# A Taxonomy of Basic Graph Pattern Motifs for Understanding SPARQL Query Logs

Jaime Salas[1], Aidan Hogan[1]

[1]*DCC, Universidad de Chile; IMFD*

### Abstract

Popular SPARQL query endpoints hosted by open knowledge graphs such as Wikidata and DBpedia process hundreds of thousands or even millions of queries per day. Making sense of queries at this scale is challenging. We propose a taxonomy of basic graph patterns (BGPs) in order to induce a hierarchical structure from such patterns found in a large query log. The leaves of this taxonomy are the raw basic graph patterns extracted from each query of the log. Each layer thereafter applies a generalisation step followed by a canonicalisation step, with each layer representing an increasingly coarse partition based on an increasingly more general motif. Generalisations are applied for constant subjects/objects (nodes), constant predicates (edge labels), direction, constant/variable distinction, and homomorphic equivalence. We discuss use-cases, define these generalisation steps, and apply them to induce a taxonomy of BGPs from a subset of the Wikidata query log.

### Keywords

SPARQL, canonicalisation, query logs, graph queries, Wikidata

## 1. Introduction

There are now hundreds of public SPARQL endpoints available on the Web [14], with some of the largest, such as those hosted by DBpedia [12] and Wikidata [6], evaluating in the order of hundreds of thousands or millions of queries per day. Although supporting such a volume of queries poses significant engineering and scientific challenges [14, 6], samples of these logs have been published [6, 12], and provide significant research opportunities [7]. Specifically, these query logs contain key insights into the distribution of queries of interest to different clients in practice in terms of the complexity of the query patterns [13, 3], which operators are more (or less) frequently used [3, 12], which operators tend to be used together [3], whether queries originate from humans or bots [9, 17], how clients refine or modify queries from one request to the next [16], etc.

Thus, given a large SPARQL query log such as that published for Wikidata [6], or as part of the Linked SPARQL Queries dataset [12], there is a wide range of analyses that a researcher or database administrator can run in order to gain insights into trends in those queries. What we argue is missing is a way to organise and "browse" the queries of such logs hierarchically. We

envisage the ability to make sense of large SPARQL query logs via a taxonomy that begins with very general (equivalence) classes of queries that become increasingly more specific in lower layers, providing first a high-level view of categories that open up into more and more specific categories of queries.

The design of such a taxonomy is challenging, and indeed there are potentially many ways in which such a taxonomy can be defined. But we think the idea is worth exploring in order to help bring order to large SPARQL query logs. In this work, we take some initial steps in this direction by focusing on the case of *basic graph patterns* [4], proposing a taxonomy to induce a hierarchical structure from them. This structure is based on generalisation and canonicalisation [10] steps applied iteratively at each layer of the hierarchy, inducing a set of motifs at each layer that constitute a partition of the queries considered. We currently foresee five use-cases:

**Human vs. bot detection [9, 17]:** The taxonomy can help identify motifs for highly-regular queries, such as parameterised queries used by bots.

**Caching [15, 8]:** The taxonomy can identify frequent motifs that, if cached, could increase cache hit rates and improve query evaluation performance.

**Optimisation [1]:** The taxonomy could be used to decide on specialised indexes and optimisations to reduce the costs for frequent motifs.

**Benchmarking [11, 2]:** The taxonomy can be used to classify different types of graph patterns for benchmarking, enabling the comparison of different engines for different motifs.

**Log exploration [3]:** The taxonomy can help, for example, to identify unusual traffic through highly-specific motifs with large numbers of instances.

**Paper outline**  In Section 2, we discuss related works that classify or generalise queries from SPARQL logs. Section 3 presents the taxonomy we propose, while Section 4 presents the results of applying our taxonomy to a sample of the Wikidata query logs. Section 5 concludes.

## 2. Related Work

The use of SPARQL query logs for understanding user demands has become a rich topic of interest, leading to a wide body of research. For space reasons, we focus on those works that specifically look at generalising and/or classifying real-world queries in such logs.

The first category of related works extract high-level query patterns for the purpose of analysing logs. Bonifati et al. [3] present a detailed analysis of the Wikidata and LSQ query logs, where they present frequency distributions relating to query size, operator usage, etc., but also look at the graph structure of basic graph patterns, defining a list of different graph *shapes* that correspond to a specific subset of the high-level motifs explored here.

The second category of works process log queries for the purposes of benchmarking. Saleem et al. [11] propose a framework, called FEASIBLE, that extracts different features from queries, such as join vertex degree and shape, number of triple pattern and their selectivity, etc.; these features are used to select a small number of representative queries for a benchmark. Angles et

al. [2] propose high-level categories of subqueries (BGPs, property paths, etc.) for the Wikidata query log that are then used to compare the performance of different engines.

In terms of novelty, rather than pre-defining shapes/motifs, we instead pre-define a set of generalisation steps that can produce arbitrary motifs. We are also not aware of works that propose a hierarchical way to structure such motifs.

## 3. A Taxonomy of Basic Graph Pattern Motifs

In this section, we present our proposed taxonomy of BGPs based on increasingly high-level motifs induced by a sequence of generalisation steps. We first present definitions, then an example, and finally summarise how we implemented software to compute the taxonomy.

**Preliminaries.**  Let $\mathbf{I}$, $\mathbf{L}$ and $\mathbf{V}$ denote the set of all IRI, literal and variable terms, respectively. A *triple pattern* $t = (s, p, o) \in (\mathbf{I} \cup \mathbf{L} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{L} \cup \mathbf{V})$ is an RDF triple allowing variables in any position. We use $\mathbf{C} = \mathbf{I} \cup \mathbf{L}$ to denote constants where the distinction between IRIs and literals does not matter. A set of triple patterns is called a *basic graph pattern* (*BGP*).[1] Letting $B$ denote a BGP, we denote by $\mathrm{vars}(B)$ the set of variables appearing in $B$, by $\mathrm{cons}(B)$ the set of constants appearing in $B$, by $\mathrm{nodes}(B)$ the set of subject/object terms (i.e., nodes) in $B$, and by $\mathrm{labs}(B)$ the set of predicate terms (i.e., edge labels) in $B$.

Given a BGP $B$, we will define a sequence of generalisation steps that form increasingly high-level equivalence classes of BGPs induced by patterns that we call *motifs*. Specifically, a *generalisation step $s$* is a transformation of a BGP that yields an equivalence relation $\sim_s$ on BGPs such that, given two BGPs $A$ and $B$, we say that $A \sim_s B$ if and only if $s(A) = s(B)$. We then call $s(A)$ (or equivalently $s(B)$) an *s-motif*. To define specific generalisation steps, we introduce a partial *term mapping* $\tau : \mathbf{C} \cup \mathbf{V} \to \mathbf{C} \cup \mathbf{V}$ whose domain is denoted by $\mathrm{dom}(\tau)$. Given a BGP $B$, we denote by $\tau(B)$ the *image* of $B$ under $\tau$, i.e., $\tau(B) = \{(\tau'(s), \tau'(p), \tau'(o)) \mid (s, p, o) \in B\}$ where $\tau'(x) = \tau(x)$ for all $x \in \mathrm{dom}(\tau)$, and $\tau'(x) = x$ for all $x \notin \mathrm{dom}(\tau)$; i.e., $\tau$ rewrites some of the terms in $B$, leaving terms for which it is not defined as they are. Letting s, p and o denote the subject, predicate and object positions, we may also use *selective images* to rewrite terms only in selected positions; for example, $\tau_{s,o}(B) = \{(\tau'(s), p, \tau'(o)) \mid (s, p, o) \in B\}$ only rewrites subject and object terms with $\tau$.

**Proposed Taxonomy.**  There are innumerable distinct generalisation steps that one could consider, and indeed the "best" choice may depend on external factors such as the particular use-cases, logs, etc., involved. The order in which the generalisation steps are applied can also affect the resulting taxonomy. In the following, we propose a concrete set of generalisation steps inspired by the idea of incrementally generalising the graph structure of BGPs. As a zeroth step, we abstract away variable names from the input BGPs (modulo isomorphism). In subsequent steps we abstract away the values of particular constants in order to yield an increasingly abstract graph structure. We then abstract away edge labels and direction in order to yield a directed and then undirected graph, respectively. Thereafter we abstract away the distinction between constants and variables, and finally we compute the core of the graph. As

---

[1] We assume that blank nodes appearing in a BGP are mapped to fresh variables [10].

aforementioned, this is one way in which generalisation steps can be applied. Exploring other possible steps, and the motifs and taxonomies they yield, is an interesting topic for future work.

**Definition 3.1.** We define the following generalisation steps, where $A$ and $B$ are BGPs:

0. The zeroth step generalises variable names, capturing isomorphism modulo variables. We say that $A \sim_0 B$ if and only if there exists a one-to-one term mapping $\alpha : \mathrm{vars}(A) \to \mathrm{vars}(B)$ such that $\alpha(A) = B$.

1. The first step further generalises constant nodes in a BGP, capturing isomorphism modulo variables and constant nodes. We say that $A \sim_1 B$ if and only if there exists a one-to-one term mapping $\beta : \mathrm{nodes}(A) \cap \mathrm{cons}(A) \to \mathrm{nodes}(B) \cap \mathrm{cons}(B)$ such that $\beta_{\mathrm{s,o}}(A) \sim_0 B$.

2. The second step further generalises constant edge labels (predicates) in a BGP, capturing isomorphism modulo variables and constants. We say that $A \sim_2 B$ if and only if there exists a one-to-one term mapping $\gamma : \mathrm{labs}(A) \cap \mathrm{cons}(A) \to \mathrm{labs}(B) \cap \mathrm{cons}(B)$ such that $\gamma_{\mathrm{p}}(A) \sim_1 B$.

3. The third step generalises edge labels, capturing isomorphism of the directed graph of the BGP while still distinguishing constants and variables. Let $\mathsf{c} \in \mathbf{C}$ and $\mathsf{v} \in \mathbf{V}$ denote a reserved constant and variable, respectively. Let $\delta : \mathbf{C} \cup \mathbf{V} \to \{\mathsf{c}, \mathsf{v}\}$ denote a term mapping such that $\delta(x) = \mathsf{c}$ for all $x \in \mathbf{C}$ and $\delta(x) = \mathsf{v}$ for all $x \in \mathbf{V}$. We say that $A \sim_3 B$ if and only if $\delta_{\mathrm{p}}(A) \sim_2 \delta_{\mathrm{p}}(B)$.

4. The fourth step generalises edge direction, capturing isomorphism of the undirected graph of the BGP while still distinguishing constants and variables. Let $B^\pm = \{(s, p, o) \mid (s, p, o) \in B \text{ or } (o, p, s) \in B\}$ denote the completion of $B$. We say that $A \sim_4 B$ if and only if $A^\pm \sim_3 B^\pm$.

5. The fifth step generalises constant and variable distinction, capturing isomorphism of the undirected graph of the BGP without distinguishing constants and variables. Here we choose to map constants to variables. We say that $A \sim_5 B$ if and only if there exists a one-to-one term mapping $\epsilon : \mathrm{cons}(A) \cup \mathrm{cons}(B) \to \mathbf{V} \setminus (\mathrm{vars}(A) \cup \mathrm{vars}(B))$ such that $\epsilon(A) \sim_4 \epsilon(B)$.

6. The sixth step generalises non-core edges and nodes, capturing homomorphic equivalence of the undirected graph of the BGPs. We say that there is a homomorphism from $A$ to $B$, denoted $A \to B$, if and only if there exists a term mapping $\mu : \mathrm{vars}(A) \to \mathrm{vars}(B) \cup \mathrm{cons}(B)$ such that $\mu(A) \subseteq B$. We then say that $A$ and $B$ are homomorphically equivalent modulo variables, denoted $A \simeq_0 B$, if and only if $A \to B$ and $B \to A$. Let us overload the previous definitions, where we denote by $\simeq_{0,\ldots,5}$ the steps $\sim_{0,\ldots,5}$ replacing the zeroth step $\sim_0$ with $\simeq_0$. We say that $A \sim_6 B$ if and only if $A \simeq_5 B$.

Each generalisation step $s$ induces a partition of a set of BGPs $\mathcal{B}$ via the quotient set $\mathcal{B}/\!\sim_s$.

**Example.** Take the following SPARQL query, which intuitively asks for information about papers published in AMW that cite other AMW papers:

```
SELECT ?x ?p ?o WHERE { ?x :venue :AMW . ?y :venue :AMW . ?x :cites ?y . ?x ?p ?o . }
```

Figure 1 presents the result of the previously defined generalisation steps for the BGP of this query, where we use circles to represent existential values whose particular value does not
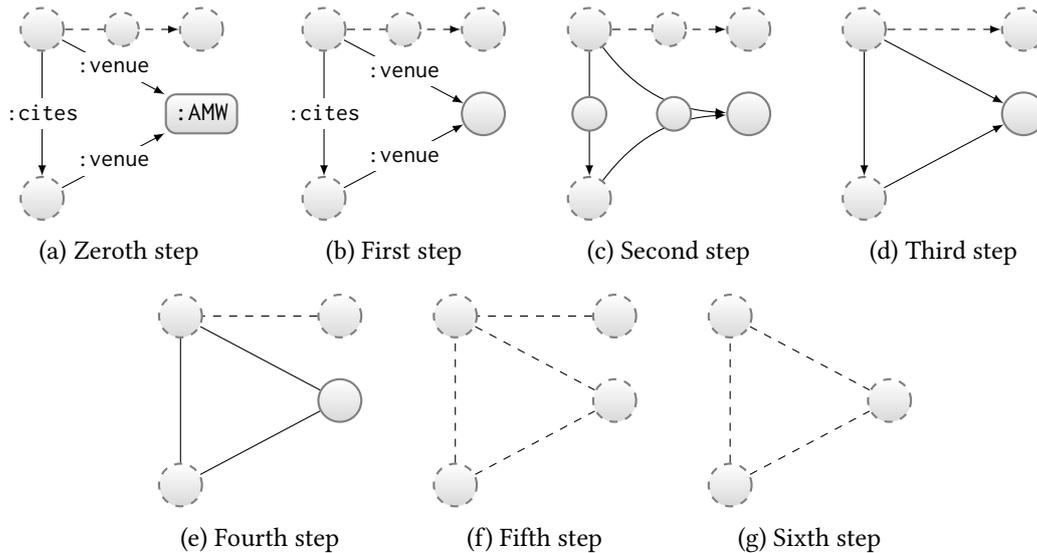
**Figure 1:** Proposed generalisation steps applied to an example SPARQL query

matter, solid lines to indicate constants, and dashed lines to indicate variables. Each step creates an increasingly high-level motif. Applying these generalisations to a set of BGPs then generates a partition according to BGPs having the same motif at that level. By design, each generalisation step builds upon the previous, which means that the partition induced by each step is strictly finer that the one that came before, inducing a hierarchical taxonomy.[2]

**Implementation.** We implemented the extraction of the aforementioned taxonomy using the QCan package [10], which offers methods for canonicalising SPARQL queries. Specifically, QCan represents SPARQL BGPs as RDF graphs in a reified form using *r-graphs*. The package can then invoke the blabel package [5], which canonically labels existential values, and can also perform RDF leaning, which is used to calculate the core of the graph required for step 6.

## 4. An Analysis of Wikidata's BGPs

In this section, we extract the taxonomy from the BGPs of a large sample of queries from the Wikidata SPARQL logs [6]. We work with two samples: a subset of the robotic queries, and a subset of the organic/human queries. The samples are taken from queries received between 2018-02-26 and 2018-03-25, i.e., the most recent interval available in the Wikidata SPARQL logs. Though performance is not a focus of the current work, we note that computing the taxonomy for close to four million BGPs took a little over three hours.

---

[2]An unintended consequence of layering generalisation steps is that, by renaming predicates separately from nodes, the second step may rename a constant in a predicate position differently from how the first step renamed the same constant in a node position; correspondences for constants in predicate and node positions are lost. A solution would be to merge the first and second steps, losing some granularity, or defining the second step directly over the zeroth step without using a selective image.

**Table 1**

The number of unique motifs, and the largest equivalence class for a single motif, found for each step in the robotic and organic BGPs from Wikidata.

| (a) Robotic BGPs | | | | (b) Organic BGPs | | |
|---|---|---|---|---|---|---|
| Step | Unique | Max | | Step | Unique | Max |
| 0 | 107,583 | 239,523 | | 0 | 134,329 | 286,994 |
| 1 | 2,380 | 251,956 | | 1 | 13,315 | 388,836 |
| 2 | 240 | 590,529 | | 2 | 1,776 | 586,021 |
| 3 | 198 | 590,880 | | 3 | 1,279 | 590,948 |
| 4 | 143 | 590,880 | | 4 | 917 | 590,948 |
| 5 | 55 | 1,190,351 | | 5 | 332 | 1,290,264 |
| 6 | 3 | 1,780,647 | | 6 | 4 | 2,197,940 |

**Robotic queries**    We first present the results for robotic queries. We took a random sample of 1,000,000 queries from the set of robotic queries in the interval. We then extracted all the BGPs contained in these queries, giving us a total of 1,781,337 BGPs. Table 1a presents some high-level statistics, showing the number of motifs generated at each step, along with the motif (equivalence class) with the largest number of BGPs. The results clearly show that each successive level of the taxonomy finds less unique motifs, while finding larger equivalence classes for these queries. This is of interest for exploring the logs: a user starting at step 6 (the highest level) can expand to a reasonable number (55) of child motifs at step 5, ordered by size; expanding these motifs from step 5, they can expect, on average, 2–3 child motifs, and so forth.

In Table 2, we present the top 5 motifs at each step for these robotic queries, starting from step 1. As before, dashed lines indicate variables while solid lines indicate constants. At step 1, the motifs represent parameterised queries, where nodes (subjects/objects), specifically, are generalised. The Wikidata properties P300 and P214 seen in this step indicate an ISO 3166-2 code for countries, and a VIAF ID used by libraries, respectively. The results show that the most common BGPs are composed of single triple patterns. The top motif is unusual in that it has no variable: it stems from a syntactic shortcut used in Wikidata for selecting the languages of labels returned by a custom service. We also see that the third, fourth and fifth motifs contain two variables; such BGPs may form part of a larger query, and be contained, for example, in an OPTIONAL clause. Steps 2 and 3 are quite similar to each other. We see that the most common motif in both steps refers to a singleton BGP with a constant predicate and variable nodes, but in the second and fifth positions, we see popular join shapes. Step 4 generalises edge direction, where we see a similar set of results to the previous two steps; the motif with a single edge having a constant and variable node is more prominent here as it merges two cases from step 3: one where the subject is the variable, and the other where the object is the variable. Removing the constant/variable distinction in step 5, we see many of the simpler motifs merged into the first two motifs, but thereafter, we see different types of joins, including disconnected patterns in the fourth and fifth positions; these disconnected patterns are due to geographic queries that apply a join between two variables based on geographic distance, rather than a natural equi-join. At step 6, which looks at the cores of the step 5 graphs, we find three motifs: 1,780,647

**Table 2**
Top 5 motifs for robotic queries after each step

| Step | Rank of motif per number of occurrences | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | wb:language 251,956 (14.14%) | s:description 135,073 (7.58%) | rdfs:label 95,105 (5.33%) | wdt:P300 88,199 (4.95%) | wdt:P214 83,229 (4.67%) |
| 2 | 590,529 (33.15%) | 342,885 (19.25%) | 286,728 (16.10%) | 268,915 (15.10%) | 71,524 (4.02%) |
| 3 | 590,880 (33.17%) | 342,886 (19.25%) | 296,520 (16.65%) | 270,846 (15.20%) | 71,664 (4.02%) |
| 4 | 590,880 (33.17%) | 348,187 (19.55%) | 302,953 (17.01%) | 296,520 (16.65%) | 76,558 (4.30%) |
| 5 | 1,190,353 (66.82%) | 428,815 (24.07%) | 68,540 (3.85%) | 53,511 (3.00%) | 26,998 (1.52%) |
| 6 | 1,780,647 (99.96%) | 649 (0.04%) | 41 (0.002%) | — | — |
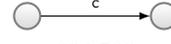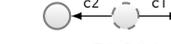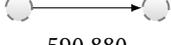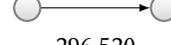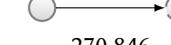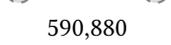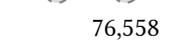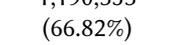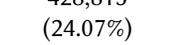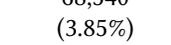
BGPs collapse to a core with a single edge between two nodes (this includes acyclic queries and queries with even-length cycles, for example), 649 BGPs collapse to a self-loop on one node (this includes all and only queries with a self-loop, i.e., a triple pattern with the same subject and object), while 41 BGPs collapse to a triangle (as per the example shown in Figure 1).

**Organic queries** Next we extract our taxonomy from the set of all 872,555 organic queries in the chosen interval, which contain 2,198,557 BGPs. Table 1b presents the overall statistics regarding the number of motifs and the largest equivalence class size at each step. Of interest is that the number of motifs is generally much higher than in the robotic case, indicating more diverse query shapes. This is to not surprising as one would expect robotic traffic to account for more regular (e.g., parameterised) queries when compared with queries created by users.

As before, Table 3 presents the top 5 motifs at each step – starting from step 1 – in the organic case. The Wikidata properties P31, P625 and P238 seen in step 1 refer to instance of, coordinate location and IATA airport code, respectively. We see the same top motif as in the case of robotic queries relating to Wikidata's label service. However, we also see a number of more complex motifs, including the fourth and fifth motif, which we suspect may be misclassified bot traffic (the logs classify organic and robotic traffic using a number of heuristics, such as user-agent, the volume of similar queries originating from a single source, etc. [6]). The fourth motif is disconnected; rather than being an equi-join, the motif represents a join based on geographic distance, where the join condition does not appear in the BGP. The fifth motif notably includes an edge with a variable, seemingly trying to retrieve all facts about airports. Looking at steps 2, 3 and 4, we see that although organic motifs share a lot in common with their robotic counterparts, they tend to be more complex, with the fifth motif in these steps having a join on three edges. At step 5, we again see the return of disconnected patterns resulting from the aforementioned geographic queries, with a large part of the fourth motif (209,322 of 211,016 instances) being accounted for by the fourth motif at step 1. At step 6, we again found an edge motif, a self-loop motif, and a triangle motif, but unlike in the robotic case, we also found a pentagon motif with a small number of instances (25 BGPs).

## 5. Conclusions

We believe that better tools are needed in order to summarise, structure, explore and understand the contents of large SPARQL logs. In this work, we have proposed a taxonomy for understanding the structure (specifically) of BGPs found in such a log. We proposed a sequence of generalisation steps that, when applied to BGPs, yield increasingly high-level motifs by incrementally abstracting away details from the BGP, generating a hierarchical taxonomy over those BGPs. Extracting the taxonomy for a sample of Wikidata queries, we gain some interesting insights into the most common motifs occurring in BGPs relating to frequently accessed predicates, frequent join types, the rarity of acyclic queries, the presence of disconnected graph patterns, and some potentially misclassified organic queries.

For future work, we would like to develop a user interface that allows for exploring the taxonomy of a given SPARQL query log: starting from step 6 (the highest level), the user is presented the motifs and the number of associated BGPs in descending order; upon clicking a motif, the sub-motifs at the lower level are presented in the same order. This would allow the user to start from a high-level view of the BGPs of the log, and "drill-down" on specific motifs for more details. We are also interested in exploring different sequences of generalisation steps that might yield more interesting insights. Another direction would be to study how different generalisation steps preserve different graph metrics; a key metric along these lines would be (hyper-)treewidth, which captures how "tree-like" a query is and predicts the complexity of evaluating the query [3].[3] Finally, it would be of interest to explore query features beyond simple BGPs; this seems non-trivial as it introduces many alternatives for generalisation steps, but could help to glean insights into how operators like paths, union, optionals, etc., are used.

---

[3]We note that step 6, which computes the core of the undirected graph, does not preserve treewidth; for example, it can collapse cycles of even length to a single edge, cycles of odd length or even cliques to a self-loop, etc.

**Table 3**
Top 5 motifs for organic queries after each step

| Step | Rank of motif per number of occurrences | | | | |
| --- | --- | --- | --- | --- | --- |
| | 1 | 2 | 3 | 4 | 5 |
| 1 | 388,836 (17.69%) | 239,752 (10.90%) | 215,078 (9.78%) | 209,322 (9.52%) | 209,084 (9.51%) |
| 2 | 586,021 (26.65%) | 413,058 (18.79%) | 250,833 (11.41%) | 221,809 (10.09%) | 209,457 (9.53%) |
| 3 | 590,948 (26.88%) | 413,175 (18.79%) | 251,147 (11.42%) | 237,488 (10.80%) | 209,474 (9.53%) |
| 4 | 590,948 (26.88%) | 413,175 (18.79%) | 286,165 (13.02%) | 251,254 (11.43%) | 209,514 (9.53%) |
| 5 | 1,290,288 (58.69%) | 347,870 (15.82%) | 238,128 (10.83%) | 211,016 (9.60%) | 38,859 (1.77%) |
| 6 | 2,197,940 (99.97%) | 404 (0.02%) | 188 (0.01%) | 25 (0.001%) | — |
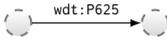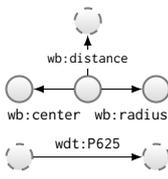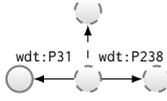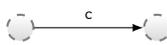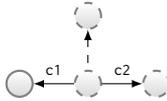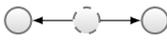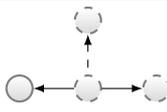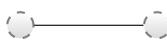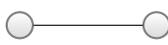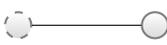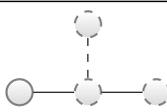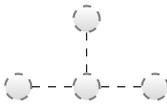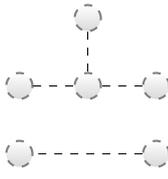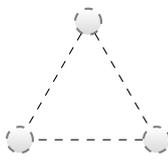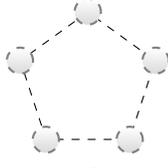
## Acknowledgments

## References

[1] Ali, W., Saleem, M., Yao, B., Hogan, A., Ngomo, A.N.: A survey of RDF stores & SPARQL engines for querying knowledge graphs. VLDB J. **31**(3), 1–26 (2022). https://doi.org/10.1007/s00778-021-00711-3

[2] Angles, R., Aranda, C.B., Hogan, A., Rojas, C., Vrgoc, D.: WDBench: A Wikidata Graph Query Benchmark. In: The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13489, pp. 714–731. Springer (2022). https://doi.org/10.1007/978-3-031-19433-7_41

[3] Bonifati, A., Martens, W., Timm, T.: An analytical study of large SPARQL query logs. VLDB J. **29**(2-3), 655–679 (2020). https://doi.org/10.1007/s00778-019-00558-9

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

[5] Hogan, A.: Canonical forms for isomorphic and equivalent RDF graphs: Algorithms for leaning and labelling blank nodes. ACM TOW **11**(4), 22:1–22:62 (2017). https://doi.org/10.1145/3068333

[6] Malyshev, S., Krötzsch, M., González, L., Gonsior, J., Bielefeldt, A.: Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph. In: The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II. pp. 376–394 (2018). https://doi.org/10.1007/978-3-030-00668-6_23

[7] Martens, W.: Towards Theory for Real-World Data. In: PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022. pp. 261–276. ACM (2022). https://doi.org/10.1145/3517804.3526066

[8] Papailiou, N., Tsoumakos, D., Karras, P., Koziris, N.: Graph-aware, workload-adaptive SPARQL query caching. In: ACM SIGMOD International Conference on Management of Data. pp. 1777–1792. ACM (2015). https://doi.org/10.1145/2723372.2723714

[9] Rietveld, L., Hoekstra, R.: Man vs. machine: Differences in SPARQL queries. In: Proceedings of the 4th USEWOD Workshop on Usage Analysis and the Web of of Data, ESWC (2014)

[10] Salas, J., Hogan, A.: Semantics and canonicalisation of SPARQL 1.1. Semantic Web **13**(5), 829–893 (2022). https://doi.org/10.3233/SW-212871

[11] Saleem, M., Mehmood, Q., Ngomo, A.N.: FEASIBLE: A feature-based SPARQL benchmark generation framework. In: The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9366, pp. 52–69. Springer (2015). https://doi.org/10.1007/978-3-319-25007-6_4

[12] Stadler, C., Saleem, M., Mehmood, Q., Buil-Aranda, C., Dumontier, M., Hogan, A., Ngomo,

A.C.N.: LSQ 2.0: A Linked Dataset of SPARQL Query Logs. Semantic Web Journal (2023), (to appear)

[13] Stegemann, T., Ziegler, J.: Pattern-Based Analysis of SPARQL Queries from the LSQ Dataset. In: Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017. CEUR Workshop Proceedings, vol. 1963. CEUR-WS.org (2017)

[14] Vandenbussche, P., Umbrich, J., Matteis, L., Hogan, A., Aranda, C.B.: SPARQLES: monitoring public SPARQL endpoints. Semantic Web **8**(6), 1049–1065 (2017). https://doi.org/10.3233/SW-170254

[15] Williams, G.T., Weaver, J.: Enabling fine-grained HTTP caching of SPARQL query results. In: The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I. pp. 762–777 (2011). https://doi.org/10.1007/978-3-642-25073-6_48

[16] Zhang, X., Wang, M., Saleem, M., Ngomo, A.N., Qi, G., Wang, H.: Revealing secrets in SPARQL session level. In: International Semantic Web Conference (ISWC). LNCS, vol. 12506, pp. 672–690. Springer (2020). https://doi.org/10.1007/978-3-030-62419-4_38

[17] Zhang, X., Wang, M., Zhao, B., Liu, R., Zhang, J., Yang, H.: Characterizing Robotic and Organic Query in SPARQL Search Sessions. In: Web and Big Data - 4th International Joint Conference, APWeb-WAIM 2020, Tianjin, China, September 18-20, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12317, pp. 270–285. Springer (2020). https://doi.org/10.1007/978-3-030-60259-8_21