Enabling Web-scale Knowledge Graphs Querying

Amr Azzam

Vienna University of Economics and Business, Austria
aazzam@wu.ac.at

Abstract. Knowledge Graphs (KGs) have become an asset for integrating and consuming data from heterogeneous sources. KGs have an influence on several domains such as health-care, manufacturing, transportation and energy. Over the years, the Web of Data has grown significantly. Today, answering complex queries on open KGs is practically impossible due to the SPARQL endpoints availability problem caused by well-known scalability and load balancing issues when hosting Web-size data for concurrent clients. To maintain reliable and responsive open Knowledge Graph query services, several solutions have been proposed: while SPARQL endpoints enforce restrictions on server usage such as imposing limited query execution time or providing partial query results, alternative solutions such as Triple Pattern Fragments (TPF) attempts to tackle the problem of availability by pushing query processing workload to the client-side but suffer from the unnecessary transfer of irrelevant data on complex queries as a result of the large intermediate results. The aim of our research is to develop a new generation of smart clients and servers to balance the load between servers and clients, with the best possible query execution performance, and at the same time reducing data transfer volume, by combining SPARQL endpoints, TPF and shipping compressed KG partitions. The proposed solution shall, on the server-side, offer a suitable query execution service according to the current status of the server workload. On the client-side, we plan research on novel client-side caching mechanisms on the basis of compressed and queryable KG partitions that can be distributed in a modular fashion. In addition, we plan to leverage query logs to optimize the number and the distribution of partitions as well as distributing the query load across a network of collaborative clients.

Keywords: Knowledge Graphs · Availability · Query Performance

1 Introduction

Knowledge Graphs (KGs) have emerged as a rising data management and knowledge representation framework to provide scalable knowledge models that can capture facts about entities as well as relations among these entities. Several implementations of Knowledge Graphs have been introduced in diverse areas such as the pharmaceutical industry [28], IT services, telecommunication, and government [24][26]. The proliferation of the KG concept offers the potential for building creative products and services

that introduce a wide range of commercial applications. For instance, Google's Knowledge Graph [1], Knowledge Vault [11], Microsoft Satori[2] and Facebook's Entities Graph[3].

In addition to these commercial Knowledge Graphs, currently existing open interlinked KGs include DBpedia [4], Yago [20], Wikidata [30] and NELL [10]. These open KGs are typically published following the Linked Data principles [7], using a semi-structured RDF data model and support querying through SPARQL query language. However, there are several open challenges in order to maintain public SPARQL services on the Web, serving multiple concurrent clients.

That is, providing reliable public access to KGs through SPARQL querying is still an open issue due to the unpredictable number of clients executing arbitrary SPARQL queries. To mitigate these availability problems, data providers who expose SPARQL endpoints typically add several constraints on the queries such as limiting the query execution time on the server or limiting the number of retrieved results. Another solution, Triple Pattern Fragments(TPF) [29] provides a more balanced query processing between the client and the server but with the cost of high network traffic. Finally, SaGe [22] introduces a Web preemption mechanism that prevents the long-running queries from consuming the server resources. However, SaGe lacks the ability to handle the potential load of several concurrent complex queries at a time.

The proposed doctoral thesis aims to address the open research questions related to the trade-off between the availability and performance in Web Knowledge Graph interfaces. The main challenge is to provide a Knowledge Graph querying interface that maintains high availability alongside high query execution performance, besides minimizing the data transfer.

2 State of the Art

Overall four orthogonal approaches have been proposed in the literature that enable hosting and querying open Knowledge Graphs that we will describe in the following:

2.1 SPARQL Endpoints

SPARQL endpoints offer SPARQL query interface over Knowledge Graphs. First, The submitted queries are executed on top of a triple store such as Jena TDB [25], Stardog[4] and Virtuoso [12]. Then, the SPARQL query results are shipped via HTTP to the requesting clients [13].

Although current SPARQL endpoints provide a high performance query processing, it requires to run under low query workloads due to the excessive consumption of the long-running queries to the server CPU and memory. Knowledge Graphs (KGs) that are exposed via SPARQL endpoints suffer from well-known problems of low availability and recurrent downtime [9, 29].

[1] https://developers.google.com/knowledge-graph
[2] https://www.microsoft.com/en-us/research/project/knowledge-mining-api/
[3] https://developers.facebook.com/docs/graph-api/
[4] https://www.stardog.com/

In order to tackle these challenges to provide live queryable Knowledge Graphs, SPARQL endpoints with high demands generally introduce a set of usage restrictions to ensure fair utilization of the server resources. For instance, Data providers impose a time quota restriction [3] as DBpedia administrators set a running time quantum of 120 seconds on the server for each submitted query, limit results sizes to 10K results, refuse the complex queries and limit the number of parallel requests per IP.[5]

2.2 Linked Data Fragments

Linked Data Fragments framework (LDF) [29] laid the basis to define Triple Pattern Fragments (TPF) a simple interface that attempts to tackle the problem of availability through providing intelligent TPF clients which shift complex query processing to the client-side, but with the cost of the increased network overhead due to potentially unnecessary transfer of large intermediate results. This can lead to longer query execution time that lowers the overall performance.

To address the drawbacks of TPF, Bindings-Restricted Triple Pattern Fragments [19] (brTPF) is proposed as an extended interface of TPF that gives a slight boost to the performance of the query execution through attaching intermediate results to triple pattern requests together with distributing the join between the client and the server using the bind join strategy [15]. In this manner, brTPF [19] reduces the number of HTTP requests in addition to minimizing the amount of data transferred compared to the ”vanilla” TPF. However, brTPF still would require a potentially high number of HTTP requests. In addition to the shortcoming of the ability to scale with large datasets.

2.3 SaGe

SaGe [22] is a Web preemption based SPARQL query interface designed to avoid the starvation of the simple queries waiting for the complex ones that consume the server resources. SaGe utilized a Round-Robin algorithm to maintain a fair allocation of server resources between queries. To this end, SaGe formalizes a model that enables to suspend and proceed queries with the mechanism to save the state of the query execution to the client for later resumption. Additionally, SaGe has implemented some client-side operators such as ORDER BY, OPTIONAL as well as aggregation functions to execute parts of the query on the client-side.

Experiments show that SaGe enhances the average completion time per client in addition to reducing the average network traffic per client. However, SaGe still extensively consumes the server resources. Besides, the performance of SaGe is degrading with the increasing sizes of the Knowledge Graphs, plus the execution time of concurrent complex queries is potentially increasing significantly.

2.4 Data Dumps

Last, Data Dumps provide a possibility to access Knowledge Graphs through granting access to download KG data thereafter the clients can execute SPARQL queries

[5] https://wiki.dbpedia.org/public-sparql-endpoint

on their local machines. This approach, however, somewhat defeats the vision of live Knowledge Graph querying which is to offer live querying Web data. Furthermore, even if bandwidth to download full data dumps is not considered, their sheer size may be prohibitive in terms of local query processing for clients with limited resources.

3 Problem Statement and Contributions

To maintain querying interfaces on Knowledge Graphs on the Web, state-of-the-art SPARQL query processing techniques can be categorized into three main strategies that are explained in the following:

S1 **Query Shipping:** Knowledge Graphs are exposed for public querying through a full SPARQL endpoint with high query performance but with low server availability. The endpoints are responsible for executing the full SPARQL queries and only retrieve the query results.

S2 **Data Shipping:** To alleviate the low availability problem of **S1**, several client-side solutions such as Squin[17, 18] which perform query execution on the Web through KG traversal. These approaches try to retrieve RDF data that can be processed locally. Unfortunately, the evaluation of complex queries is impractical due to the non-deferenceable URIs besides many non-trivial queries require the full KG dump to be shipped to the client. Hence, These approaches increase the availability of the server yet require strong client machines.

S3 **Hybrid Shipping:** Hybrid shipping approaches attempt to overcome the weaknesses of S1 and S2 through a more balanced client/server distribution such as the aforementioned approaches TPF [29] and SaGe [22]. However, these approaches have several potential issues which were discussed in Section 2.

In this dissertation, we plan to design, build and evaluate a KG interface that distributes the load of query evaluation between clients and servers *by fruitfully combining* data shipping, query shipping and extending the space of hybrid shipping methods, recombining them in novel ways, under the following hypothesis:

Hypothesis: Our hypothesis is that each of the three discussed shipping strategies to KG query services has its pros and cons for different scenarios, query workloads, and concurrency parameters. Therefore, we aim at developing hybrid approaches that combine all three strategies in the most efficient manner, depending on server load, client resources, and potential collaboration among clients.

The main contribution of this dissertation shall, therefore, be to propose an efficient approach to execute SPARQL queries on remote Knowledge Graphs while balancing the trade-off between the high availability of the Knowledge Graph server and the efficient query execution. Generally, we expect to reduce the overall server cost as we enhance the usage of CPU, caching and concurrency.

According to the problem statement, the hypothesis and the proposed contribution, we have derived the following more concrete research questions:

RQ1 How can we achieve significant speedups to the decentralized querying of Knowledge Graphs by developing a novel client/server distribution?
This research question can be further divided into three sub-questions, corresponding to the smart client and server-side respectively:

RQ1.1 Can compressed partitions shipping reduce the load on servers as a novel intermediate solution in between TPF and downloading full dumps?

RQ1.2 How can a (distributed) caching mechanism in smart clients further enhance KG availability?

RQ1.3 How can log analysis help to find trending queries and improve the graph partitioning?

RQ2 How can we build a framework of hybrid server interfaces that dynamically select the appropriate interface based on the given query, client capabilities, and the current server load?

RQ2.1 Which further novel optimization of joins and other operators in a hybrid setting can yield further performance improvements?

RQ2.2 Client collaboration: How can clients - sharing their processing and caching resources - collaboratively improve query processing?

RQ3 How can we build efficient update strategies for the server data (i.e. graph partitions) and the smart client metadata (i.e. discoverability metadata)?

4 Research Methodology and Approach

We divide the research process into the set following tasks, to be carried out for each of the aforementioned RQ's:

T1 Investigation of the state-of-the-art research that is relevant to the identified problem. This includes the study of literature about Web Knowledge Graphs query interfaces, RDF data partitioning, peer to peer query processing, caching mechanisms, join optimization and indexing in the areas of Semantic Web and Databases.

T2 Definition of solutions to the currently existing limitations requires the identification of novel contributions. In addition to providing a concrete prototype implementation for the proposed contributions.

T3 Extensive experimental evaluation for the proposed contributions will be conducted in comparison to state-of-the-art approaches. In addition, the experimental setup will be designed according to the studied research questions based on (new)existing benchmarks and the evaluation criteria.

This research approach has been followed in the contribution related to RQ1, smart-KG, which we first presented in [5], we have already gone through these steps. We identified a gap in terms of processing full dumps on the client-side vs. only shipping part(ition)s of the KG to the client. We have already implemented a prototype for this proposed solution[6] and performed an extensive experimental evaluation following the plan described in Section 5 to compare smart-KG to other state-of-the-art approaches. Am analysis of intermediate results is presented in Section 6 below.

5 Evaluation Plan

In this section, we describe the details of our evaluation plan to compare our proposed approach with state-of-the-art approaches. This particularly includes the choice of suitable baselines, benchmark datasets, query workloads, and evaluation metrics. The goal

[6] https://ai.wu.ac.at/smartkg

of the experimental evaluation is to assess the performance of the implemented solutions to the challenges associated with the formulated research questions by conducting a series of experiments and analyzing the insights. The evaluation plan is explained in the following:

Knowledge Graphs and Query Benchmarks

For the experimental evaluation, we will use synthetic as well as real-world RDF Knowledge Graph datasets of variable sizes.

We use three synthetic datasets from Waterloo SPARQL Diversity Benchmark (WatDiv) [2] which is a recent benchmark that provides a wide spectrum of queries with varying structural characteristics and selectivity classes with sizes of 10M, 100M, and 1B triples. In addition, we will employ the synthetic LUBM data generator to create a dataset of 1.36 billion triples. Moreover, we will use Berlin SPARQL Benchmark V3.1 (BSBM) [8] with three datasets from one up to three million products which will generate three different dataset sizes 350, 700 and 1050 million RDF triples.

Additionally, we use real-world datasets. We will use SPARQL queries from FEASIBLE [27] for RQ1.2 in order to test the optimization with respect to query logs. FEASIBLE is a set of queries that have been generated by real users of DBpedia [21] dataset (v.2015A). Furthermore, we plan to use YAGO2 [20] which is a real dataset extracted from Wikipedia, WordNet, and GeoNames. Finally, Bio2RDF [6] is a life science RDF Knowledge Graph that connects a set of different biological datasets with 4.64 billion. Both YAGO2 and Bio2RDF do not provide benchmark queries, therefore we have reused a set of representative test queries that were created to test the performance of distributed SPARQL query engines [16].

Evaluation Metrics We plan to consider evaluation metrics that provide an insight into the trade-off between server availability, query execution performance, and client resources consumption. Our evaluation considers the following metrics:

- **Number of timeouts:** Number of queries that time out. We set a timeout of 5 min. for WatDiv and 30 min. for DBpedia queries.
- **Average workload completion time per client:** Elapsed time spent by a client executing a workload of queries, measured with the `time` command of Linux.
- **Server/Client Resource Consumption:** We report on CPU usage per core, RAM usage, and network traffic.
- **Average time for the first tuple:** The time for first results (TFFT) for a query is the time between the query starting and the production of the first query results.
- **Average number of requests and data transfer:** the number of requests that the smart client sent to the server to get complete results for a query. In addition, the total transferred data when executing a SPARQL query.
- **The Diefficiency metrics dief@t and dief@k:** Two experiment metrics that are able to capture and evaluate systems that produce incremental results [1].
  dief@t measures the diefficiency of a query engine during the first t time units of query execution. It computes the area under the curve of the answer distribution function until t time unit. In our experiments, we will consider the dief@5 and dief@10 in seconds as a time unit.
  dief@k measures the diefficiency of an engine while producing the first k answers

when executing a query. We compare the performance of the different systems at different answer completeness $k = 25\%, k = 50\%, k = 75\%, k = 100\%$.

6 Intermediate Results

smart-KG is a novel approach that introduces a new paradigm to distribute the query processing between the client and the server through combining shipping compressed Knowledge Graph partitions influenced by characteristic sets [23, 14] with intermediate results shipped using TPF. The experimental evaluation demonstrated that smart-KG outperforms existing approaches in server resource usage in addition to the average workload execution time as well as fewer timeout queries under highly concurrent query workloads. On the other hand, SPARQL endpoints and SaGe have a better performance than smart-KG with less number of clients and small-scale Knowledge Graphs. That is, although smart-KG has better average workload execution time, TPF and SaGe outperform smart-KG in certain types of queries.

In our recent research [5], which we briefly described above, we have investigated RQ1 and especially the question RQ1.1. In this research, we introduced a novel paradigm, smart-KG, to balance the load of evaluating SPARQL queries on Web Knowledge Graphs by leveraging shipping compressed KG partitions. We presented a KG partitioning technique named *Family-Based Partitioning* which is, based on *characteristic sets* [23, 14] as an initial partition heuristics, designed to combine the set of predicates are shared between subjects of the same type. Family-Based Partitioning allowed us to have descent KG partitions to be shipped over the Web.

Our empirical evaluation showed that smart-KG has significantly outperformed the state-of-the-art server- and client-side Knowledge Graphs query engines. In our study [5], we reported the performance of smart-KG in comparison to the currently existing approaches SPARQL endpoints represented by Virtuoso, Triple Pattern Fragments (TPF) and SaGe on three sizes of the synthetic dataset Watdiv on a benchmark query workload [19]; plus, we tested the performance of the systems on the real-world DBpedia [21] dataset (v.2015A).

As shown in Figures 1 and 2, smart-KG outperformed query performance of the compared systems at increasing number of clients and variant dataset sizes. smart-KG has no timeout queries in WatDiv-100M workload at different numbers of concurrent clients (1, 10, 20, 40 and 80). Moreover, smart-KG showed a superior average workload execution time per client compared to the other systems specifically with more than 20 concurrent clients. We should emphasize that in our experiments so far, going up to 80 clients, as Figure 1 shows, we did not yet manage to stress smart-KG: on the server-side, where it more or less still showed almost constant effort by client. It could be expected that this behavior degrades at even higher client numbers, which we plan to investigate in the future.

We also plan to re-assess our results from [5] wrt. the proposed metrics in Section 5 in more setups and analyze which types of queries, datasets and setups favor smart-KG with respect to other approaches.

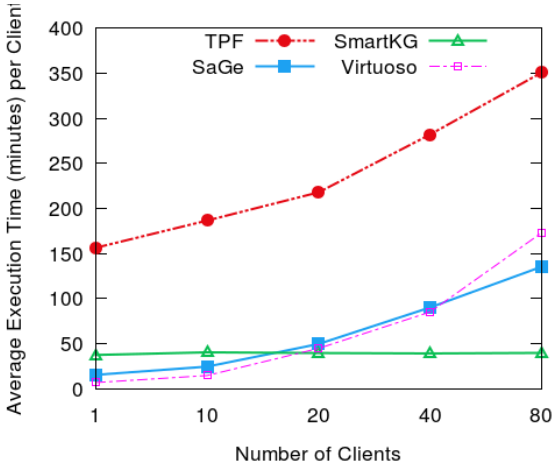

Fig. 1. Average execution time on Watdiv-100M, from [5]

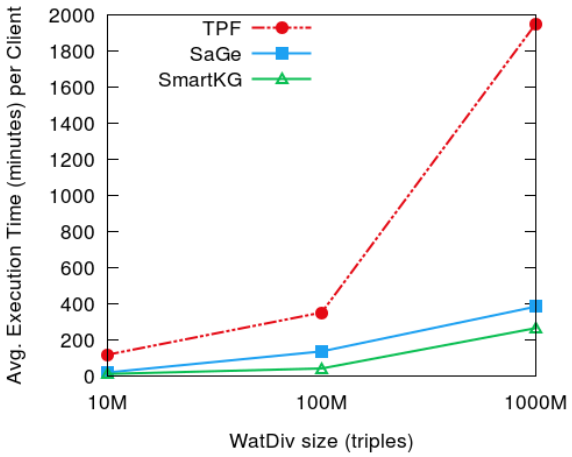

Fig. 2. Performance on the workloads (80 clients) at increasing KG sizes, from [5]

7 Conclusions and Lessons Learned

In this doctoral work, we aim to tackle the lack of reliable live public querying to Knowledge Graphs on the Web. We have formulated 3 main research questions that aim to democratize the access to Knowledge Graphs by enabling Web-scale SPARQL querying. Our intermediate results on RQ1 provide an insight into how shipping compressed graph partitions that can be locally queried could balance the load between servers and clients. Our empirical results demonstrate significant improvements in server availability with enhanced query performance.

Our current work addressing RQ1 will investigate further other partitioning strategies that could provide a reasonable trade-off of shipping sizes. In addition, we plan to explore the space of query-driven partitions through analyzing the Knowledge Graphs

query logs so that we could achieve the promised balancing between efficient query execution and the availability of the public services.

Thereafter, we intend to determine suitable heuristics for a cost model in RQ2.1 in order to explore the space of feasible query plans so that the proposed framework could find the best query execution plan based on the server and the client available resources.

Lastly, we plan to address RQ2.2. We will explore building a peer-to-peer collaborative smart clients network in order to enhance the server availability through sharing the shipped graph partitions rather than downloading it from the KG server. This will lead us to a decentralized architecture for KG querying.

As for RQ3, we intend to explore novel update strategies to the compressed graph partitions in order to avoid the overhead of the partitions regeneration in case of evolving Knowledge Graphs.

Acknowledgments

This work has been supported by the European Union Horizon 2020 research and innovation programme under grant 731601 (SPECIAL) and by the Austrian Research Promotion Agency (FFG) grant no. 861213 (CitySPIN). I thank my doctoral supervisor Prof. Dr. Axel Polleres and my co-authors Dr. Javier D. Fernández and Dr. Maribel Acosta and Martin Beno for their helpful discussions, comments and feedback.

References

1. M. Acosta, M.-E. Vidal, and Y. Sure-Vetter. Diefficiency metrics: Measuring the continuous efficiency of query processing approaches. pages 3–19, 10 2017.
2. G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified stress testing of rdf data management systems. volume 8796, pages 197–212, 10 2014.
3. C. Aranda, A. Hogan, J. Umbrich, and P.-Y. Vandenbussche. Sparql web-querying infrastructure: Ready for action? *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference*, pages 277–293, 01 2013.
4. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. volume 6, pages 722–735, 01 2007.
5. A. Azzam, J. D. Fernández, M. Acosta, M. Beno, and A. Polleres. Smart-kg: Hybrid shipping for sparql querying on the web. In *Proceedings of The Web Conference 2020*, WWW '20, page 984–994, New York, NY, USA, 2020. Association for Computing Machinery.
6. F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault, and J. Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge system. *Journal of biomedical informatics*, 41:706–16, 04 2008.
7. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
8. C. Bizer and A. Schultz. The berlin sparql benchmark. *Int. J. Semantic Web Inf. Syst.*, 5:1–24, 04 2009.
9. C. Buil-Aranda, A. Hogan, J. Umbrich, and P.-Y. Vandenbussche. Sparql web-querying infrastructure: Ready for action? In H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. Noy, C. Welty, and K. Janowicz, editors, *The Semantic Web – ISWC 2013*, pages 277–293, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

10. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
11. X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610, 2014. Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Geremy Heitz.
12. O. Erling and I. Mikhailov. RDF support in the Virtuoso DBMS. In *Networked Knowledge-Networked Media*, pages 7–24. Springer, 2009.
13. L. Feigenbaum, G. T. Williams, K. G. Clark, and E. Torres. SPARQL 1.1 protocol. *Recommendation, W3C, March*, 2013.
14. A. Gubichev and T. Neumann. Exploiting the query structure for efficient join ordering in SPARQL queries. In *EDBT*, volume 14, pages 439–450, 2014.
15. L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *VLDB*, 1997.
16. R. Harbi, I. Abdelaziz, P. Kalnis, N. Mamoulis, Y. Ebrahim, and M. Sahli. Accelerating SPARQL queries by exploiting hash-based locality and adaptive partitioning. *The VLDB Journal*, 25(3):355–380, June 2016.
17. O. Hartig. Squin: a traversal based query execution system for the web of linked data. In *Proc. of SIGMOD*, pages 1081–1084. ACM, 2013.
18. O. Hartig, C. Bizer, and J. C. Freytag. Executing SPARQL queries over the web of linked data. In *Proc. of ISWC*, pages 293–309, 2009.
19. O. Hartig and C. Buil-Aranda. Bindings-restricted triple pattern fragments. In *Proc. of ODBASE*, volume 10033 of *LNCS*, pages 762–779, 10 2016.
20. J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28 – 61, 2013. Artificial Intelligence, Wikipedia and Semi-Structured Resources.
21. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.
22. T. Minier, H. Skaf-Molli, and P. Molli. Sage: Web preemption for public SPARQL query services. In *The Web Conference*, pages 1268–1278. ACM, 2019.
23. T. Neumann and G. Moerkotte. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *Proc. of ICDE*, pages 984–994. IEEE, 2011.
24. N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM*, 62 (8):36–43, 2019.
25. A. Owens, A. Seaborne, N. Gibbins, and mc schraefel. Clustered tdb: A clustered triple store for jena. Project report, November 2008.
26. S. Ronzhin, E. Folmer, Maria, Brattinga, W. Beek, Lemmens, and v. Veer. Kadaster knowledge graph: Beyond the fifth star of open data. *Information*, 10:310, 10 2019.
27. M. Saleem, Q. Mehmood, and A.-C. Ngonga Ngomo. Feasible: A feature-based sparql benchmark generation framework. In *International Semantic Web Conference (ISWC)*, 2015.
28. F. Shen and Y. Lee. Knowledge discovery from biomedical ontologies in cross domains. *PloS one*, 11:e0160005, 08 2016.
29. R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, and P. Colpaert. Triple Pattern Fragments: a low-cost knowledge graph interface for the Web. *Journal of Web Semantics*, 37–38:184–206, Mar. 2016.
30. D. Vrandečić and M. Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57:78–85, 09 2014.