LSQ 2.0: A Linked Dataset of
SPARQL Query Logs

Claus Stadlera,*, Muhammad Saleema, Qaiser Mehmoodb, Carlos Buí-Aranda, Michel Dumontierd,
Aidan Hogane, Axel-Cyrille Ngonga Ngomoa

a Universität Leipzig, IFI/AKSW, PO 100920, D-04009 Leipzig
b Insight Center for Data Analytics, National University of Ireland, Galway
c IMFD; Informatics Department, Universidad Técnica Federico Santa María, Chile
d Institute of Data Science, Maastricht University, Maastricht, The Netherlands
e IMFD; Department of Computer Science, University of Chile, Santiago, Chile

Abstract. We present the Linked SPARQL Queries (LSQ) dataset, which currently describes 43.95 million executions of 11.56
million unique SPARQL queries extracted from the logs of 27 different endpoints. The LSQ dataset provides RDF descriptions of
each such query, which are indexed in a public LSQ endpoint, allowing interested parties to find queries with the characteristics
they require. We begin by describing the use cases envisaged for the LSQ dataset, which include applications for research on
common features of queries, for building custom benchmarks, and for designing user interfaces. We then discuss how LSQ has
been used in practice since the release of four initial SPARQL logs in 2015. We discuss the model and vocabulary that we use to
represent these queries in RDF. We then provide a brief overview of the 27 endpoints from which we extracted queries in terms of
the domain to which they pertain and the data they contain. We provide statistics on the queries included from each log, including
the number of query executions, unique queries, as well as distributions of queries for a variety of selected characteristics. We
finally discuss how the LSQ dataset is hosted and how it can be accessed and leveraged by interested parties for their use cases.

Keywords: SPARQL federation, Web of Data, RDF

1. Introduction

Since its initial recommendation in 2008 [70], the
SPARQL query language for RDF has received consid-
erable adoption, where it is used on hundreds of public
query endpoints accessible over the Web [92]. The most
prominent of these endpoints receive millions of queries
per month [12], or even per day [57]. There is much
to be learnt from queries received by such endpoints,
where research on SPARQL would benefit—and has
already benefited—from access to real-world queries
to help focus both applied and theoretical research on
commonly seen forms of queries [59].

To exemplify how access to real-world queries can
directly benefit research on SPARQL, first consider the
complexity results of SPARQL [67], which show that
evaluation of SPARQL queries is intractable (PSPACE-

© 2022 – IOS Press and the authors. All rights reserved
have shown that these fragments cover many of the
queries seen in practice [24, 58], where query logs help
to bridge the theory and practice of SPARQL [59].

Another use case for a large collection of rea-
world queries pertains to benchmarking. For over a
decade, the SPARQL community has relied on synthetic
datasets and queries (e.g., LUBM [40], Berlin [19]),
or real-world datasets and hand-crafted queries (e.g.,
BTC [63], FedBench [83]) to perform benchmark-
ing. However, Ałuç et al. [7] and Saleem et al. [82]
find the queries of these benchmarks to often be
too narrow and simplistic. Building benchmarks from
real-world queries can help tune implementations and
guide research towards better support for the types
of queries most commonly encountered in practical
settings [13, 16, 62, 65, 79, 100]. Yet another use
case is caching [50, 54, 99]. Here, real-world queries
can be used to simulate practical workloads experi-
enced by endpoints. The usability of SPARQL inter-
faces [24, 25, 52, 74] can also benefit from query logs,
as these logs can reveal patterns in how users increment-
tively build their queries, as has recently been studied by
Bonifati et al. [24] in DBpedia logs. These use cases
do not touch on the subject of queries, as they have
be discussed in more detail in Section 2.

Recognising the value of query logs, a number of
such collections have been published previously, in-
cluding contributions from USEWOD [55, as well as
Wikidata [57]. These logs have been widely used and
analysed by a variety of authors (e.g., [12, 21, 23, 57, 68, 73]). However, i) these logs are provided in ad-hoc
formats, varying in terms of syntax and information pro-
vided depending on the particular SPARQL implemen-
tation used to host the endpoint. ii) Typically, queries
are published as strings, meaning (for example) that a
client would need to use a SPARQL query parser and
some procedural code to find queries matching particu-
lar structures or characteristics. iii) Moreover, runtime
statistics in terms of—for example—the selectivity of in-
dividual query patterns with respect to the base dataset
of the endpoint are not provided. iv) Furthermore, these
datasets have generally been limited to publishing logs
from a small number (1–4) of endpoints.

In this dataset description paper, we extend upon our
previous work [77], which reported on the initial release
of the Linked SPARQL Query Dataset (LSQ). The goal
of LSQ is to publish queries from a variety of SPARQL
logs in a consistent format and associate these queries
with rich metadata, including both static metadata (i.e.,
considering only the query) and runtime metadata (i.e.,
considering the query and the dataset). In particular,
we propose an RDF representation of queries that cap-
tures their source, structure, static metadata and runtime
metadata. These RDF descriptions of queries are in-
dexed in a SPARQL endpoint. Thus, they allow clients
to retrieve the queries of interest to them using a SPARQL
endpoint. The LSQ dataset has grown considerably: LSQ 2.0
now features logs from 27 endpoints (22 of which are from Bio2RDF) compared with 4 initial end-
points. As a result, the number of query executions
described by the LSQ 2.0 dataset has grown from
5.68 million to 43.95 million.

Based on the experiences gained from the first
version of LSQ, we have improved the RDF model
to provide better modularity and more detailed
metadata, facilitating new ways in which clients
can select the queries of interest to them; we have
likewise updated the LSQ vocabulary accordingly.

We have re-engineered the extraction framework,
which takes as input raw logs produced by a vari-
ety of popular SPARQL engines and Web servers,
producing an output RDF graph in the LSQ 2.0
data model describing the queries. The RDFization
process can now be scaled as it leverages Apache
Spark\(^2\). The LSQ software framework has been
released as open source.

We have evaluated the new queries locally in a
Virtuoso instance in order to gain runtime statistics
(including estimates of the number of results, the
selectivity of patterns, overall runtimes, etc.), and
have updated the statistical analysis of the queries
featured by LSQ to include the additional data
provided by the new endpoints.

Since the initial release, LSQ has been used by a
variety of diverse research works on SPARQL [2, 69, 72, 83, 90, 96–98, 101]. To exemplify the value
of LSQ, we discuss the various ways in which the
dataset has been used in these past years.

LSQ 2.0 is available at http://aksw.github.io/LSQ/.
The rest of the paper is structured as follows:

\(^1\)http://usewod.org/; retr. 2015/04/14.

\(^2\)https://spark.apache.org/
– Section 2 describes use cases envisaged for LSQ.
– Section 3 details the model and vocabulary used by LSQ to represent and describe SPARQL queries.
– Section 4 describes how LSQ is published following Linked Data principles and best practices.
– Section 5 first describes the datasets for which LSQ indexes queries, and then provides details on the raw logs from which queries are extracted.
– Section 6 provides an analysis of the LSQ dataset itself, as well as the information it contains.
– Section 7 describes how LSQ has been adopted for the past six years since its initial release.
– Section 8 concludes and discusses future directions for the LSQ dataset.

2. Use Cases

To help motivate the Linked SPARQL Queries dataset, we first discuss some potential use cases that we envisage. We then list some general requirements for LSQ that arise from these use cases.

UC1 Custom Benchmarks A number of benchmarks have been proposed recently based on real-world queries observed in logs [16, 62, 79, 100]. The LSQ dataset can support the creation of such benchmarks, allowing users to select queries from a diverse selection of logs based on custom criteria matching the metadata provided by LSQ. Queries may be selected so as to provide a general benchmark that is representative of real-world workloads, or a specialised benchmark focused on particular query characteristics, such as path expressions, multi-way joins, and aggregation queries.

UC2 SPARQL Adoption Various works have analysed SPARQL query logs in order to understand how features of the SPARQL standard are used “in the wild” as well as to extract structural properties of real-world queries [12, 21, 23, 24, 57, 68, 73]. In turn, this family of works has led to the definition of tractable fragments of queries that are common in practice [20, 58]. LSQ can facilitate further research on the use of SPARQL in the wild as it compiles logs from different domains.

UC3 Caching Techniques for SPARQL caching [50, 60, 66, 99] aim to re-use solutions across multiple queries. Caching allows for reducing the computational requirements needed to evaluate a workload, particularly in cases where queries are often repeated and the underlying data do not change too frequently. The LSQ dataset can again provide a sequence of real-world queries for benchmarking caching systems in realistic settings.

UC4 Usability Aside from efficiency, a crucial aspect of SPARQL research and development is to explore techniques that allow non-expert users to express queries against endpoints more easily. A number of techniques have been proposed to enhance the usability of SPARQL endpoints, including works on auto-completion [25, 52, 74], query relaxation [38, 43, 95] and query builders [10, 27, 44, 94]. Such works could use the LSQ dataset to investigate patterns in how users iteratively formulate more complex queries, causes for queries with empty results, as well as to detect the most important features that interfaces must support.

UC5 Optimisation Understanding the most common cases encountered in real-world queries can allow for optimising implementations towards those cases. One such optimisation is to define workload-aware schemes for local [8, 9] and distributed [4, 28, 45] indexing that attempt to group data commonly requested together in the same region of storage; other optimisations look at scheduling the execution of parallel query requests in an effective and fair manner [56], or propose efficient algorithms for frequently encountered patterns in queries [58]. The LSQ dataset can provide diverse examples of real workloads to help configure and evaluate such techniques.

UC6 Meta-Querying The final use case is admittedly more speculative. By meta-querying, we refer to LSQ being used to query for queries of interest, for example, to find the (most common) queries that are asked about specific resources, such as finding out what queries are being asked involving dbpedia:Zika_virus, or what frequent co-occurrences of resources appear in queries. Meta-querying along these lines may help to understand what are the common information needs of users.

These six use cases are intended to help motivate the dataset, to give ideas of potential applications, and also to help distil some key requirements for the design of the dataset. The list should not be considered complete, as other use cases will naturally arise in future. We identify the following facets of the dataset as relevant to support the aforementioned six use cases.

F1 Static Query Features LSQ should describe the key features of each query independently of the
dataset. These include SPARQL keywords (e.g., UNION, DISTINCT), syntactic features (e.g., property paths), and structural features (e.g., multi-way joins, number of projected variables, statistics relating to basic graph patterns (BGPs), etc.). Furthermore, the query should make the resources it mentions explicit. Static features are of key importance to UC1, UC2, UC4, UC5 and UC6.

**F2 Provenance** LSQ should provide provenance metadata about the execution of each query, including the endpoint it was issued to, a timestamp of when it was executed, and an anonymised identifier for the client. Timestamps are of particular importance to UC3 and UC4, while an anonymised identifier for the client is mostly of importance to UC4.

**F3 Runtime Query Statistics** LSQ should include statistics of the evaluation of the query over the original dataset, including the number of results returned, the estimated runtime, and the selectivity of individual patterns in the query. Again, making such statistics available allows clients to select and analyse queries with regard to these features without having to execute them over the original dataset. Runtime statistics are of particular importance to UC1, UC3, UC4 and UC5.

These facets guide the design of the LSQ dataset in terms of what is included, and how the descriptions of individual queries are represented in RDF.

### 3. Data Model & Vocabulary

In this section, we describe the data model and vocabulary employed by LSQ for describing SPARQL queries. First, we identify a number of desiderata:

**D1 Generality** The data model should facilitate a variety of use cases and cover at least the aforementioned facets (F1–F3) without the need for clients to parse the raw query strings.

**D2 Conciseness** With logs containing millions of queries, the data model should be relatively concise—in terms of triples produced per query—to keep LSQ at a manageable volume of data.

**D3 Usability** Core competency questions over the dataset (e.g., find all queries using a particular feature) should be expressible in terms of simple queries that are efficient to evaluate.

**D4 Linked Data Compatibility** URIs should be dereferenceable so as to abide by the Linked Data Principles. Terms from external well-known vocabularies should be re-used where appropriate. Links to other datasets should be provided.

It is important to note that some of these desiderata are incompatible. For example, D2 is in direct conflict with D1 as adding more meta-data for queries can increase generality, but decreases conciseness. D2 can also be seen as being in conflict with D3 and D4, as D3 can be achieved by adding “shortcut” representations for common needs, while D4 requires the addition of links to external datasets, both of which reduce conciseness. Consequently, the data model must find a balance between providing a detailed description of each query, being useful for various purposes, and keeping the overall dataset relatively concise and manageable.

In Figure 1 we provide an overview of the model used to represent queries in RDF, while in Listing 1 we provide a snippet of the top-level data generated for a query found in the SWDF logs. We now discuss the groups of features described for each query.

**Query instance** We define a “query” to be uniquely identified by the syntactic query string (independently of the endpoint, the particular execution, etc.). We type these queries with `lsqv:Query`. Instances of this class are linked to the query string using `lsqv:query`, and to various instances of local and remote executions. Other links are provided to other resources that capture further details of the static features of the query, its structure, as well as runtime statistics of its local execution (on our server) as information about its remote execution (on the original server).

**Static features** Next we define some static features of the query, independent of the dataset over which it is evaluated. These include links to its individual join variables, triple patterns, and basic graph patterns; the SPARQL features that is uses; its number of projected variables, basic graph patterns, join variables, triple patterns; the maximum, mean and median degree of its join variables; and the maximum and minimum size of its basic graph patterns. The triple patterns and basic graph patterns themselves link to the SPIN representation of the query included in the description (and discussed presently); the triple patterns, in turn, link to

---

3Note that for the purposes of presentation, we abbreviate some of the details of the query, including the IRIs used to identify local query executions.
Listing 1: An example LSQ/RDF representation of a SPARQL query in Turtle syntax

```turtle
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix lsqr: <http://lsq.aksw.org/> .
@prefix lsqv: <http://lsq.aksw.org/vocab#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swc: <http://data.semanticweb.org/ns/swc/ontology#> .
@prefix swr: <http://data.semanticweb.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix prov: <http://www.w3.org/ns/prov#> .

# Primary resource describing the query found with the SWDF logs
lsqr:lsqQuery-3wBd2uKotB_-vUxnngs6ZNsGPhjID09c7igOU24y8
  lsqv:hasLocalExec lsqr:localExec-v9fBp3IaiVXMII12Bx1ijcXK3iy-axTgRvU2c7NY8 ;
  lsqv:hasRemoteExec lsqr:re-data.semanticweb.org-sparql_2014-05-22T16:08:17Z ,
  lsqv:hasStructuralFeatures lsqr:lsqQuery-3wBd2uKotB_-v9fBp3IaiVXMII12Bx1ijcXK3iy-axTgRvU2c7NY8-sf ;
  lsqv:hasBgpCount 1 ;
  lsqv:hasJoinVertexCount 1 ;
  lsqv:hasJoinVertexDegreeMean 2 ;
  lsqv:hasJoinVertexDegreeMedian 2 ;
  lsqv:hasProjectVarCount 1 ;
  lsqv:hasTriplePatternCountMax 2 ;
  lsqv:hasTriplePatternCountMean 2 ;
  lsqv:hasTriplePatternCountMedian 2 ;
  lsqv:hasUsesFeatures lsqv:fn-isLiteral , lsqv:Select , lsqv:Limit , lsqv:Functions , lsqv:Group , lsqv:Filter ,
  lsqv:Distinct , lsqv: TriplePattern .

# Static features of the query
lsqr:lsqQuery-3wBd2uKotB_-v9fBp3IaiVXMII12Bx1ijcXK3iy-axTgRvU2c7NY8-sf
  lsqv:hasRdfType lsqr:bgp- _xMckkke-V9RM3ddISuw-Nij_j278nTShwi1WUMK7tGf ;
  lsqr:joinVertexCount 1 ;
  lsqr:joinVertexDegreeMean 2 ;
  lsqr:joinVertexDegreeMedian 2 ;
  lsqr:projectVarCount 1 ;
  lsqr:triplePatternCountMax 2 ;
  lsqr:triplePatternCountMean 2 ;
  lsqr:triplePatternCountMedian 2 ;
  lsqv:usesFeatures lsqv: fn-isLiteral , lsqv: Select , lsqv: Limit , lsqv: Functions , lsqv: Group , lsqv: Filter ,
  lsqv: Distinct , lsqv: TriplePattern .

# Remote execution no. 1 on the original endpoint
lsqr:re-data.semanticweb.org-sparql_2014-05-22T16:08:17Z
  lsqv:endpoint swr:sparql ;
  lsqv:hostHash "05UQpDtofxAsrJk7yzGfDolFGylMFw5446KcRZDcBkU" .

# Remote execution no. 2 on the original endpoint
  lsqv:endpoint swr:sparql ;
  lsqv:hostHash "7aF9vqgizRufH7-c0-doQlX-oxJx-xFbnOKEEv_E" .

# Local execution to extract statistics
lsqr:localExec-v9fBp3IaiVXMII12Bx1ijcXK3iy-axTgRvU2c7NY8-xc
  lsqv:benchmarkRun lsqr:xc-swdf_2020-09-23_at_23-09-2020_17:10:19 ;
  lsqr:hasQueryExec lsqr:queryExec-Cmv75cybBxwkep_crvD1F3p3iq29t8HM1DFl1CHqU .

# Results of local execution
lsqr:queryExec-Cmv75cybBxwkep_crvD1F3p3iq29t8HM1DFl1CHqU
  prov:atTime "2020-09-23T16:27:36.325Z"^^xsd:dateTime ;
  lsqr:countingDuration 0.00846651 ;
  lsqr:evalDuration 0.00868635 ;
  lsqr:resultCount 16 .

# The full data further include a SPIN description of the query, a list of BGPs within the query,
# a list of triple patterns and terms within the query, as well as execution statistics for individual
# BGPs, triple patterns and sub-BGPs induced by join variables
```
the remote execution is also linked to the originating endpoint using `lsqv:endpoint`.\(^6\) Given that these meta-data constitute provenance for the query, we use the PROV Ontology (PROV-O)\(^{[51]}\) for modelling the time, date and agent involved in the remote execution.

**Local execution** In most cases, the log of the remote executions will not provide statistics about the execution of the query in terms of how many results were returned, how long it took, how selective were the individual patterns, and so forth. Hence we re-execute the queries offline against the original dataset to generate runtime statistics about the query. Local executions were run on a machine with 64 core Intel(R) Xeon(R)...

---

\(^{[4]}\) Given a query \(Q\) and dataset \(D\), let \(Q(D)\) denote the result(s) of evaluating \(Q\) over \(D\). Two queries \(Q_1\) and \(Q_2\) are then defined to be equivalent if and only if \(Q_1(D) = Q_2(D)\) for every dataset \(D\).

\(^{[5]}\) A “salt” in cryptography is a privately-held arbitrary string that is combined (e.g., concatenated) with the input being hashed in order to avoid attacks based on precomputed tables (e.g., of common values or, in this case, of a collection of I.P.’s of interest).
CPU E5-2683 v4 @ 2.10GHz, and 528 GB RAM running Ubuntu 18.04.5 LTS using Virtuoso 7.2.\(^7\) Due to the large number of queries to evaluate, we set a query timeout of one minute. The statistics generated for the number of results of the runtime for the query, as well as the number of results and the selectivity for each individual triple pattern.\(^8\) Runtime statistics are computed in a controlled environment that abstract away external factors such as the load on the endpoint server, etc.; however, due to the costs involved in evaluating such queries, we compute these only for one query engine, namely Virtuoso 7.2, where runtime estimates may thus vary for other engines.

**Summary** The meta-data described in this section aim to strike a balance in terms of the four desiderata mentioned previously. In terms of **Generality**, we provide detailed meta-data for static query features, for provenance, and for runtime query statistics. In terms of **Conciseness**, though the detailed meta-data do require potentially many triples to be encoded for each query, we take steps to reduce this number by re-using resources insofar as appropriate where, for example, each unique query string is encoded once per log, with one set of static features, one SPIN representation, and one set of local executions, being subsequently linked to its different remote executions (rather than duplicate the former meta-data each time the same query string appears in the log). In terms of **Usability**, we provide some “shortcut triples” that allow for quickly finding queries of interest; for example, the static features of the query are largely of this form, where all such meta-data could in principle be computed from the SPIN representation, but using rather complex SPARQL queries over LSQ; the static query features are thus presented to make it easier to find queries, for example, with a certain range of numbers of triple patterns, or queries using `DISTINCT` and `GROUP BY`, etc. We will discuss **Linked Data Compatibility** in the section that follows.

### 4. Publication

The LSQ dataset is published as Linked Data. Before describing the current contents of LSQ, we discuss in more detail how LSQ has been published.

\(^7\)The configuration used for Virtuoso was `MaxQueryMem = 320`, `NumberOfBuffers = 20050000`, and `MaxDirtyBuffers = 20000000`.

\(^8\)The selectivity of the triple pattern is the ratio of triples from the dataset that it selects.

### 5. LSQ 2.0 Logs

We now describe the content of the LSQ 2.0 dataset. In order to collect raw SPARQL query logs, we sent...
mails both to the public-lod@w3.org mailing list and to individual providers of endpoints. We also incorporated logs from LSQ 1.0 [77] and a sample of queries from the Wikidata logs [57]. We thus acquired access to the logs of 27 endpoints, 22 of which are part of Bio2RDF release 3 [33]. Table 2 provides high-level statistics of the query logs from which we extract the LSQ dataset, including the query executions registered; the unique query strings; the number of queries providing a runtime error, or returning zero results; as well as the percentage of unique queries using SELECT, CONSTRUCT, DESCRIBE or ASK. Aside from the initial log of LSQ, only one log is already publicly available, namely Wikidata [57], of which we include a subset described in our data model.

<table>
<thead>
<tr>
<th>Method</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linked Data IRIs</td>
<td><a href="http://lsq.aksw.org/lsqQuery-3wBd2uKotB_-vUxmgS6ZNsGPhJmIDD9c7ig0UI24y8">http://lsq.aksw.org/lsqQuery-3wBd2uKotB_-vUxmgS6ZNsGPhJmIDD9c7ig0UI24y8</a> (example)</td>
</tr>
<tr>
<td>Vocabulary</td>
<td><a href="http://lsq.aksw.org/vocab">http://lsq.aksw.org/vocab</a></td>
</tr>
<tr>
<td>Dumps</td>
<td><a href="http://lsq.aksw.org/downloads">http://lsq.aksw.org/downloads</a></td>
</tr>
<tr>
<td>SPARQL Endpoint</td>
<td><a href="http://lsq.aksw.org/sparql">http://lsq.aksw.org/sparql</a></td>
</tr>
<tr>
<td>Catalogue</td>
<td></td>
</tr>
<tr>
<td>Datahub</td>
<td><a href="https://datahub.io/dataset/lsq">https://datahub.io/dataset/lsq</a></td>
</tr>
<tr>
<td>LOV</td>
<td><a href="https://lov.linkeddata.es/dataset/lov/vocabs/lsq">https://lov.linkeddata.es/dataset/lov/vocabs/lsq</a></td>
</tr>
<tr>
<td>prefix.cc</td>
<td><a href="http://prefix.cc/lsqv">http://prefix.cc/lsqv</a></td>
</tr>
</tbody>
</table>

Table 1. Locations from which LSQ can be accessed including an example Linked Data IRI, the vocabulary, dumps, the SPARQL endpoint, as well as locations where LSQ is indexed, including DataHub, Linked Open Vocabularies (LOV) and prefix.cc

 AFFYMETRIX is a biomedical Linked Dataset describing probesets found in DNA microarrays [33].

 BIOMODELS is a biomedical Linked Dataset describing mathematical models of biological systems [33].

 BioPortal is a biomedical Linked Dataset cataloguing biomedical ontologies [33].

 CTD: Comparative Toxicogenomics Database is a biomedical Linked Dataset that describes how environmental chemicals relate to diseases [33].

 DBPEDIA is a cross-domain Linked Dataset that is primarily extracted from Wikipedia [53].

 DBSNP: Single Nucleotide Polymorphism Database is a biomedical Linked Dataset that describes single nucleotide substitutions and short deletion and insertion polymorphisms [33].

 DRUGBANK is a biomedical Linked Dataset that describes drugs and drug targets [33].

 GENAGE is a biomedical Linked Dataset that describes human and other genes linked with ageing [33].

 GENDR: Dietary Restriction Gene Database is a biomedical Linked Dataset that describes genes associated with dietary restrictions [33].

 GO: Gene Ontology is a biomedical ontology that describes gene, gene products, and their functions [33].

 GOA: Gene Ontology Annotation is a biomedical Linked Dataset that provides annotations on proteins, RNA and protein complexes [33].

 HGNC: HUGO Gene Nomenclature Committee is a biomedical Linked Dataset that describes human gene nomenclature [33].

 iREFFINDEX is a biomedical Linked Dataset that indexes interaction data for proteins [33].

 KEGG: Kyoto Encyclopedia of Genes and Genomes is a biomedical Linked Dataset that describes functions of genes and biological systems [33].

 LINKEDGEODATA is a geographical Linked Data extracted primarily from Open Street Map [87].

 LINKEDSQP: Linked Structured Product Labelling is a biomedical Linked Dataset that contains meta-data about drug labels sourced from DailyMed [33].

 MGI: Mouse Genome Informatics is a biomedical Linked Dataset that describes mouse genes, alleles, and strains [33].
NCBI Gene is a biomedical Linked Dataset that describes gene-related information given by the National Center for Biotechnology Information (NCBI) [33].

Online Mendelian Inheritance in Man (OMIM) is a biomedical Linked Dataset that catalogues human genes as well as genetic traits and disorders [33].

PHARMGKB is a biomedical Linked Dataset describing how genetic variations impact drug responses [33].

SABIORK: System for the Analysis of Biochemical Pathways – Reaction Kinetics is a biomedical Linked Dataset that describes biochemical reactions [33].

SGD: Saccharomyces Genome Database is a biomedical Linked Dataset describing the biology and genetics of the yeast Saccharomyces cerevisiae [33].

SIDER: Side Effect Resource is a biomedical Linked Dataset describing the side effects of drugs [33].

SWDF: Semantic Web Dog Food is a bibliographical Linked Dataset describing papers, presentations and people participating in top Semantic Web related conferences and workshops [61].

TAXONOMY: NCBI Taxonomy is a biomedical Linked Dataset that describes all organisms found in genetic databases [33].

WIKIDATA is a collaboratively edited knowledge graph hosted by the Wikimedia foundation [57].

WORMBASE is a biomedical Linked Dataset that describes the biology and genome of worms [33].

6. LSQ 2.0 Query Statistics

We now look in more detail at the composition of the queries currently included in the LSQ dataset. In particular, we first look at some high-level statistics for queries in the dataset, before looking at the static features of the query, the agents making the queries, as well as runtime statistics computed against the corresponding dataset. Finally we discuss the composition of the LSQ dataset itself.

High-level statistics: Table 2 provides a high-level analysis of the queries (both query executions and unique queries) appearing in each of the logs considered. From the overall row, we see that LSQ contains 43.95 million query executions and 11.56 million unique queries, implying that each query is executed, on average, 3.8 times within each log. Of the unique queries, 7.7 million (66.9%) have runtime errors; and 2.3 million (20.0%) have no errors but return empty results. A high ratio of runtime errors come from the Bio2RDF logs. The majority of queries are CONSTRUCT queries (60.0%), followed by SELECT (32.3%), DESCRIBE (7.1%) and ASK (0.5%). We find that CONSTRUCT queries are particularly prevalent on Bio2RDF endpoints, while DESCRIBE queries are particularly prevalent on DBPEDIA and Wikidata endpoints, possibly due to the use of such queries for dereferencing Linked Data IRIs through the endpoint.

Static features: Turning to static features, we first look at the percentages of unique queries without parse errors using different SPARQL features (note that we will analyse joins in BGP and property paths later). Table 3 provides statistics for the usage of different features of SPARQL. We see that FILTER is among the most widely used features, along with SPARQL functions and expressions (note that almost all filters use such expressions). This feature is followed by DISTINCT and other solution modifiers, UNION, OPTIONAL, etc. Notably these are all SPARQL 1.0 features. The SERVICE keyword is commonly used on WIKIDATA since the Wikidata Query Service provides a custom service for retrieving multilingual labels as preferred/available.

Next, in Table 4, we provide three types of statistics about the basic graph patterns and property path features used. First, we present the unique number of subject, predicate and object terms used in the BGP of the logs in order to characterise their diversity. We see that DBPEDIA, LINKEDGEODATA and WIKIDATA offer the most diversity, particularly in terms of predicates found in the queries. Second, we present the percentage of queries with different types of joins in the basic graph patterns [81]. Each join variable in a basic graph pattern is analysed in order to understand how they connect triple patterns. We say that a join vertex has an “outgoing link” if it appears as a subject of a triple pattern, and that it has an “incoming link” if it appears as predicate or object. The join types are then defined as follows:

**STAR** has multiple outgoing but no incoming links.

**PATH** has one incoming and one outgoing link.

**HYBRID** has at least one incoming and outgoing link and three or more links overall.

**SINK** has multiple incoming but no outgoing links.

From Table 4, we see that the majority of queries have no joins, but where present, STAR joins are the most frequent, followed by HYBRID and SINK joins. Third, we
while Bio2RDF logs exhibit little use of this feature. The most used such feature is the diversity of the different datasets. Note that client underlie the executions registered in order to compare how many clients (anonymised IPs) and unique queries Next we look at Provenance: Executions and Agents analyse property paths, most queries feature joins; in order to benchmark or

Table 2

<table>
<thead>
<tr>
<th>DATASET</th>
<th>QE</th>
<th>UQ</th>
<th>RE</th>
<th>ZR</th>
<th>SEL (%)</th>
<th>CON (%)</th>
<th>DES (%)</th>
<th>ASK (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFYMETRIX</td>
<td>1,229,339</td>
<td>311,096</td>
<td>277,983</td>
<td>31,659</td>
<td>16.47</td>
<td>83.21</td>
<td>0.02</td>
<td>0.30</td>
</tr>
<tr>
<td>BioModels</td>
<td>1,238,375</td>
<td>435,232</td>
<td>412,984</td>
<td>21,692</td>
<td>41.18</td>
<td>58.75</td>
<td>0.00</td>
<td>0.06</td>
</tr>
<tr>
<td>BIOPORTAL</td>
<td>1,337,804</td>
<td>89,664</td>
<td>85,273</td>
<td>3,389</td>
<td>64.88</td>
<td>34.78</td>
<td>0.00</td>
<td>0.34</td>
</tr>
<tr>
<td>CTD</td>
<td>940,390</td>
<td>287,296</td>
<td>266,999</td>
<td>19,824</td>
<td>11.98</td>
<td>87.76</td>
<td>0.00</td>
<td>0.26</td>
</tr>
<tr>
<td>DBpedia</td>
<td>6,535,500</td>
<td>4,258,941</td>
<td>1,259,972</td>
<td>1,755,338</td>
<td>69.90</td>
<td>3.59</td>
<td>25.23</td>
<td>1.28</td>
</tr>
<tr>
<td>DBSNP</td>
<td>794,023</td>
<td>269,498</td>
<td>267,662</td>
<td>1,698</td>
<td>4.99</td>
<td>94.99</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>DrugBank</td>
<td>1,613,951</td>
<td>379,233</td>
<td>372,022</td>
<td>6,186</td>
<td>46.67</td>
<td>53.33</td>
<td>0.05</td>
<td>0.48</td>
</tr>
<tr>
<td>GENAGE</td>
<td>589,211</td>
<td>265,067</td>
<td>263,205</td>
<td>1,661</td>
<td>5.55</td>
<td>94.43</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>GENDR</td>
<td>690,864</td>
<td>270,697</td>
<td>262,776</td>
<td>7,726</td>
<td>7.53</td>
<td>92.47</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>GO</td>
<td>1,839,991</td>
<td>121,542</td>
<td>88,743</td>
<td>30,082</td>
<td>98.31</td>
<td>0.03</td>
<td>0.67</td>
<td>1.31</td>
</tr>
<tr>
<td>GOA</td>
<td>3,544,273</td>
<td>343,836</td>
<td>310,800</td>
<td>32,317</td>
<td>26.18</td>
<td>73.69</td>
<td>0.06</td>
<td>0.07</td>
</tr>
<tr>
<td>HGNC</td>
<td>1,529,681</td>
<td>335,541</td>
<td>290,483</td>
<td>44,093</td>
<td>22.78</td>
<td>76.89</td>
<td>0.08</td>
<td>0.26</td>
</tr>
<tr>
<td>iRefIndex</td>
<td>1,560,704</td>
<td>309,777</td>
<td>289,546</td>
<td>19,858</td>
<td>18.10</td>
<td>81.88</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>KEGG</td>
<td>66,830</td>
<td>19,871</td>
<td>10,386</td>
<td>8,004</td>
<td>92.04</td>
<td>0.28</td>
<td>99.69</td>
<td>0.00</td>
</tr>
<tr>
<td>LinkedGeoData</td>
<td>154,884</td>
<td>61,897</td>
<td>11,028</td>
<td>13,990</td>
<td>98.58</td>
<td>1.00</td>
<td>0.02</td>
<td>0.40</td>
</tr>
<tr>
<td>LinkedSQP</td>
<td>337,001</td>
<td>204,112</td>
<td>203,534</td>
<td>310</td>
<td>99.69</td>
<td>0.00</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>MGI</td>
<td>1,316,673</td>
<td>319,627</td>
<td>277,080</td>
<td>33,781</td>
<td>78.60</td>
<td>21.21</td>
<td>0.05</td>
<td>0.23</td>
</tr>
<tr>
<td>NCBigene</td>
<td>770,716</td>
<td>216,832</td>
<td>215,938</td>
<td>718</td>
<td>91.26</td>
<td>8.74</td>
<td>0.00</td>
<td>0.04</td>
</tr>
<tr>
<td>OMIM</td>
<td>1,506,621</td>
<td>335,541</td>
<td>290,483</td>
<td>44,093</td>
<td>22.78</td>
<td>76.89</td>
<td>0.08</td>
<td>0.26</td>
</tr>
<tr>
<td>PharmGKB</td>
<td>94,540</td>
<td>24,000</td>
<td>14,597</td>
<td>8,649</td>
<td>60.35</td>
<td>39.65</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>SABIORK</td>
<td>922,407</td>
<td>274,098</td>
<td>253,733</td>
<td>19,938</td>
<td>7.91</td>
<td>92.07</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>SGD</td>
<td>973,281</td>
<td>318,641</td>
<td>309,593</td>
<td>7,199</td>
<td>16.06</td>
<td>80.30</td>
<td>0.30</td>
<td>3.12</td>
</tr>
<tr>
<td>SIDER</td>
<td>599,287</td>
<td>277,766</td>
<td>274,963</td>
<td>1,965</td>
<td>9.38</td>
<td>90.59</td>
<td>0.00</td>
<td>0.03</td>
</tr>
<tr>
<td>SWDF</td>
<td>1,415,567</td>
<td>101,423</td>
<td>30,792</td>
<td>36,789</td>
<td>73.57</td>
<td>0.06</td>
<td>26.17</td>
<td>0.21</td>
</tr>
<tr>
<td>Taxonomy</td>
<td>7,698,898</td>
<td>354,582</td>
<td>334,290</td>
<td>20,041</td>
<td>15.83</td>
<td>84.16</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>WIKIDATA</td>
<td>3,298,254</td>
<td>844,256</td>
<td>520,976</td>
<td>150,395</td>
<td>95.03</td>
<td>0.13</td>
<td>0.87</td>
<td>4.77</td>
</tr>
<tr>
<td>WormBase</td>
<td>1,353,316</td>
<td>498,170</td>
<td>496,325</td>
<td>1,660</td>
<td>49.33</td>
<td>50.66</td>
<td>0.00</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Overall | 43,952,379 | 11,557,656 | 7,729,223 | 2,312,530 | 36.14 | 57.8 | 1.89 | 0.60 |

13Lorenz curves visualise (in)equality in distributions for a given quantity over a given set of elements: a coordinate (x, y) indicates that ratio x of elements (given in ascending order by their quantity) are associated with ratio y of the total quantity. The solid black line indicates a hypothetical equality where each element is associated with the same quality. For example, in Figure 2 on the DBpedia curve, the point (0.80, 0.29) denotes that 80% of clients invoke 29% of the executions (or 20% of the clients invoke 71% of the executions).
Percentage of unique queries without parse errors using the specified SPARQL feature (Sol. Mod. includes the solution modifiers ORDER BY, OFFSET, and LIMIT. Agg. includes aggregation features GROUP BY, HAVING, AVG, SUM, COUNT, MAX, and MIN; Neg. includes NOT EXISTS, and EXISTS. Bind. includes VALUES and BINDING; Graph includes FROM, FROM NAMED, and GRAPH; Func. includes SPARQL functions and expressions)

<table>
<thead>
<tr>
<th>DATASET</th>
<th>UNION OPTIONAL DISTINCT FILTER REGEX SERVICE</th>
<th>SUB-Q. SOL. M. AGG. NEG. BIND. GRAPH FUNC.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFYMATRIX</td>
<td>3.68</td>
<td>0.02</td>
</tr>
<tr>
<td>BIOMODELS</td>
<td>2.64</td>
<td>0.01</td>
</tr>
<tr>
<td>BIOPORTAL</td>
<td>1.50</td>
<td>0.06</td>
</tr>
<tr>
<td>CTD</td>
<td>3.99</td>
<td>0.02</td>
</tr>
<tr>
<td>DBPEDIA</td>
<td>28.68</td>
<td>19.97</td>
</tr>
<tr>
<td>dbSNP</td>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td>DrugBank</td>
<td>2.58</td>
<td>15.55</td>
</tr>
<tr>
<td>GENAGE</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>GENDR</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>GO</td>
<td>9.08</td>
<td>0.16</td>
</tr>
<tr>
<td>GOA</td>
<td>4.17</td>
<td>0.01</td>
</tr>
<tr>
<td>HGNC</td>
<td>3.16</td>
<td>0.02</td>
</tr>
<tr>
<td>irefIndex</td>
<td>9.99</td>
<td>1.00</td>
</tr>
<tr>
<td>KEGG</td>
<td>11.64</td>
<td>1.13</td>
</tr>
<tr>
<td>LinkedGeoData</td>
<td>1.15</td>
<td>19.13</td>
</tr>
<tr>
<td>LinkedSQP</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>MGI</td>
<td>3.57</td>
<td>0.02</td>
</tr>
<tr>
<td>NCBIGENE</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>OMIM</td>
<td>3.52</td>
<td>1.10</td>
</tr>
<tr>
<td>PharmGKB</td>
<td>33.05</td>
<td>0.00</td>
</tr>
<tr>
<td>SABIORK</td>
<td>4.15</td>
<td>0.01</td>
</tr>
<tr>
<td>SGD</td>
<td>1.63</td>
<td>0.01</td>
</tr>
<tr>
<td>SIDER</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>SWDF</td>
<td>40.13</td>
<td>34.08</td>
</tr>
<tr>
<td>Taxonomy</td>
<td>3.19</td>
<td>0.01</td>
</tr>
<tr>
<td>Wikidata</td>
<td>9.27</td>
<td>29.21</td>
</tr>
<tr>
<td>Wormbase</td>
<td>14.16</td>
<td>4.46</td>
</tr>
</tbody>
</table>

Overall 7.22 4.67 10.23 67.57 1.63 2.17 0.66 9.14 3.77 0.00 0.34 3.34 67.57

Static and Runtime Statistics  Next, in order to characterise how complex the queries are to evaluate, in Table 5 we present some relevant static and runtime statistics, where static statistics can be computed from the query string, while runtime statistics require evaluating the query locally (only queries that were successfully run are counted; see Table 2 for statistics on runtime errors). Regarding runtimes, we recall that these were run with a one minute timeout, which represents the max runtime. We see that LINKEDGEODATA contains the most costly queries to run, which appears to correlate with larger result sizes and a larger mean join-vertex degree. Relatively high runtimes are also seen for the KEGG dataset. The simplest queries to run are found in the GENAGE, GENDR and TAXONOMY datasets. These results suggest, for example, that LINKEDGEODATA might be more suitable for consumers looking for a challenging benchmark.

LSQ dataset statistics  The LSQ 2.0 dataset, describing 43.95 million executions of 11.56 million unique
Table 4

<table>
<thead>
<tr>
<th>DATASET</th>
<th>BGP TERMS</th>
<th>JOIN TYPES (%)</th>
<th>PROP. PATH FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SUBJ.</td>
<td>PRED.</td>
<td>OBJ.</td>
</tr>
<tr>
<td>AFFYMETRIX</td>
<td>17,912</td>
<td>432</td>
<td>27,398</td>
</tr>
<tr>
<td>BioModels</td>
<td>14,055</td>
<td>347</td>
<td>120,148</td>
</tr>
<tr>
<td>BIOPORTAL</td>
<td>9,275</td>
<td>130</td>
<td>6,275</td>
</tr>
<tr>
<td>CTD</td>
<td>14,927</td>
<td>276</td>
<td>22,320</td>
</tr>
<tr>
<td>DBpedia</td>
<td>912,943</td>
<td>10,842</td>
<td>1,104,732</td>
</tr>
<tr>
<td>DBSNP</td>
<td>12,825</td>
<td>112</td>
<td>6,069</td>
</tr>
<tr>
<td>DrugBank</td>
<td>37,578</td>
<td>989</td>
<td>34,601</td>
</tr>
<tr>
<td>GenAge</td>
<td>2,666</td>
<td>113</td>
<td>11,875</td>
</tr>
<tr>
<td>GENDR</td>
<td>5,664</td>
<td>104</td>
<td>705</td>
</tr>
<tr>
<td>GO</td>
<td>35,504</td>
<td>394</td>
<td>59,362</td>
</tr>
<tr>
<td>GOA</td>
<td>33,593</td>
<td>204</td>
<td>22,044</td>
</tr>
<tr>
<td>HGNC</td>
<td>23,430</td>
<td>414</td>
<td>36,857</td>
</tr>
<tr>
<td>iREFIndex</td>
<td>20,067</td>
<td>171</td>
<td>28,069</td>
</tr>
<tr>
<td>KEGG</td>
<td>5,620</td>
<td>251</td>
<td>8,964</td>
</tr>
<tr>
<td>LINKEDGEOData</td>
<td>13,498</td>
<td>5,991</td>
<td>2,628</td>
</tr>
<tr>
<td>LINKEDSQP</td>
<td>326</td>
<td>55</td>
<td>144</td>
</tr>
<tr>
<td>MGI</td>
<td>28,702</td>
<td>391</td>
<td>23,867</td>
</tr>
<tr>
<td>NCBIGene</td>
<td>11,753</td>
<td>254</td>
<td>4,427</td>
</tr>
<tr>
<td>OMIM</td>
<td>23,504</td>
<td>623</td>
<td>50,229</td>
</tr>
<tr>
<td>PharmGKB</td>
<td>1,099</td>
<td>83</td>
<td>13,548</td>
</tr>
<tr>
<td>SABIORK</td>
<td>14,224</td>
<td>156</td>
<td>19,775</td>
</tr>
<tr>
<td>SGD</td>
<td>7,228</td>
<td>508</td>
<td>13,460</td>
</tr>
<tr>
<td>SIDER</td>
<td>8,792</td>
<td>152</td>
<td>3,589</td>
</tr>
<tr>
<td>SWDF</td>
<td>25,640</td>
<td>420</td>
<td>10,823</td>
</tr>
<tr>
<td>TAXONOMY</td>
<td>16,201</td>
<td>207</td>
<td>97,298</td>
</tr>
<tr>
<td>WIKIDATA</td>
<td>47,871</td>
<td>11,779</td>
<td>263,974</td>
</tr>
<tr>
<td>WORMBASE</td>
<td>53,807</td>
<td>148</td>
<td>24,083</td>
</tr>
<tr>
<td>Overall</td>
<td>1,398,704</td>
<td>35,546</td>
<td>2,017,264</td>
</tr>
</tbody>
</table>

queries, contains 1.24 billion triples, split into 27 named graphs (one for each of the datasets listed).\(^{14}\)

7. LSQ Adoption

In this section we present how LSQ has been adopted since its initial release with four logs in 2015. We organise this discussion following the motivational use cases we originally envisaged, as presented in Section 2. Table 6 provides an overview of the research works that have used LSQ, and the relevant use case(s) that they target. We now discuss these works in more detail; note that in the case of works that relate to multiple use cases, we will discuss them once in what we identify to be the “primary” related use case. We further discuss some works that have used the LSQ dataset for use cases beyond the six we had originally envisaged.

UC1: Custom Benchmarks  LSQ has been adopted in various works for creating custom benchmarks.

- Saleem et al. [79] present a framework for generating benchmarks that can be used to evaluate SPARQL endpoints under typical workloads; the benchmarks generate query types depending on the features of the queries submitted to the endpoint, where LSQ is used for testing.

\(^{14}\)We exclude some named graphs created by Virtuoso.
Later works by Saleem et al. further propose frameworks for generating benchmarks from LSQ for the purposes of evaluating query containment [80] and federated query evaluation [78], as well as comparing existing SPARQL benchmarks against LSQ in order to understand how representative they are of real workloads [82].

Hernández et al. [42] present an empirical study of the efficiency of graph database engines for answering SPARQL queries over Wikidata; they refer to LSQ to verify that the query shapes considered for evaluation correspond with other analyses of real-world SPARQL queries.

Fernández et al. [35] evaluate various archiving techniques and querying strategies for RDF archives that store historical data; in their evaluation, they select the 200 most frequent triple patterns from the DBPEDIA query set in LSQ.

Azzam et al. [15] use LSQ for retrieving highly-demanding queries from the dataset in order to evaluate their system for dividing the load processed by different SPARQL servers.

Bigerl et al. [18] develop a tensor-based triple store, where they used LSQ as input to the FEASIBLE framework to generate a custom benchmark.

Azzam et al. [14] present a system that dynamically delegates query processing load between clients and servers. The authors use the Linked Data Fragments client/server approach improving it with the aforementioned technique and use 16 queries from LSQ to complement their evaluation.

Davoudian et al. [30] present a system that partitions graphs depending on the access frequency to their nodes. In this way the system implements workload-aware partitioning. The authors use LSQ for evaluating their approach.

Desouki et al. [32] propose a method to generate synthetic benchmark data. To generate these synthetic data they use other RDF graphs available, such as SWDF and DBpedia 2016. They benchmark their approach using queries from LSQ.

Röder et al. [71] develop a method to predict the performance of knowledge graph query engines; to do so the authors use a stochastic generation model that is able to generate graphs of arbitrary sizes similar to the input graph. They use LSQ as a benchmark of real-world queries.

**UC2: SPARQL Adoption** Other works have used LSQ to understand how SPARQL is being used in practice.

Han et al. [41] provide a statistical analysis of the queries of LSQ, surveying both syntactic features, such as the number of triple patterns, the SPARQL features used, the frequency of well-designed patterns; as well as semantic properties, such as monotonicity, weak-monotonicity, non-monotonicity and satisfiability.

Bonifati et al. [21, 22] conduct detailed analysis of the queries in various logs, including LSQ; they
Table 5
Comparison of the mean values of runtime statistics across all query logs (PVs = Project Variables, BGPs = Basic Graph Patterns, TPs = Triple Patterns, JVs = Join Vertices, MJVD = Mean Join Vertex Degree, MTPS = Mean Triple Pattern Selectivity)

<table>
<thead>
<tr>
<th>DATASET</th>
<th>STATIC STATISTICS (mean)</th>
<th>RUNTIME STATISTICS (mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PVs</td>
<td>BGPs</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>AFFYMETRIX</td>
<td>1.93</td>
<td>1.06</td>
</tr>
<tr>
<td>BIOMODELS</td>
<td>1.24</td>
<td>1.04</td>
</tr>
<tr>
<td>BIOPORTAL</td>
<td>1.16</td>
<td>1.03</td>
</tr>
<tr>
<td>CTD</td>
<td>2.56</td>
<td>1.05</td>
</tr>
<tr>
<td>DBPEDIA</td>
<td>2.78</td>
<td>2.37</td>
</tr>
<tr>
<td>DBSNP</td>
<td>1.09</td>
<td>1.02</td>
</tr>
<tr>
<td>DRUGBANK</td>
<td>2.61</td>
<td>1.05</td>
</tr>
<tr>
<td>GENAGE</td>
<td>1.88</td>
<td>1.00</td>
</tr>
<tr>
<td>GENDR</td>
<td>2.73</td>
<td>1.00</td>
</tr>
<tr>
<td>GO</td>
<td>1.46</td>
<td>1.10</td>
</tr>
<tr>
<td>GOA</td>
<td>1.87</td>
<td>1.03</td>
</tr>
<tr>
<td>HGNC</td>
<td>1.91</td>
<td>1.05</td>
</tr>
<tr>
<td>iRefIndex</td>
<td>2.92</td>
<td>1.13</td>
</tr>
<tr>
<td>KEGG</td>
<td>2.27</td>
<td>1.15</td>
</tr>
<tr>
<td>LINKEDGeoData</td>
<td>2.27</td>
<td>1.16</td>
</tr>
<tr>
<td>LINKEDSQP</td>
<td>2.01</td>
<td>1.00</td>
</tr>
<tr>
<td>MGI</td>
<td>2.04</td>
<td>1.04</td>
</tr>
<tr>
<td>NCBIgene</td>
<td>1.39</td>
<td>1.02</td>
</tr>
<tr>
<td>OMIM</td>
<td>1.83</td>
<td>1.07</td>
</tr>
<tr>
<td>PharmGKB</td>
<td>1.96</td>
<td>1.34</td>
</tr>
<tr>
<td>SABIORK</td>
<td>2.96</td>
<td>1.05</td>
</tr>
<tr>
<td>SGD</td>
<td>1.45</td>
<td>1.12</td>
</tr>
<tr>
<td>SIDER</td>
<td>1.34</td>
<td>1.00</td>
</tr>
<tr>
<td>SWDF</td>
<td>4.04</td>
<td>3.37</td>
</tr>
<tr>
<td>TAXONOMY</td>
<td>1.77</td>
<td>1.17</td>
</tr>
<tr>
<td>Wikidata</td>
<td>3.00</td>
<td>2.47</td>
</tr>
<tr>
<td>Wormbase</td>
<td>1.56</td>
<td>1.25</td>
</tr>
<tr>
<td>Overall</td>
<td>2.07</td>
<td>1.26</td>
</tr>
</tbody>
</table>

study a variety of phenomena in these queries, including their shape, their (hyper)treewidth, common abstract patterns found in the property paths, “streaks” that represent a sequence of user reformulations from a seed query, and more besides.

UC3: Caching
LSQ can also be used to simulate real workloads for systems that explore caching techniques.

– Knuth et al. [49] propose a middleware component to which applications register and get notifications when the results of their SPARQL queries change; the authors study the problem of scheduling refresh queries for a large number of registered queries and use LSQ to validate their approach.

– Akhtar et al. [2, 3] propose an approach to capture changes in an RDF dataset and update a cache system in front of the SPARQL endpoint exposing that data; their approach consists of a change metric that quantifies the changes in an RDF dataset, and a weighting function that assigns importance to recent changes; they use LSQ to verify their approach for real workloads.

– Salas and Hogan [76] propose a method for query canonicalisation, which consists in mapping congruous queries—i.e., queries that are equivalent modulo variable names—to the same query string; their main use case is to increase the hit rate of SPARQL caches, where they use LSQ to test efficiency on real-world queries and to see how many congruent queries can be found in real workloads.

– Savafi et al. [75] study SPARQL adoption using LSQ so they can later provide queries to sum-


Research works making use of the LSQ dataset since its initial release, ordered by year and then alphabetically by author name, with relevant use cases indicated (UC1: Custom Benchmarks; UC2: SPARQL Adoption; UC3: Caching; UC4: Usability; UC5: Optimisation; UC6: Meta-Querying)

<table>
<thead>
<tr>
<th>NAME</th>
<th>YEAR</th>
<th>UC 1</th>
<th>UC 2</th>
<th>UC 3</th>
<th>UC 4</th>
<th>UC 5</th>
<th>UC 6</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saleem et al. [79]</td>
<td>2015</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benedetti and Bergamaschi [17]</td>
<td>2016</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Georgala et al. [39]</td>
<td>2016</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Han et al. [41]</td>
<td>2016</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hernandez et al. [42]</td>
<td>2016</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knuth et al. [49]</td>
<td>2016</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rico et al. [72]</td>
<td>2016</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schoenfisch and Stuckenschmidt [84]</td>
<td>2016</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Song et al. [86]</td>
<td>2016</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonifati et al. [21]</td>
<td>2017</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dellal et al. [31]</td>
<td>2017</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fokou et al. [37]</td>
<td>2017</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stegmann and Ziegler [88]</td>
<td>2017</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thakkar et al. [89]</td>
<td>2017</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Akhtar et al. [2]</td>
<td>2018</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonifati et al. [22]</td>
<td>2018</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Darari et al. [29]</td>
<td>2018</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Martens and Trautner [58]</td>
<td>2018</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Salas and Hogan [76]</td>
<td>2018</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saleem et al. [78]</td>
<td>2018</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saleem et al. [80]</td>
<td>2018</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Varga et al. [93]</td>
<td>2018</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Viswanathan et al. [96]</td>
<td>2018</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Akhtar et al. [3]</td>
<td>2019</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cheng and Hartig [26]</td>
<td>2019</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fafalios and Tzitzikas [34]</td>
<td>2019</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fernandez et al. [35]</td>
<td>2019</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Potoniec [69]</td>
<td>2019</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saleem et al. [82]</td>
<td>2019</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thost and Dobly [90]</td>
<td>2019</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wang et al. [98]</td>
<td>2019</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Savali et al. [75]</td>
<td>2019</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Singh et al. [85]</td>
<td>2019</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Azzam et al. [15]</td>
<td>2020</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bigerl et al. [18]</td>
<td>2020</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonifati et al. [24]</td>
<td>2020</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Figueira et al. [36]</td>
<td>2020</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jian et al. [47]</td>
<td>2020</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zhang et al. [101]</td>
<td>2020</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aebeleo et al. [1]</td>
<td>2021</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Almendros-Jimenez et al. [6]</td>
<td>2021</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Azzam et al. [14]</td>
<td>2021</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Davoudian et al. [30]</td>
<td>2021</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Desouki et al. [32]</td>
<td>2021</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Röder et al. [71]</td>
<td>2021</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wang et al. [97]</td>
<td>2021</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
marise the Knowledge Graphs such that they can be more efficiently accessed from and stored on mobile devices with limited resources.

**UC4: Usability** LSQ also has applications for improving the usability of SPARQL endpoints.

- Arenas et al. [11] propose a method for reverse-engineering SPARQL queries, which attempts to construct a query that will return a given set of positive examples as results, but not a second set of negative examples; the authors use LSQ to show that the approach scales well in the data size, number of examples, and in the size of the smallest query that fits the data.

- Benedetti and Bergamaschi [17] present a system (LODeX) that allows users to explore SPARQL endpoints more easily through a formal model defined over the endpoint schema; they show that LODEX is able to generate 77.6% of the 5 million queries contained in the original LSQ dataset.

- Dellal et al. [31] proposes query relaxation methods for queries with empty results, based on finding minimal failing subqueries (generating empty results) and maximal succeeding subqueries (generating non-empty results) to aid the user [37]. The paper refers to LSQ to establish that queries with empty results are common in practice.

- Stegemann and Ziegler [88] propose new operators for the SPARQL language that allow for composing path queries more easily; the authors evaluated their approach with a user study and analysis of the extent to which their language is able to express the real-world queries found in LSQ.

- Viswanathan et al. [96] propose a different form of query relaxation, which generalises a specific resource to a variable on which specific restrictions are added that correspond to relevant characteristics of the resource; they use LSQ to understand how entities are queried in practice.

- Potonie [69] proposes an interactive system for learning SPARQL queries from positive and negative examples; he uses the DBPEDIA queries of LSQ for experiments.

- Wang et al. [98] present an approach for explaining missing results for a SPARQL query—based on answering “why-not” questions that ask why a specific result is not included—to help users refine their initial queries; the authors search LSQ for queries useful for their approach.

- Bonifati et al. [24] analyse “streaks” in DBpedia query logs, where a streak is defined as a sequence of similar queries in chronological order, capturing the idea of a user refining and/or extending an initial query towards a final query.

- Jian et al. [47] use LSQ to evaluate their approach for SPARQL query relaxation (to generalise users’ queries) and query restriction (to refine users’ queries) based on approximation and heuristics.

- Zhang et al. [101] propose a method to model client behaviour when formulating SPARQL queries in order to predict their intent and optimise queries. They use LSQ for their evaluation.


- Wang et al. [97] focus on providing explanations for SPARQL query similarity measures. The authors provide similarity scores using several explainable models based on Linear Regression, Support Vector Regression, Ridge Regression, and Random Forest Regression. They use LSQ to evaluate their query classification.

**UC5: Optimisation** The LSQ dataset can also be used to identify and study fragments that are commonly used in practice and can be evaluated efficiently using dedicated algorithms.

- The aforementioned analyses by Han et al. [41] and Bonifati et al. [21, 22] suggest that well-designed patterns, queries of bounded treewidth, etc., make for promising fragments.

- In the context of probabilistic Ontology-Based Data Access (OBDA), Schoenfisch and Stuckenschmidt [84] analyse the ratio of safe queries—whose evaluation is tractable in data complexity—versus unsafe queries—whose evaluation is #P-hard; they show that over 97.9% of the LSQ queries are safe, and can be efficiently evaluated.

- Song et al. [86] use LSQ to analyse how nested OPTIONAL clauses affect query response times; they propose a way to approximate solutions for deeply-nested well-designed patterns.

- Martens and Trautner [58] later take the property paths extracted by Bonifati et al. [21] from LSQ and other sources, defining simple transitive ex-
pressions that subsume almost all property path
expressions seen in practice, while allowing more
efficient evaluation than the general case.
– Cheng and Hartig [26] introduce a monotonic ver-
sion of the \texttt{OPTIONAL} operator to SPARQL called
\texttt{OPT+}; a possible downside of the operator is an
increase in query result sizes, where they use the
LSQ dataset to study how \texttt{OPTIONAL} and \texttt{OPT+}
behave for real-world queries.
– Building upon the work of Martens and Trautner
[58], Figueira et al. [36] specifically study the
containment problem for restricted classes of Con-
junctive Regular Path Queries (CRPQs), which
are akin to BGPs with property paths; aside from
complexity results, they show the coverage of the
different classes for logs that include LSQ [24].

\textbf{UC6: Meta-Querying} A handful of works have also
used LSQ in the context of meta-querying, where
queries are found based on the resources they contain.
– Rico et al. [72] observe that analogous DB-
pedia properties are often defined in two dis-

tinct namespaces—e.g., \texttt{dbo:birthPlace} and
\texttt{dbp:birthPlace}—where they propose methods
to automatically expand SPARQL queries to cap-
ture solutions involving analogous properties; they
show that only 0.2\% of the DBPedia queries in
LSQ mention properties from both namespaces.
– Varga et al. [93] provide an RDF-based metamodel
for BI 2.0 systems, which allows for capturing the
schema of a dataset, as well as previous queries
that have been posed against that dataset by other
users; the authors propose to re-use parts of the
LSQ vocabulary in their model; they further in-
stantiate their model using LSQ to retrieve queries
asked about countries.

\textbf{Discussion} Per Table 6, we see that the original ver-
sion of LSQ has been used in a wide variety of research
works for a variety of purposes. Complementing other
SPARQL query logs such as Wikidata’s [57], we be-
lieve that LSQ 2.0, with its extended set of queries, will
likewise serve as a useful resource to help align the
theory and practice of SPARQL research.

8. Conclusions and Future Directions

In this paper, we have described the Linked SPARQL
Queries v.2 (LSQ 2.0) dataset, which represents queries
in logs as RDF, allowing clients to quickly find real-
world queries that may be of interest to them. We have
described a number of use cases for LSQ, including
the generation of custom benchmarks, the analysis of how
SPARQL is used in practice, the evaluation of caching
systems, the exploration of techniques to improve the
usability of SPARQL services, the targeted optimisa-
tion of queries with characteristics commonly found
in real workloads, as well as the ability to find queries
relating to specific resources. We then described the

\begin{itemize}
\item Fafalios and Tzitzikas [34] present a query evalu-
\textit{ation strategy}, called SPARQL-LD, that combines
link traversal and query processing at SPARQL
endpoints; they provide a method for checking
if a SPARQL query can be answered through
link traversal, and analyse a large corpus of real
SPARQL query logs—including LSQ—for find-
ing the frequency and distribution of answerable
and non-answerable query patterns; they also use
LSQ to evaluate their approach.
\item Singh et al. [85] use the LSQ vocabulary for pro-
\textit{viding a benchmark} for Question Answering over
Linked Data. The authors use the LSQ vocabulary
to represent the SPARQL query related features
prior to generating the benchmark.
\item Thost and Dolby [90] present QED: a system for
generating concise RDF graphs that are sufficient
to produce solutions from a given query, which can
be used for benchmarking, for compliance testing,
for training query-by-example models, etc.; they
apply their system over LSQ queries to generate
datasets from DBPEDIA.
\item Aebeloe et al. [1] present a decentralised archi-
tecture based on blockchain that allows users to
propose updates to faulty or outdated data, tracing
back their origin, and query older versions of the
data. They use LSQ queries for their evaluation.
\end{itemize}
model and vocabulary used to represent LSQ, including static features of queries, a SPIN representation, provenance encoding the agents and endpoints from which the query originate, as well as runtime statistics generated through local executions of the queries against their corresponding dataset. We then discussed how LSQ is published, thereafter describing the datasets and queries featured in the current version of LSQ. Finally we discussed how LSQ has been used for research purposes since its initial release in 2015.

As discussed in Section 7, since its initial release, LSQ has been adopted by a variety of research works for a variety of purposes. In terms of future directions, we will look to continue adding further logs with further queries to the dataset. Looking at how LSQ has been adopted in the literature has also revealed ways in which the metadata for LSQ could be extended in a future version, such as to add information about monotonicity and satisfiability [41], or information about (hyper)treewidth [21, 22], for example. It may also be useful to provide a canonical version of the query string [76]; this could perhaps be leveraged, for example, when evaluating caching methods. Another useful feature would be to add questions in natural language that verbalise each query, which could be used, for example, in order to create datasets for training and testing question answering systems, as well as enabling users to find relevant queries through keyword search; given the large number of queries in the dataset, an automated approach may be applicable [64].

As discussed by Martens and Trautner [59], query logs allow to bridge the theory and practice of SPARQL. They serve an important role, ensuring that the research conducted by the community is guided by the requirements and trends that emerge in practice. We thus believe that LSQ (2.0) will continue to serve an important role in SPARQL research in the coming years.

Acknowledgements We thank the OpenLink Software team for hosting the DBpedia SPARQL endpoint and for making the logs available to us. Hogan was supported by Fondecyt Grant No. 1181896 and by ANID – Millennium Science Initiative Program – Code ICN17_002. Buit-Aranda was supported by Fondecyt Iniciación Grant No. 11170714 and by ANID – Millennium Science Initiative Program – Code ICN17_002. This work was also partially supported by the German Federal Ministry of Education and Research (BMBF) within the EuroStars project E!114681 3DFed under the grant no 01QE2114, project RAK1 (01MD19012D) and project KnowGraphs (No 860801).

References

M. Dumontier, A. Callahan, J. Cruz-Toledo, P. Ansell, 
A. Davoudian, L. Chen, H. Tu, and M. Liu. A Workload-
A. Clemmer and S. Davies. Smeagol: a “specific-to-general”
Semantic Web interface paradigm for novices. In
O. Caró, H. Naacke, M. A. Bazzini, and B. Amann. HAQWA:
a hash-based and Query Workload Aware Distributed RDF
S. Cheng and O. Hartig. OPT+: A Monotonic Alternative to
A. Clemmer and S. Davies. Smeagol: a “specific-to-general”
Semantic Web interface paradigm for novices. In
O. Caró, H. Naacke, M. A. Bazzini, and B. Amann. HAQWA:
a hash-based and Query Workload Aware Distributed RDF
F. Darani, W. Nutt, G. Pirro, and S. Ruzniewski. Completeness
A. Davoudian, L. Chen, H. Tu, and M. Liu. A Workload-
Adaptive Streaming Partitioner for Distributed Graph Stores.
I. Dellali, S. Jean, A. Hadjiali, B. Chardin, and M. Baron. On
addressing the empty answer problem in uncertain knowledge
M. Dumontier, A. Callahan, J. Cruz-Toledo, P. Ansell, 
V. Emonet, F. Belleau, and A. Droit. Bio2RDF Release 3: A
larger, more connected network of Linked Data for the Life
17 F. Benedetti and S. Bergamaschi. A model for visual building
19 C. Bizer and A. Schultz. The Berlin SPARQL Benchmark. 
21 A. Bonifati, W. Martens, and T. Timm. An Analytical Study of
22 A. Bonifati, W. Martens, and T. Timm. DARQL: Deep Analysis
23 A. Bonifati, W. Martens, and T. Timm. Navigating the Maze of
24 A. Bonifati, W. Martens, and T. Timm. An analytical study of
25 S. Campinhas. Live SPARQL Auto-Completion. In ISWC
26 S. Cheng and O. Hartig. OPT+: A Monotonic Alternative to
27 A. Clemmer and S. Davies. Smeagol: a “specific-to-general”
Semantic Web interface paradigm for novices. In
regular path queries. In International Conference on Princi-
29 G. Fokou, S. Jean, A. Hadijali, and M. Baron. Handling failing
30 R. Frosini, A. Calli, A. Poulouvasilis, and P. T. Wood. Flexible
31 K. Georgala, M. A. Sherif, and A.-C. N. Ngomo. An efficient
32 Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL
33 X. Han, Z. Feng, X. Zhang, X. Wang, G. Rao, and S. Jiang. On
the statistical analysis of practical SPARQL queries. In
35 A. Hogan, M. Mellotte, G. Powell, and D. Stampoulis. Towards
36 F. Hogenboom, V. Mleja, F. Frasincar, and U. Kaymak. RDF-
37 K. Hose and R. Schenkel. WARP: Workload-aware replication
and partitioning for RDF. In Data Engineering meets the Semantic Web (DESWEB@ICDE), pages 1–6. IEEE Computer Society, 2013.
40 H. Knublauch, J. A. Hendler, and K. Idehen. SPiN – Overview
41 M. Knuth, O. Hartig, and H. Suck. Scheduling refresh queries
for keeping results from a SPARQL endpoint up-to-date. In
42 T. Lampo, M. Vidal, J. Danilow, and E. Ruckhaus. To Cache
or Not To Cache: The Effects of Warming Cache in Complex


