	0	1	0	0	0
q6	5	1	0	0	1	0	0	0
q7	8	9	1	8	0	0	0	0
q8	5	1	0	0	1	0	0	0
q9	0	3	0	0	1	0	0	0
q10	1814176	2	0	0	1	0	1	0
q11	0	2	0	0	2	1	0	1
q12	12840	4	0	0	2	2	0	0
q13	2	3	1	0	1	0	0	0
q14	1	1	1	0	1	0	0	0
q15	4	3	0	2	1	0	0	0
q16	61089	2	1	0	1	1	0	0
q17	6	2	0	1	2	0	0	0
q18	32	1	0	0	0	0	0	0
q19	163	2	0	0	0	0	0	0
q20	312	3	0	0	0	0	0	0

Dataset, Queries and Federations Benchmark: we used the last three DB-pedia Live dataset dumps available [7] [8][9], and queries from the DBpedia SPARQL Benchmark [10]. Table 1 summarize the characteristics of the queries. 600 Fragments of this dataset were defined using random basic graph pattern queries that concern the benchmark queries, having 100 fragments of each size between 1 and 6. The fragments were randomly distributed in 100 endpoints, each fragment may be assigned to 0-3 endpoints.

In order to characterize FEDRA behavior, we set up federations that comprise a varying number of participants: 10, 25, 50 and 100.

Virtuoso[10] endpoints are used, and timeouts were set up to 1,800 secs. and 100,000 tuples. Listing (3) presents a query that defines a fragment [11].

Divergence among replicated fragments: To study the impact of data updates, and divergence over the staleness of query answers, federations members have included fragments taken from the three considered epochs of the dataset.

---

[7] http://live.dbpedia.org/dumps/dbpedia_2013_07_18.nt.bz2

[8] http://live.dbpedia.org/dumps/dbpedia_2013_06_17.nt.bz2

[9] http://live.dbpedia.org/dumps/dbpedia_2013_05_17.nt.bz2

[10] http://virtuoso.openlinksw.com/, November 2013.

[11] All the fragments definitions and federations configurations are available at the project website: https://sites.google.com/site/fedrasourceselection/

Listing 3: DBpedia fragment 2

```
CONSTRUCT WHERE {
    ?x1   rdf:type   ?x3 .
    ?x1   dbpedia:nearestCity   ?x4 .
    ?x1   dbpedia:iucnCategory   ?x5 .
}
```

Implementations: FEDRA is implemented using Java 1.7 and the Jena 2.11.0 library [12]. FEDRA produces SPARQL 1.1 queries where each triple pattern is annotated with a service clause that indicates where it will be executed. We have done a likewise reference implementation of DAW [12], as its code is not available for comparison. These queries are posed to FedX3.0[13]. FedX is an state-of-the-art federated engine that process both 1.0 and 1.1 SPARQL queries, this characteristic is crucial to show that FEDRA benefits the engine with respect to its source selection strategy.

Evaluation Metrics: *i*) *Number of Selected Public Sources (NSPS):* corresponds to the sum of the number of times the public endpoint has been selected per triple pattern. *ii*) *Number of Selected Sources (NSS):* corresponds to the sum of the number of sources that has been selected per triple pattern. *iii*) *Execution Time (ET):* corresponds to elapsed time since the query is posed by the user and the answers are completely produced. It is detailed in source selection time (SST), and query execution by the underlying engine (ETUE). Time is expressed in seconds (secs.). A timeout of 600 secs. has been enforced. Time was measured using System.currentTimeMillis() provided by Java and divided by 1000. *iv*) *Completeness (C):* corresponds to the size of the bag intersection between the obtained answers and the expected answers divided by the number of expected answers; where the expected answers are the ones obtained from an endpoint containing the whole dataset. *v*) *Staleness (S):* corresponds to the size of the bag difference between the obtained answers and all the occurrences of the expected answers, divided by the number of obtained answers. *vi*) *Divergence (Div):* corresponds to the distance between a replicated fragment and the same fragment in the public endpoint. A Δ-distance is used, it corresponds to the elapsed time since the last updated was included.

3.1   Impact of the Age Limit over the Results Completeness and Staleness

Table 2 shows the source selection and execution results over a 100 endpoints federation for three different age limit. As the age limits increases up to two months, the number of stale values in the answer remain low, and the answer completeness remain high. Additionally, increasing the age limit allows to consider more appropiate replicas to execute the queries and in consequence there is a reduction in number of times the public endpoint is selected.

---

[12] http://jena.apache.org/, November 2013.

[13] http://www.fluidops.com/fedx/, November 2013.

Table 2: Impact of the age limit (0, 1 and 2 months) over the FEDRA source selection and subsequent execution by FedX.

Query	Div	NSS	NSPS	SST	ETUE	# Answers	C	S
q1	0 months	1	1	0,7670	0,9400	3	1,0000	0,0000
	1 month	1	0	0,9840	1,3000	3	1,0000	0,0000
	2 months	1	0	0,5280	2,1300	3	1,0000	0,0000
q2	0 months	1	1	0,3090	0,9900	1	1,0000	0,0000
	1 month	1	1	0,9090	1,7700	1	1,0000	0,0000
	2 months	2	0	0,3080	3,2400	1	1,0000	0,0000
q3	0 months	5	3	2,8300	39,0500	144	0,2000	0,0000
	1 month	5	2	7,6080	64,9000	144	0,2000	0,0000
	2 months	5	0	3,6560	1,6700	0	0,0000	0,0000
q4	0 months	5	5	1,1090	1,5900	39	1,0000	0,0000
	1 month	5	2	3,3610	0,8200	39	1,0000	0,0000
	2 months	5	2	0,9990	13,9000	39	1,0000	0,0000
q5	0 months	12	3	2,0210	5,7500	83790	1,0000	0,0000
	1 month	17	2	9,0570	300,0000	0	0,0000	0,0000
	2 months	18	0	2,7170	47,2500	139649	1,0000	0,0000
q6	0 months	1	0	1,4240	0,9400	5	1,0000	0,0000
	1 month	1	0	2,0030	0,8800	5	1,0000	0,0000
	2 months	1	0	0,9380	1,9600	5	1,0000	0,0000
q7	0 months	12	3	3,0350	0,9700	8	1,0000	0,0000
	1 month	15	1	7,4240	0,9300	8	1,0000	0,0000
	2 months	14	0	4,0070	5,7700	8	1,0000	0,0000
q8	0 months	1	1	1,3310	0,9600	5	1,0000	0,0000
	1 month	1	0	2,9670	0,8900	5	1,0000	0,0000
	2 months	1	0	0,9030	2,5700	5	1,0000	0,0000
q9	0 months	3	3	1,1620	0,9300	0	1,0000	0,0000
	1 month	3	1	2,2580	0,9900	576	1,0000	1,0000
	2 months	3	1	1,5160	2,1900	576	1,0000	1,0000
q10	0 months	10	2	1,2430	300,0000	123771	0,0000	0,0000
	1 month	13	2	3,3350	300,0000	165026	0,0000	0,0048
	2 months	15	0	1,5110	300,0000	65772	0,0000	0,0000
q11	0 months	2	2	0,8010	0,9200	6	1,0000	1,0000
	1 month	2	1	1,0720	1,2100	6	1,0000	1,0000
	2 months	2	0	0,7360	1,8100	6	1,0000	1,0000
q12	0 months	4	2	1,5140	300,0000	0	0,0000	0,0000
	1 month	4	1	4,2710	300,0000	0	0,0000	0,0000
	2 months	4	0	2,7910	300,0000	0	0,0000	0,0000
q13	0 months	3	2	1,9270	1,3000	2	1,0000	0,0000
	1 month	3	1	3,7320	1,5200	2	1,0000	0,0000
	2 months	3	0	1,9500	2,1600	2	1,0000	0,0000
q14	0 months	1	0	0,6700	0,9200	1	1,0000	0,0000
	1 month	1	0	0,7020	2,0900	1	1,0000	0,0000
	2 months	1	0	0,6170	2,1200	1	1,0000	0,0000
q15	0 months	1	0	1,3200	0,9600	4	1,0000	0,0000
	1 month	1	0	3,6130	1,3000	4	1,0000	0,0000
	2 months	1	0	2,3030	1,2700	4	1,0000	0,0000
q16	0 months	2	2	1,6320	278,3700	21642	0,3500	0,0004
	1 month	2	1	1,3620	300,0000	14546	0,2300	0,0327
	2 months	2	0	1,6940	300,0000	14542	0,2300	0,0327
q17	0 months	1	0	1,9510	1,3000	0	0,0000	0,0000
	1 month	1	0	2,1440	1,4300	0	0,0000	0,0000
	2 months	1	0	2,0670	2,1400	0	0,0000	0,0000
q18	0 months	1	1	0,9960	1,6000	32	1,0000	0,0000
	1 month	1	0	0,7250	1,2200	32	1,0000	0,0000
	2 months	1	0	0,6580	1,9000	32	1,0000	0,0000
q19	0 months	2	2	1,4400	1,3300	163	1,0000	0,0000
	1 month	2	1	1,5190	0,8000	163	1,0000	0,0000
	2 months	2	0	1,4270	2,4400	163	1,0000	0,0000
q20	0 months	3	3	1,9240	26,1400	24	0,0700	0,0000
	1 month	3	2	2,4330	95,7200	24	0,0700	0,0000
	2 months	3	0	2,1810	94,3100	24	0,0700	0,0000

3.2 Reduction of the Number of Selected Sources

The benchmark queries were processed using DAW and FEDRA source selection strategies over federations of sizes 10, 25 and 50, and an age limit of one month. For each size, five random federations were considered, and the average of the results is presented in Table 3. Additionally, FedX execution plan was used to compute its number of selected sources. We hypothesize that FEDRA is able to reduce the number of selected sources and the number of selected public endpoints using the endpoints descriptions. The results suggest that FEDRA is able to reduce the number of selected sources in more than one order of magnitude, and the number of selected public endpoints remains low, and the number of selected sources remains stable independently of the federation size for FEDRA, while it increases with the federation size for FedX and DAW. As depicted by the evolution of NSPS for each query, we can see that public endpoint nearly disappear for the two months age limit and consequently, the load on the public endpoint effectively decreases. In general, FEDRA selects less sources than DAW, but DAW source selection present a greater reduction of the number of times the public endpoint is selected.

3.3 Preservation of the Answer

The selected sources in the previous from the previous part were used to annotate SPARQL 1.1 queries using the SERVICE clause, than the annotated queries were executed using FedX. We hypothesize that executing the queries produced by FEDRA leads to a similar number of answers than executing the queries directly by FedX, possibly some stale values may appear since the used age limit is one month. Table 4 shows the achieved completeness reported by these executions. Some query executions present unexpected results, even if the public endpoint was the only one selected (e.g, q12). An analysis of the cases were the completeness is low and the staleness is high evidences that FedX execution present some problems when special characters are present in the queries or when OPTIONALs and FILTERs are used with SERVICE clases.

3.4 Fedra Cost

Tables 3 and 4 present the source selection time (SST), and execution time (ETUE) for the previous experiment. For most of the queries FEDRA SST time is slightly greater than DAW but in most of the queries lower than FedX. Nevertheless, FEDRA reduces the total execution time in more than 75% of the queries and this reduction may be huge. In some queries the execution time actually increases, we compare FedX execution plans to understand why, and notice that the plan obtained from a query with SERVICE clauses presents less opportunities for FedX optimizations, then maybe another way of providing the source selection to fedX may be used, like filtering the endpoints file and providing FedX just with the ones that have been selected.

Table 3: FEDRA Steadiness of the Number of Selected Sources (NSS), the results correspond to the average of five executions over random federations

Query	Approach	10 endpoints Federation			25 endpoints Federation			50 endpoints Federation		
		NSS	NSPS	SST	NSS	NSPS	SST	NSS	NSPS	SST
q1	FedX	10.8	1.0	1.34	24.2	1.0	1.32	47.4	1.0	1.62
	DAW	3.6	0.0	0.30	4.4	0	0.28	11.0	0.0	0.35
	FEDRA	1.0	1.0	0.30	1.0	0.0	0.32	1.0	1.0	0.41
q2	FedX	21.6	1.0	1.55	24.2	1.0	1.61	47.4	1.0	1.72
	DAW	1.0	0.6	0.27	1.0	0.4	0.23	1.0	0.0	0.23
	FEDRA	1.0	1.0	0.27	1.0	1.0	0.27	1.0	1.0	0.31
q3	FedX	21.6	5.0	1.37	121.8	5.0	1.42	238.6	5.0	1.94
	DAW	34.4	4.0	0.64	76.4	3.2	0.85	143.6	2.4	1.96
	FEDRA	5.0	5.0	0.62	5.0	4.0	1.30	5.0	4.0	2.12
q4	FedX	43.2	5.0	1.52	121.0	5.0	1.54	237.0	5.0	1.74
	DAW	12.8	2.0	0.44	15.2	2	0.40	35.0	2.0	0.68
	FEDRA	5.0	5.0	0.43	5.0	2.0	0.48	5.0	5.0	0.84
q5	FedX	32.4	4.0	1.35	98.0	4.0	1.53	192.0	4.0	1.64
	DAW	25.4	2.4	0.55	57.0	2.2	0.81	108.4	2.0	1.22
	FEDRA	6.6	3.0	0.54	7.8	3.0	0.84	12.4	4.0	1.45
q6	FedX	10.8	1.0	1.32	24.2	1.0	1.53	47.4	1.0	1.36
	DAW	10.0	1.0	0.48	23.0	1	0.65	43.2	1.0	1.66
	FEDRA	1.0	0.0	0.34	1.0	0.0	0.43	1.0	0.0	0.69
q7	FedX	10.8	4.0	1.35	96.8	4.0	1.51	189.6	4.0	1.93
	DAW	16.6	0.0	0.78	29.0	0	1.23	54.8	0.0	2.31
	FEDRA	5.0	4.0	0.79	8.0	2.0	1.58	10.2	3.0	2.33
q8	FedX	21.6	1.0	1.45	24.2	1.0	1.51	47.4	1.0	1.50
	DAW	4.6	0.0	0.38	11.8	0	0.38	20.8	0.0	0.58
	FEDRA	1.0	1.0	0.34	1.0	1.0	0.43	1.0	1.0	0.61
q9	FedX	10.8	3.0	1.55	73.0	3.0	1.45	143.0	3.0	1.95
	DAW	12.4	2.0	0.39	27.2	1.6	0.45	47.2	1.6	0.88
	FEDRA	3.0	3.0	0.41	3.0	3.0	0.65	3.0	3.0	0.77
q10	FedX	10.8	2.0	1.48	49.2	2.0	1.33	96.4	2.0	1.81
	DAW	14.2	1.6	0.43	29.2	1.4	0.46	53.4	1.0	1.09
	FEDRA	3.0	2.0	0.53	5.2	2.0	0.84	9.2	2.0	1.29
q11	FedX	21.6	2.0	1.44	43.4	2.0	1.22	94.8	2.0	1.52
	DAW	7.2	1.0	0.36	11.6	1	0.38	18.8	1.0	0.53
	FEDRA	2.0	2.0	0.35	2.0	1.0	0.43	2.0	1.0	0.57
q12	FedX	10.8	4.0	1.26	98.0	4.0	1.41	192.0	4.0	1.85
	DAW	26.4	2.6	0.63	60.4	3	0.71	113.6	2.2	1.56
	FEDRA	4.0	4.0	0.59	4.0	3.0	0.76	4.0	2.0	1.62
q13	FedX	32.4	3.0	1.37	73.0	3.0	1.56	143.0	3.0	1.91
	DAW	18.0	2.0	0.53	39.6	1.6	0.51	73.0	0.8	0.87
	FEDRA	3.0	3.0	0.45	3.0	3.0	0.58	3.0	2.0	1.05
q14	FedX	54.0	1.0	1.40	24.2	1.0	1.46	47.4	1.0	1.91
	DAW	4.6	0.0	0.30	9.8	0	0.31	18.6	0.0	0.40
	FEDRA	1.0	1.0	0.32	1.0	1.0	0.36	1.0	0.0	0.47
q15	FedX	54.0	1.0	1.46	24.2	1.0	1.52	47.4	1.0	1.88
	DAW	5.0	0.0	0.45	12.8	0	0.75	17.0	0.0	0.93
	FEDRA	1.0	1.0	0.46	1.0	1.0	0.79	1.0	0.0	1.26
q16	FedX	43.2	2.0	1.57	48.8	2.0	1.60	95.6	2.0	1.69
	DAW	13.2	2.0	0.41	28.0	1.6	0.51	52.8	1.6	0.80
	FEDRA	2.0	2.0	0.39	2.0	2.0	0.53	2.0	2.0	0.85
q17	FedX	10.8	1.0	1.58	24.2	1.0	1.53	47.4	1.0	1.72
	DAW	6.8	0.0	0.58	14.2	0	0.54	24.2	0.0	1.26
	FEDRA	1.0	0.0	0.43	1.0	0.0	0.63	1.0	0.0	0.91
q18	FedX	43.2	1.0	1.44	24.2	1.0	1.49	47.4	1.0	1.69
	DAW	4.0	0.0	0.32	9.4	0	0.35	15.0	0.0	0.44
	FEDRA	1.0	1.0	0.31	1.0	1.0	0.38	1.0	1.0	0.48
q19	FedX	10.8	2.0	1.19	48.8	2.0	1.34	95.6	2.0	1.53
	DAW	12.4	0.8	0.42	28.0	0.6	0.54	50.2	0.6	0.78
	FEDRA	2.0	2.0	0.39	2.0	2.0	0.51	2.0	2.0	0.92
q20	FedX	32.4	3.0	1.34	73.0	3.0	1.00	143.0	3.0	1.85
	DAW	18.4	2.0	0.48	38.0	2	0.55	69.8	1.2	1.31
	FEDRA	3.0	3.0	0.46	3.0	3.0	0.75	3.0	3.0	1.23

Table 4: Enhanced scalability thanks to the source selection approaches over different federations sizes

Query	Approach	10 endpoints Federation				25 endpoints Federation				50 endpoints Federation			
		ETUE	# Answers	C	S	ETUE	# Answers	C	S	ETUE	# Answers	C	S
q1	FedX	1.05	3.0	1.00	0.00	1.92	3.0	1.00	0.00	2.64	3.0	1.00	0.00
	DAW	1.58	3.0	1.00	0.00	1.23	3.0	1.00	0.00	1.70	3.0	1.00	0.00
	Fedra	0.83	3.0	1.00	0.00	1.00	3.0	1.00	0.00	1.22	3.0	1.00	0.00
q2	FedX	1.41	1.0	1.00	0.00	1.64	1.0	1.00	0.00	1.55	1.0	1.00	0.00
	DAW	1.71	1.0	1.00	0.00	1.13	1.0	1.00	0.00	0.96	1.0	1.00	0.00
	Fedra	0.68	1.0	1.00	0.00	0.75	1.0	1.00	0.00	0.95	1.0	1.00	0.00
q3	FedX	300.00	0.0	0.00	0.00	300.00	0.0	0.00	0.00	300.00	0.0	0.00	0.00
	DAW	300.00	2381.6	0.74	0.00	300.00	6586.6	0.80	0.00	276.95	9890.0	0.80	0.00
	Fedra	0.60	144.0	0.20	0.00	0.74	144.0	0.20	0.00	0.92	144.0	0.20	0.00
q4	FedX	3.54	421.2	1.00	0.00	14.27	943.8	1.00	0.00	48.57	1864.2	1.00	0.00
	DAW	1.71	0.0	0.00	0.00	2.24	0.0	0.00	0.00	1.53	0.0	0.00	0.00
	Fedra	0.75	39.0	1.00	0.00	0.76	39.0	1.00	0.00	1.19	39.0	1.00	0.00
q5	FedX	300.00	22405471.0	1.00	0.00	300.00	1662636.8	0.20	0.00	300.00	0.0	0.00	0.00
	DAW	300.00	1675486.8	0.60	0.00	300.00	0.0	0.00	0.00	182.54	0.0	0.00	0.00
	Fedra	0.95	20481.4	0.60	0.00	1.08	16758.0	0.40	0.00	1.31	59584.0	0.60	0.00
q6	FedX	1.10	54.0	1.00	0.00	1.08	121.0	1.00	0.00	1.27	240.0	1.00	0.00
	DAW	1.67	50.0	1.00	0.00	2.25	115.0	1.00	0.00	2.80	216.0	1.00	0.00
	Fedra	34.10	5.0	1.00	0.00	34.37	5.0	1.00	0.00	89.49	5.0	1.00	0.00
q7	FedX	1.32	8.0	1.00	0.00	1.11	8.0	1.00	0.00	1.31	8.0	1.00	0.00
	DAW	1.67	8.0	1.00	0.00	2.34	8.0	1.00	0.00	2.86	8.0	1.00	0.00
	Fedra	0.57	8.0	1.00	0.00	0.83	8.0	1.00	0.00	1.11	8.0	1.00	0.00
q8	FedX	1.15	54.0	1.00	0.00	0.94	121.0	1.00	0.00	1.57	240.0	1.00	0.00
	DAW	1.59	23.0	1.00	0.00	2.59	59.0	1.00	0.00	2.32	104.0	1.00	0.00
	Fedra	0.86	5.0	1.00	0.00	1.04	5.0	1.00	0.00	1.28	5.0	1.00	0.00
q9	FedX	47.66	728524.8	1.00	1.00	300.00	4392483.6	1.00	1.00	300.00	4731512.4	1.00	1.00
	DAW	18.64	17625.2	1.00	1.00	156.73	48615.0	1.00	0.80	120.45	276364.2	1.00	1.00
	Fedra	300.00	576.0	1.00	1.00	300.00	576.0	1.00	1.00	300.00	576.0	1.00	1.00
q10	FedX	300.00	1051826.6	0.03	0.02	300.00	1015418.2	0.02	0.01	300.00	1900207.4	0.02	0.01
	DAW	300.00	93712.8	0.00	0.00	222.01	105893.6	0.00	0.00	165.71	52953.6	0.00	0.00
	Fedra	0.60	17567.0	0.00	0.00	0.81	41244.8	0.00	0.00	1.28	113765.6	0.00	0.00
q11	FedX	1.59	10.8	1.00	1.00	1.73	24.2	1.00	1.00	3.01	47.8	1.00	1.00
	DAW	1.32	63.6	1.00	1.00	1.57	138.0	1.00	1.00	3.12	350.4	1.00	1.00
	Fedra	0.75	6.0	1.00	1.00	0.91	6.0	1.00	1.00	0.95	6.0	1.00	1.00
q12	FedX	300.00	0.0	0.00	0.00	300.00	0.0	0.00	0.00	300.00	0.0	0.00	0.00
	DAW	300.00	0.0	0.00	0.00	300.00	0.0	0.00	0.00	300.00	0.0	0.00	0.00
	Fedra	0.71	0.0	0.00	0.00	0.68	0.0	0.00	0.00	1.23	0.0	0.00	0.00
q13	FedX	16.67	2.0	1.00	0.00	214.07	2.0	1.00	0.00	300.00	2.0	1.00	0.00
	DAW	10.51	2.0	1.00	0.00	177.39	2.0	1.00	0.00	256.30	2.0	1.00	0.00
	Fedra	261.63	2.0	1.00	0.00	278.49	2.0	1.00	0.00	300.00	2.0	1.00	0.00
q14	FedX	1.33	1.0	1.00	0.00	2.41	1.0	1.00	0.00	2.18	1.0	1.00	0.00
	DAW	1.37	1.0	1.00	0.00	1.90	1.0	1.00	0.00	2.76	1.0	1.00	0.00
	Fedra	0.78	1.0	1.00	0.00	0.73	1.0	1.00	0.00	0.98	1.0	1.00	0.00
q15	FedX	1.22	43.2	1.00	0.00	2.85	96.8	1.00	0.00	1.81	191.2	1.00	0.00
	DAW	1.64	20.0	1.00	0.00	2.01	51.2	1.00	0.00	3.29	68.0	1.00	0.00
	Fedra	32.09	4.0	1.00	0.00	32.83	4.0	1.00	0.00	36.54	4.0	1.00	0.00
q16	FedX	300.00	7142.4	0.11	0.07	300.00	2764.4	0.04	0.06	300.00	1208.2	0.01	0.15
	DAW	300.00	5640.0	0.08	0.06	300.00	2965.0	0.04	0.08	300.00	1402.0	0.02	0.19
	Fedra	121.89	21224.0	0.33	0.03	181.42	17922.0	0.28	0.03	124.27	9925.0	0.15	0.05
q17	FedX	1.37	64.8	1.00	0.00	2.72	145.2	1.00	0.00	2.83	286.8	1.00	0.00
	DAW	1.76	0.0	0.00	0.00	1.78	0.0	0.00	0.00	2.81	0.0	0.00	0.00
	Fedra	0.67	0.0	0.00	0.00	0.68	0.0	0.00	0.00	0.94	0.0	0.00	0.00
q18	FedX	1.59	345.6	1.00	0.00	300.00	774.4	1.00	0.00	300.00	1529.6	1.00	0.00
	DAW	1.71	128.0	1.00	0.00	2.02	300.8	1.00	0.00	2.17	473.8	1.00	0.00
	Fedra	0.63	32.0	1.00	0.00	0.88	32.0	1.00	0.00	0.94	32.0	1.00	0.00
q19	FedX	2.40	19038.4	1.00	0.00	9.71	98193.2	1.00	0.00	12.61	379105.4	1.00	0.00
	DAW	2.88	5737.6	1.00	0.00	3.33	28977.4	1.00	0.00	8.04	53846.6	1.00	0.00
	Fedra	0.86	163.0	1.00	0.00	0.73	163.0	1.00	0.00	1.13	163.0	1.00	0.00
q20	FedX	300.00	0.0	0.00	0.00	2.46	0.0	0.00	0.00	2.65	0.0	0.00	0.00
	DAW	9.55	12021.8	1.00	0.00	69.25	74661.2	0.80	0.00	84.95	591487.0	1.00	0.00
	Fedra	300.00	24.0	0.07	0.00	300.00	24.0	0.07	0.00	300.00	24.0	0.07	0.00

4 Related Work

The Semantic Web community has proposed different approaches to consume Linked Data from federations of endpoints [1, 4, 8, 13]. Although source selection and query processing techniques have successfully implemented, none of these approaches is able to exploit information about data replication to enhance performance and answer completeness. Recently Saleem et al. propose DAW [12], a source selection technique that relies on data summarization to describe RDF replicas and thus, reduces the number of selected endpoints. For each triple pattern in a SPARQL query, DAW exploits information encoded in source summaries to rank relevant sources in terms of how much they can contribute to the answer. Source summaries are expressed as min-wise independent permutation vectors (MIPs) that index all the predicates in a source. Although properties of MIPs are exploited to efficiently estimate the overlap of two sources, since Linked Data can frequently change, DAW source summaries may need to be regularly recomputed to avoid obsolete answers. To overcome this limitation, FEDRA provides a more abstract description of the sources which is less sensible to data changes; data provenance and timestamps are stored to control divergence.

Divergence and data replication, in general, have been widely studied in distributed systems and databases. There is a fundamental trade-off between data consistency, availability and tolerance to failures. In one extreme, distributed systems implement the ACID transactional model and ensure strong consistency [5]. Although ACID transactional models have been extensively implemented, it has been shown that in large-scale systems where data is partitioned, ensuring ACID transactions and mutual consistency is not feasible [5]. Based on these results, FEDRA does not rely on strong consistency that would constrain Linked Data participants. Contrary, FEDRA exploits source descriptions to reduce the number of contacted endpoints while satisfies divergence thresholds. Thus, depending on the divergence tolerated by a user, a query can be performed against replicas if accessing the original source is not possible.

5 Conclusions and Future Work

We presented FEDRA a source selection approach that takes advantage of replicated data, and reduces the number of selected endpoints given a threshold of divergence. A lower value of divergence allows query engines to produce fresher answers but at the risk of failing to get answers. Contrary, a higher value risks engines to retrieve answers that are obsolete, while chances of producing answers increases as the space of possible selected sources is larger.

In the future, we plan to consider a finer-grained granularity of divergence. Instead of considering divergence of a replica, we will compute divergence for specific predicates, *e.g.*, divergence of predicates that represent links between datasets. In addition, we plan to integrate DAW with FEDRA to handle replicas with missing descriptions.

References

1. M. Acosta, M.-E. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus. Anapsid: An adaptive query processing engine for sparql endpoints. In Aroyo et al. [3], pages 18–34.
2. C. B. Aranda, A. Hogan, J. Umbrich, and P.-Y. Vandenbussche. Sparql web-querying infrastructure: Ready for action? In H. Alani, L. Kagal, A. Fokoue, P. T. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty, and K. Janowicz, editors, *International Semantic Web Conference (2)*, volume 8219 of *Lecture Notes in Computer Science*, pages 277–293. Springer, 2013.
3. L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy, and E. Blomqvist, editors. *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*. Springer, 2011.
4. C. Basca and A. Bernstein. Avalanche: Putting the spirit of the web back into semantic web querying. In A. Polleres and H. Chen, editors, *ISWC Posters&Demos*, volume 658 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
5. E. A. Brewer. Pushing the cap: Strategies for consistency and availability. *IEEE Computer*, 45(2):23–29, 2012.
6. S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani. *Algorithms*. McGraw-Hill, 2008.
7. W. M. Golab, M. R. Rahman, A. AuYoung, K. Keeton, and X. S. Li. Eventually consistent: not what you were expecting? *Commun. ACM*, 57(3):38–44, 2014.
8. O. Görlitz and S. Staab. Splendid: Sparql endpoint federation exploiting void descriptions. In O. Hartig, A. Harth, and J. Sequeda, editors, *COLD*, volume 782 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.
9. T. Käfer, A. Abdelrahman, J. Umbrich, P. O'Byrne, and A. Hogan. Observing linked data dynamics. In P. Cimiano, Ó. Corcho, V. Presutti, L. Hollink, and S. Rudolph, editors, *ESWC*, volume 7882 of *Lecture Notes in Computer Science*, pages 213–227. Springer, 2013.
10. M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo. Dbpedia sparql benchmark - performance assessment with real queries on real data. In Aroyo et al. [3], pages 454–469.
11. A. Passant and P. N. Mendes. sparqlpush: Proactive notification of data updates in rdf stores using pubsubhubbub. In *SFSW*, 2010.
12. M. Saleem, A.-C. N. Ngomo, J. X. Parreira, H. F. Deus, and M. Hauswirth. Daw: Duplicate-aware federated query processing over the web of data. In H. Alani, L. Kagal, A. Fokoue, P. T. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty, and K. Janowicz, editors, *International Semantic Web Conference (1)*, volume 8218 of *Lecture Notes in Computer Science*, pages 574–590. Springer, 2013.
13. A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In Aroyo et al. [3], pages 601–616.
14. R. Verborgh, M. Vander Sande, P. Colpaert, S. Coppens, E. Mannens, and R. Van de Walle. Web-scale querying through linked data fragments. In *Proceedings of the 7th Workshop on Linked Data on the Web (Apr 2014)*, 2014.