Getting and hosting your own copy of Wikidata

Wolfgang Fahl[1], Tim Holzheim[1], Andrea Westerinen[2], Christoph Lange[1,3] and Stefan Decker[1,3]

[1] *RWTH Aachen University, Computer Science i5, Aachen, Germany*
[2] *OntoInsights LLC, Elkton, MD, United States*
[3] *Fraunhofer FIT, Sankt Augustin, Germany*

Abstract

Wikidata is a very large, crowd sourced, general knowledge graph that is backed by a worldwide community. Its original purpose was to link different versions of Wikipedia articles across multiple languages. Access to Wikidata is provided by the non-profit Wikimedia Foundation and recently also by Wikimedia Enterprise as a commercial service. The query access via the public Wikidata Query Service (WDQS) has limits that make larger queries with millions of results next to impossible, due to a one minute timeout restriction. Beyond addressing the timeout restriction, hosting a copy of Wikidata may be desirable in order to have a more reliable service, quicker response times, less user load, and better control over the infrastructure.

It is not easy, but it is possible to get and host your own copy of Wikidata. The data and software needed to run a complete Wikidata instance are available as open source or accessible via free licenses.

In this paper, we report on both successful and failed attempts to get and host your own copy of Wikidata, using different triple store servers.

We share recommendations for the needed hardware and software, provide documented scripts to semi-automate the procedures, and document things to avoid.

Keywords

Wikidata, RDF Bulk Import, SPARQL

1. Introduction

In recent years Wikidata [1] has gained increasing visibility and value as an open knowledge graph. Accessing its information is constrained, however, since the resources behind the Wikidata Query Service (WDQS)[1] are limited and used by many people and services (such as bots that scan or update Wikidata automatically). In addition, due to the size of Wikidata (∼17B triples), even a simple query can hit the WDQS query timeout limit[2].

Public alternatives to WDQS are (to our knowledge) rare. We only know of the three endpoints shown in Table 1. And, these alternatives have their own limitations. The QLever[3] Wikidata endpoint has still limitations in the SPARQL language features that are accepted. The Virtuoso[4]

Wikidata'22: Wikidata workshop at ISWC 2022
0000-0002-0821-6995 (W. Fahl); 0000-0003-2533-6363 (T. Holzheim); 0000-0002-8589-5573 (A. Westerinen); 0000-0001-9879-3827 (C. Lange); 0000-0001-6324-7164 (S. Decker)

[1] https://query.wikidata.org/
[2] https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/query_limits
[3] https://qlever.cs.uni-freiburg.de/wikidata
[4] https://wikidata.demo.openlinksw.com/sparql

Table 1
Public Wikidata query endpoints

Provider	Triple Store	URL
University of Freiburg	QLever	https://qlever.cs.uni-freiburg.de/wikidata
OpenLink Software	Virtuoso	https://wikidata.demo.openlinksw.com/sparql
Wikimedia Foundation	Blazegraph	https://query.wikidata.org/

endpoint provided by OpenLink Software has data that is many months old.

A key decision factor for getting and hosting your own copy of Wikidata is the cost-benefit relation. The cost includes the infrastructure plus the effort to get, host and update the data. The benefit depends on the use cases and what Wikidata content your use cases need - see section 1.1 below.

The current public Wikidata Query Service infrastructure uses Blazegraph [2]. That implementation is based on open-source software that is no longer in development and is approaching execution limitations. As stated in a Wikimedia Foundation Blog entry "Blazegraph maxing out in size poses the greatest risk for catastrophic failure, as it would effectively prevent WDQS from being updated further [...]" [3] and therefore, the Wikimedia Foundation is currently investigating various stop-gap and replacement strategies [4]. Having second source options to host a complete copy of Wikidata in a reproducible way is an important alternative, especially for the transition phase of the upcoming years.

1.1. Content of Wikidata

Wikidata originally started as a knowledge graph for the international Wikipedia encyclopedias. From there it evolved to a folksonomy-style [5] general knowledge graph.

The Wikidata statistics web page[5] shows a pie chart of the entity types having the largest number of instances, but the data is already 2 years old, at the time of this writing. Khatun [6] provided a table of the top 50 "subgraphs" - which are essential the entity classes - of Wikidata and corresponding charts. A more current statistic based on our Stardog[6] import of 2022-06 is shown in Figure 1.

If you are interested in a large number of Wikidata items, then there is a higher probability that having your own copy of Wikidata will be beneficial. Our use case in the ConfIDent project [7] revolves around the entities, academic conference, scientific event series, scholarly article, proceedings and author, which constitute a substantial portion of the Wikidata knowledge graph. Since essential queries for this project where not possible via the publicly available endpoints, we decided to get and host our own copies of Wikidata.

1.2. Size of Wikidata as a moving target

Figure 2 shows how the size of Wikidata dumps has grown substantially over the 2018-2022 time frame. The “Triples” column in Table 3 indicates the difference in import size. From the

[5]https://www.wikidata.org/wiki/Wikidata:Statistics
[6]https://www.stardog.com/

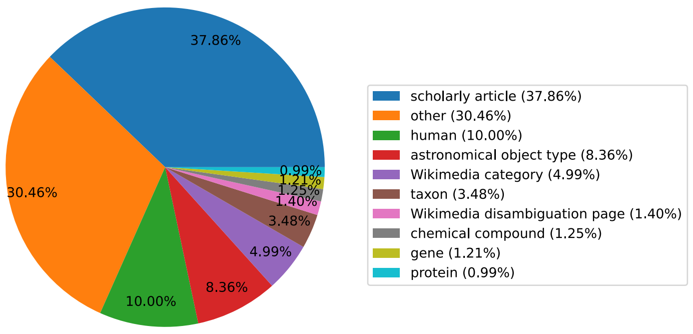

Figure 1: Wikidata instanceof/P31 statistics 2022-07

first reported success in 2017 to the most recent in 2022-07, Wikidata has grown by more than a factor 5. Therefore the reproducibility and comparability of the results are limited.

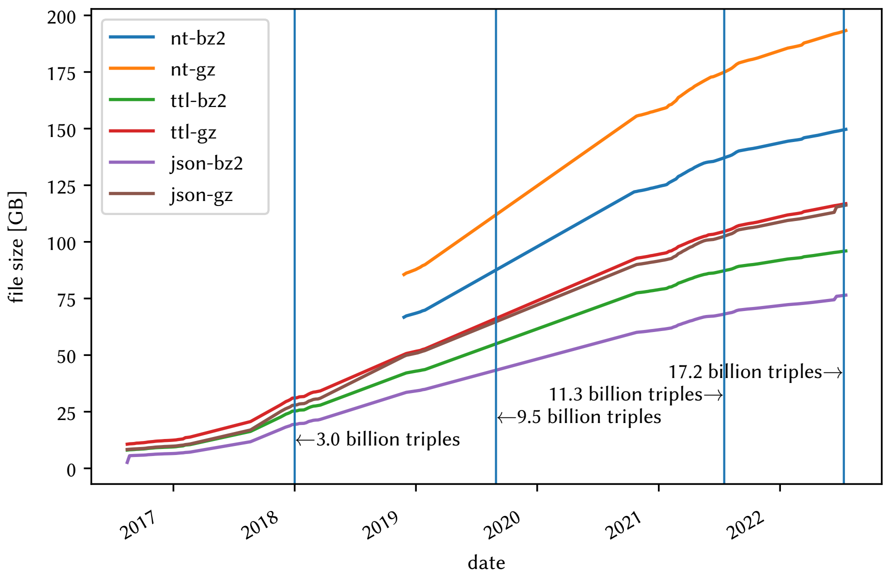

Figure 2: Evolution of the Wikidata dump file size by format and compression

1.3. Getting and hosting your own copy of Wikidata

If you intend to get and host your own copy of Wikidata, the following aspects need to be considered:

Reliability of the import - Will you actually get a running system in the end or run into some kind of software or hardware limit you didn't foresee?

Needed Resources - What kind of computer will you need - e.g. how much RAM and disk space are needed? What kind and version of operating system are needed? Can the machine be hosted virtually, and if so how much will it cost? How much time will the import take?

Usefulness of results - How useful will the result be for your use case? Will you get better performing queries? Do you need “always current” data? Will your own Wikidata server be compatible with the original one? Will the infrastructure be more reliable than the public endpoints?

Consistency How to keep your copy in sync with Wikidata? Integration of the wikidata update stream[7] into the knowledge graph and how to handle update interrupts? Possibility of publishing updates from the copy to Wikidata.

2. Materials and Methods

2.1. Wikidata copy bulk import procedure

Höfler’s [8] StackExchange question and Malyshev’s [9] "getting started" guide were the basis of our first attempt to get and host a complete copy of Wikidata in January 2018. The motivation was to run Apache Gremlin[8] queries on Wikidata. [9]

Table 2 shows the different triple stores we considered for testing the import and setup procedure.

General steps for getting a complete Wikidata copy

- Procure the hardware and software for the indexing and hosting environment (which might be two different computers)
- Download the current Wikidata dump
- Install the triple store software
- Configure the triple store
- Optionally preprocess the triples
- Run the bulk import / indexing procedure
- Optionally copy the results from the indexing machine to a target machine
- Start the server
- Optionally start a separate GUI webserver

[7]https://www.wikidata.org/wiki/Wikidata:Data_access#Recent_Changes_stream
[8]https://github.com/blazegraph/tinkerpop3
[9]Unfortunately we never got this working since we couldn’t connect the Blazegraph endpoint to the gremlin infrastructure.

Table 2
Candidate RDF Triple stores for hosting a full copy of Wikidata

Name	Homepage	Version	First Attempt
Apache Jena	https://jena.apache.org/	4.3.1	2020
Allegro Graph	https://allegrograph.com/	7.3	planned
Blazegraph	https://blazegraph.com/	2.1.5	2018
QLever	https://qlever.cs.uni-freiburg.de/wikidata	2022-01	2022-01
RDF4J	https://rdf4j.org/		planned
Stardog	https://www.stardog.com	8.0	2022-07
Virtuoso	https://virtuoso.openlinksw.com/	0.7.20	planned

Table 3
Success Reports

Date	Source	Target	Triples	Load Time	Reference
2017-12	latest-truthy.nt.gz	Apache Jena	?	8 h	[11]
2018-01	wikidata-20180101-all-BETA.ttl	Blazegraph	3 billion	4 d	[10]
2018-05	latest-all.json.gz	dgraph	?	4 d	[12]
2019-02	latest-all.ttl.gz	Apache Jena	?	2 d	[13]
2019-05	wikidata-20190513-all-BETA.ttl	Blazegraph	?	10.2 d	[14]
2019-05	wikidata-20190513-all-BETA.ttl	Virtuoso	?	43 h	-
2019-09	latest-all.ttl (2019-09)	Virtuoso	9.5 billion	9.1 h	[15]
2019-10		Blazegraph	~10 billion	5.5 d	[16]
2020-03	latest-all.nt.bz2 (2020-03-01	Virtuoso	~11.8 billion	10 h + 1 d prep	[17]
2020-06	latest-all.ttl (2020-04-28)	Apache Jena	12.9 billion	6 d 16 h	[18]
2020-07	latest-truthy.nt (2020-07-15)	Apache Jena	5.2 billion	4 d 14 h	[19]
2020-08	latest-all.nt (2020-08-15)	Apache Jena	13.8 billion	9 d 21 h	[20]
2022-02	latest-all.nt (2022-01-29)	QLever	16.9 billion	4 d 2 h	[21]
2022-02	latest-all.nt (2022-02)	Stardog	16.7 billion	9 h	[22]
2022-05	latest-all.ttl.bz2 (2022-05-29)	QLever	~17 billion	14 h	[23]
2022-06	latest-all.nt (2022-06-25)	QLever	17.2 billion	1 d 2 h	[24]
2022-07	latest-all.ttl (2022-07-12)	Stardog	17.2 billion	1 d 19 h	[25]

3. Results

3.1. Wikidata Imports

Table 3 shows the collection of reports of successful Wikidata imports. Also see the "Success Reports" section [10] describing our import attempts. In that section, we describe our experiences with the candidate RDF triple stores shown in table 2, cross referenced with the success reports in chronological order.

Blazegraph Blazegraph as a software package is quite simple to install. See the setting up Blazegraph[10] section in the SPARQL tutorial by the main author of this work. The Wikimedia Foundation also provides details on setting up Blazegraph[11].

The "getting started" procedure by Malyshev [9] was followed by us in 2018 on an Ubuntu 18.04 LTS machine (later upgraded to 20.04 LTS) and wasn't successful on the first attempt.

[10]https://wiki.bitplan.com/index.php/SPARQL
[11]https://wikitech.wikimedia.org/wiki/Wikidata_Query_Service

We had to use an SSD disk instead of a rotating disk to improve the preparation speed of the munge script that does the preprocessing of the triples. After successful import, the endpoint has been running reliably since 2018 and only needs an occasional server restart (usually after the software crashes on a query that pushes the hardware limits of 64 GB RAM).

The automation of the procedure is quite poor - there are several manual configuration steps necessary. Note that we did not bother to document an attempt that took 4 days! The hardware cost was some 100 EUR for a used server and 140 EUR for a 480 GB SSD.

Apache Jena/Fuseki Based on Andy Seaborn's [11] success report of 2017 we performed seven attempts from 2020-04 until 2020-08, when we could document a success [20]. The problems encountered have been documented on our wiki and include references to the Apache Jena tdbloader software issues that were required to be fixed. The script wikidata2jena on the Success Report web site has a fully automated procedure that worked at the time of the success.

Figure 3 shows the non linear index speed of the tdbloader2 bulk import. The non-linearity is especially bad when using a rotating disk. This is due to the B-Tree structure being applied by the Jena triple store. The needed balancing operations need to move data which causes head movements which are notoriously slow on rotating media. The few milliseconds needed for each move would have added up to almost half a year of indexing time. The remedy was to use a 4 TB SSD where the non linearity slowed down the processing speed from 100 k triples/second to a reasonable 13 k triples/second. The final import statistic was: Time = 395.522,378 seconds : Triples = 5.253.753.313 : Rate = 13.283 /s

The 4 TB SSD cost some 700 EUR. The indexing server was a 1000 EUR used Apple Mac Pro 2012/12 core with 64 GB of RAM.

The 4 TB SSD was moved to a publicly visible endpoint of the RWTH Aachen i5 that is access protected and has been operating reliably. Restarts are necessary when the 6 GB RAM allocation limit of the server has been hit.

QLever A series of import attempts were performed for QLever between 2022-02 and 2022-07. Hannah Bast's [23] import results are publicly available. Our import attempts were mostly performed on a used 128 GB RAM Server using Ubuntu 20.04 LTS. We also tried using the Apple Mac Pro 2012 machine. Our results varied depending on whether we were working from a docker image or using a native build. Two successful indexing attempts are documented on our wiki.

The procedure is fully automated using the QLever script[12] and alternatively the "official" qlever-control[13] that was inspired by the need to make the import procedure repeatable. The import took 9 h in Bast's attempt and 22 h for the fastest import on our own machine.

The public server has been quite reliable while our own machine has had a sequence of problems that we documented on our wiki and in the issues of the QLever github

[12]https://wiki.bitplan.com/index.php/QLever/script
[13]https://github.com/ad-freiburg/qlever-control

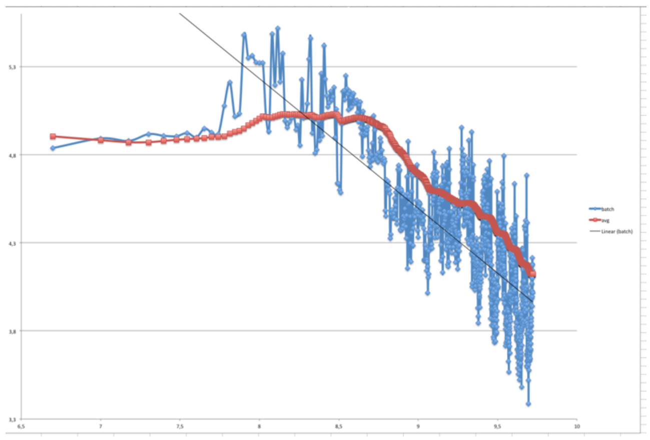

Figure 3: Non Linear Indexer performance of Apache Jena

repository[14]. The cost for the used server was 400 EUR and the cost for another 4 TB SSD had fallen to 350 EUR by this time.

Stardog Based on Evren Sirin's [22] blog entry, we performed an import attempt in 2022-07. The import was run on a AWS EC2 instance with 253 GB RAM and 2TB SSD using Stardog 8.0. For the import the ttl.bz2 dump from 2022-07-12 was used and it took one day and 19 hours to complete with a loading speed of 109.5K triples/sec [25]. The slow loading time can be compared to Sirin's results of 9h using Stardog 7.9. The differences in time might be due to some preprocessing [15] that was performed, taking additionally multiple hours, and the selected dump format. The results suggest that the n-tuples format allows faster loading times due to better parallelization. After the configuration of the server, instance the import was started with a single command.

With a cost of around 50 EUR per day we switched from an AWS instance to a self hosted server with 130GB RAM. The server has running reliably for three weeks as of 2022-08.

Virtuoso The result of Hugh Williams's [17] Wikidata import of 2020 is publicly available as a Virtuoso SPARQL endpoint described in Table 1. Our own attempt for import has not been finished yet since the needed hardware was not procured in time.

Allegro Graph Based on Craig Norvell [26] report, we intend to attempt our own import as soon as the needed hardware is procured.

3.2. Usefulness

The Wikidata Graph Pattern Benchmark [27] has been used to check the basic functionality and performance of the successful imports in comparison with public endpoints. The benchmark consists of 850 queries and covers different aspects of graph patterns. Figure 4 shows the median

[14]https://github.com/ad-freiburg/qlever
[15]starting with Stardog 7.9 this preprocessing step of partitioning the dump file is no longer required

execution time of the 850 queries of the benchmark after 10 iterations on each endpoint. This is only anecdotal evidence for the performance given the difference is in the number of triples and the hardware. All queries except 2 ran successfully at least once on each endpoint [16].

The compatibility of the servers with Wikidata and the kind of queries that are possible differs widely from endpoint to endpoint.

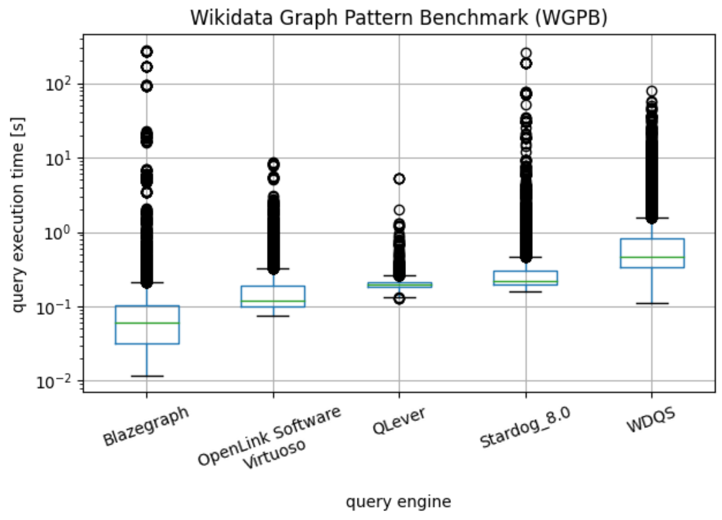

Figure 4: Wikidata Graph Pattern Benchmark (WGPB) results for different query engines.

4. Related Work

In addition to the success reports in Table 3, there has been much other Wikidata-related work performed across the industry. The following list describes this work.

Wikidata imports Höfler [8] asked what the procedure for importing a complete copy of Wikidata was on StackExchange in 2013. Malyshev [9] provided an official "Getting started" guide to host your own copy of Wikidata in 2015. The Wikimedia Foundation [28] provides a web page defining how to create a comparable Blazegraph Wikidata implementation.

Table 3 references the successful Wikidata imports of which we are aware.

Hernández et al. [29] compare the import of a 2016 Wikidata dump with 18 million entities into Blazegraph 2.1.0, Virtuoso 7.2.3, PostgreSQL 9.1.20 and Neo4J-community-2.3.1 regarding the performance of queries.

Kovács et al. [30] used the same 2016 Wikidata dump as [29] and reported on an import into Neo4J 3.3.3, Blazegraph 2.1.4 and JanusGraph 0.2.0.

[16]WGPB query J4 in line 38 has a syntax error and thus can not run successfully on any endpoint

Proposals to avoid using your own copy of Wikidata Minier et al. [31] propose mitigating the SPARQL query quota problem of service providers by splitting queries and running the individual sub-queries within the quotas. This workaround will still not create reliable and timely results if a service completely fails or is under heavy load. Also, splitting the queries requires complex analysis and processing of intermediate result sets for the heavy load use cases that are the motivation for our work.

Henselmann and Harth [32] propose an algorithm for constructing a subset of the Wikidata knowledge graph on demand. The implementation likely requires a copy of Wikidata to be able to run. This defeats the purpose of the idea if the implementation would not be provided as a service.

Aimonier and Davat [33] propose using the HyperLogLog++ algorithm to estimate cardinalities for COUNT-DISTINCT queries. This only solves part of the general problem and requires the installation and execution of a separate infrastructure.

Chalupksy [34] proposes using the Knowledge Graph Toolkit KGTK[17] to import the Wikidata dump and be able to query the result locally on a machine with low resources (laptop). [18]

Data Quality Färber defines 34 data quality metrics and analyses five knowledge graphs, Freebase, DBpedia, OpenCyc, Wikidata and YAGO, against these metrics [35]. The reproducibility of the results is limited since the work was done in 2015 when Wikidata had less than 20 million items.

5. Conclusion

The procedure for getting and hosting your own copy of Wikidata is a moving target and is not well defined, automated or repeatable. Comparison of key features such as reliability, needed resources and usefulness is based on anecdotal evidence at this time.

The download and indexing time for a round trip update is currently a full day, even with a robust hardware environment. For personal use, this is already prohibitive.

5.1. Future work

Improving the technical exchange related to successfully hosting a copy of Wikidata would be valuable to remedy the current lack of definition, automation and repeatability of the procedures. Having sound scientific performance and usefulness analyses would benefit all parties needing access to a reliable and performant Wikidata knowledge graph (that is not as limited as the current public offerings). Specifically, implementing the metric analysis according to [35] in a dashboard would be a valuable contribution.

The search for a Blazegraph alternative by the Wikimedia Foundation [4] has already provided valuable analyses that could become the basis for a general benchmark for Wikidata content

[17]https://github.com/usc-isi-i2/kgtk
[18]see documentation at https://kgtk.readthedocs.io/en/latest/import/import_wikidata/

platforms. However, the Foundation is limiting its exploration of alternatives to only open source solutions.

Hernández [29] pointed out that "testing all of these combinations of features in a systematic way would require extensive experiments outside the current scope" and indeed such extensive experiments would be valuable.

There are also other new graph databases and other approaches that should be considered for their viability especially regarding scalability and distributability. RDF4J Release 4 [36] with a new embedded triple store should also be investigated. Azure CosmosDB, HyperGraphDB and GraKn have already been mentioned in [30] as planned analysis targets. Dgraph was already targeted in 2018 by the success story [12].

Lastly, the problem of keeping a Wikidata implementation “current” must be addressed. This is difficult since data changes are frequent, while Wikidata RDF dumps are released weekly. Both a Kafka- and HTTP API-based solution is documented Wikidata query service updater evolution[19]. This needs to be further refined and provided as a service. It is valuable to note that the current updater used by the Wikimedia Foundation is not publicly available.

6. Acknowledgements

We thank all reporters of successful Wikidata imports for their valuable contribution. We thank Federico Bonelli for sponsoring the AWS instance used for the Stardog import.

This research has been partly funded by a grant of the Deutsche Forschungsgemeinschaft (DFG). [20]

References

[1] D. Vrandečić, M. Krötzsch, Wikidata, Communications of the ACM 57 (2014) 78–85. doi:10.1145/2629489.

[2] B. Bebee, Blazegraph™ db - ultra high-performance graph database supporting blueprints and rdf/sparql api, 2015. URL: https://blazegraph.com/.

[3] W. Search, WMDE, Wikidata:sparql query service/wdqs backend update/august 2021 scaling update, 2021. URL: https://m.wikidata.org/wiki/Wikidata:SPARQL_query_service/WDQS_backend_update/August_2021_scaling_update.

[4] M. Pham, G. Lederrey, A. Westerinen, et al., Wikidata query service backend update, 2022. URL: https://www.wikidata.org/wiki/Category:WDQS_backend_update.

[5] T. V. Wal, Folksonomy coinage and definition, 2007. URL: https://www.vanderwal.net/folksonomy.html.

[6] A. Khatun, Table of top 50 subgraph information, 2021. URL: https://wikitech.wikimedia.org/wiki/User:AKhatun/Wikidata_Subgraph_Analysis#Table_of_top_50_subgraph_information.

[7] T. I. T. Hannover, 2020. URL: https://projects.tib.eu/en/confident/.

[19] https://addshore.com/2022/04/wikidata-query-service-updater-evolution/

[20] ConfIDent project; see https://gepris.dfg.de/gepris/projekt/426477583

[8] P. Höfler, How can i download the complete wikidata database, 2013. URL: https://opendata.stackexchange.com/questions/107/how-can-i-download-the-complete-wikidata-database.

[9] S. Malyshev, Getting started, 2015. URL: https://github.com/wikimedia/wikidata-query-rdf/blob/master/docs/getting-started.md.

[10] W. Fahl, T. Holzheim, Get your own copy of wikidata, 2020. URL: https://wiki.bitplan.com/index.php/Get_your_own_copy_of_WikiData.

[11] A. Seaborne, Report on loading wikidata, 2017. URL: https://lists.apache.org/thread/m8jjmbckm4c31gcl73dl30z6m6jpzj5o.

[12] topicseed, Importing wikidata dumps — the easy part, 2018. URL: https://topicseed.com/blog/importing-wikidata-dumps/.

[13] C. Capol, Wikidata import in apache jena, 2019. URL: https://muncca.com/2019/02/14/wikidata-import-in-apache-jena/.

[14] A. Sanchez, [wikidata] minimal hardware requirements for loading wikidata dump in blazegraph, 2019. URL: https://lists.wikimedia.org/hyperkitty/list/wikidata@lists.wikimedia.org/message/3FIVDPUGNPGDUURWSDPYQG4W6DN2I2RR/.

[15] A. Sanchez, Virtuoso hosted wikidata instance, 2019. URL: https://lists.wikimedia.org/hyperkitty/list/wikidata@lists.wikimedia.org/message/CKN3QPV2NJ5TAKHORMYDTTXG7Y65UXAF/.

[16] A. Shoreland, Your own wikidata query service, with no limits, 2019. URL: https://addshore.com/2019/10/your-own-wikidata-query-service-with-no-limits/.

[17] H. Williams, Loading wikidata into virtuoso (open source or enterprise edition), 2020. URL: https://community.openlinksw.com/t/loading-wikidata-into-virtuoso-open-source-or-enterprise-edition/2717.

[18] J. Sourlier, Jena issue 1909 - report of a wikidata import with jena, 2020. URL: https://issues.apache.org/jira/browse/JENA-1909.

[19] W. Fahl, Wikidata import 2020-07-15, 2020. URL: https://wiki.bitplan.com/index.php/WikiData_Import_2020-07-15.

[20] W. Fahl, Wikidata import 2020-08-15, 2020. URL: https://wiki.bitplan.com/index.php/WikiData_Import_2020-08-15.

[21] W. Fahl, Wikidata import 2022-01-29, 2022. URL: https://wiki.bitplan.com/index.php/WikiData_Import_2022-01-29.

[22] P. K. Evren Sirin, Wikidata in stardog, 2022. URL: https://www.stardog.com/labs/blog/wikidata-in-stardog.

[23] H. Bast, Using qlever for wikidata, 2022. URL: https://github.com/ad-freiburg/qlever/wiki/Using-QLever-for-Wikidata.

[24] W. Fahl, Wikidata import 2022-06-25, 2022. URL: https://wiki.bitplan.com/index.php/WikiData_Import_2022-06-25.

[25] T. Holzheim, W. Fahl, Wikidata import 2022-07-12, 2022. URL: https://wiki.bitplan.com/index.php/WikiData_Import_2022-07-12.

[26] C. Norvell, Wikidata on allegrograph, 2022. URL: https://wiki.bitplan.com/index.php/Wikidata_on_Allegrograph.

[27] A. Hogan, C. Riveros, C. Rojas, A. Soto, Wikidata Graph Pattern Benchmark (WGPB) for RDF/SPARQL, 2019. URL: https://doi.org/10.5281/zenodo.4035223. doi:10.5281/zenodo.

4035223.

[28] Wikimedia, Wikidata query service, 2017. URL: https://wikitech.wikimedia.org/wiki/Wikidata_Query_Service.

[29] D. Hernández, A. Hogan, C. Riveros, C. Rojas, E. Zerega, Querying wikidata: Comparing sparql, relational and graph databases, in: P. Groth, E. Simperl, A. J. G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck, Y. Gil (Eds.), The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II, volume 9982 of *Lecture Notes in Computer Science*, 2016, pp. 88–103. URL: https://doi.org/10.1007/978-3-319-46547-0_10. doi:10.1007/978-3-319-46547-0_10.

[30] T. Kovács, G. Simon, G. Mezei, Benchmarking graph database backends - what works well with wikidata?, Acta Cybern. 24 (2019) 43–60. URL: https://doi.org/10.14232/actacyb.24.1.2019.5. doi:10.14232/actacyb.24.1.2019.5.

[31] T. Minier, H. Skaf-Molli, P. Molli, SaGe: Web preemption for public SPARQL query services, in: The World Wide Web Conference on - WWW '19, ACM Press, 2019, pp. 1268––1278. doi:10.1145/3308558.3313652.

[32] D. Henselmann, A. Harth, Constructing demand-driven wikidata subsets, in: L. Kaffee, S. Razniewski, A. Hogan (Eds.), Proceedings of the 2nd Wikidata Workshop (Wikidata 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24, 2021, volume 2982 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2982/paper-10.pdf.

[33] J. Aimonier-Davat, H. Skaf-Molli, P. Molli, A. Grall, T. Minier, Online approximative SPARQL query processing for COUNT-DISTINCT queries with web preemption, Semantic Web 13 (2022) 735–755. doi:10.3233/sw-222842.

[34] H. Chalupsky, P. Szekely, F. Ilievski, D. Garijo, K. Shenoy, Creating and querying personalized versions of wikidata on a laptop, 2021. doi:10.48550/ARXIV.2108.07119.

[35] M. Färber, F. Bartscherer, C. Menne, A. Rettinger, Linked data quality of DBpedia, freebase, OpenCyc, wikidata, and YAGO, Semantic Web 9 (2017) 77–129. doi:10.3233/sw-170275.

[36] Eclipse, Rdf4j 4.0.0 released, 2022. URL: https://rdf4j.org/news/2022/04/24/rdf4j-4.0.0-released/.