Leveraging RDF-Based Data Structures for Optimized Traversal in Decentralized Query Systems

Himanshu Arora
him.arora0497@gmail.com

***Abstract*—The vast, decentralized nature of web data presents significant obstacles for efficient querying, especially in environments where data sources are unindexed and distributed across the web. Traditional approaches, including Link Traversal Query Processing (LTQP), struggle with performance issues such as slow query execution and an overwhelming number of HTTP requests due to the lack of centralized indexing. This study proposes an innovative optimization technique that utilizes RDF-based data structures to enhance query efficiency in such decentralized contexts. By introducing a structure-aware indexing framework, we enable the query engine to better anticipate the relevant data sources for traversal, thus reducing unnecessary queries and streamlining the retrieval process. Our method uses explicit data structure mappings, referred to as shape-based indexes, which align with the inherent schema-like properties of RDF data. Early experimental evaluations demonstrate that this approach can significantly reduce query processing time by up to 75% and decrease the number of redundant resource accesses by over 90%. This work provides a new direction for improving the scalability and efficiency of querying in decentralized data systems, offering a viable alternative to the challenges posed by the traditional methods of data retrieval in the open web.**

I. Introduction

The World Wide Web (WWW) inherently operates as a decentralized information space. While aggregating substantial segments of the web into centralized endpoints enhances the ease of querying and reduces latency, such centralization raises ethical and legal concerns, including surveillance, monopolization, and data misuse. These concerns underscore the necessity of exploring alternative, decentralized query mechanisms capable of operating efficiently without full data centralization.

Early attempts to support queries over web data include query languages such as webSQL [1] and SPARQL. Both languages aim to enable conjunctive querying over distributed sources. However, webSQL depends heavily on web indexing [1], which presents scalability issues when applied to the dynamic and expansive nature of the web. Indexing at such a scale is resource-intensive and demands frequent updates to stay relevant. Moreover, index-based techniques can inadvertently exclude data sources, reducing the organic discovery potential of the web.

In contrast, SPARQL is tightly coupled with the principles of linked data, which encourage the use of Internationalized Resource Identifiers (IRIs) to interconnect data across the web. While this model promotes serendipitous data discovery without relying on indexes, most SPARQL query implementations still depend on centralized or federated infrastructures. Consequently, truly decentralized and indexing-independent query solutions remain largely experimental.

Link Traversal Query Processing (LTQP) [2] represents one such decentralized solution. LTQP queries are evaluated over a dynamic triple store that evolves during query execution. Starting from a user-supplied set of seed IRIs, the query engine incrementally dereferences linked IRIs, populating the triple store on-the-fly. This live querying approach offers significant flexibility and reduces reliance on prior indexing. However, it also introduces key challenges.

One major issue lies in the theoretically unbounded search space of the WWW [3], making exhaustive exploration impractical. Additionally, performance is hampered by the latency and unpredictability of HTTP requests, which constitute a significant bottleneck [3]. To address this, LTQP frameworks often utilize *reachability criteria* [2], which limit exploration to data reachable through IRIs found within already acquired documents. While this strategy improves efficiency, it does not solve all issues—particularly the difficulty of query planning in the absence of source metadata.

To mitigate the lack of source awareness, current LTQP engines employ rule-based heuristics for join ordering and execution [4]. While effective in some contexts, these heuristics are suboptimal in highly dynamic environments where the cost of dereferencing irrelevant documents is high.

Historically, LTQP was developed with the open web in mind. Recent research, however, has turned toward structured linked data environments, where predictable publishing patterns can inform query optimization. These patterns, known as *structural assumptions* [5], serve as implicit contracts between data publishers and consumers. They help guide query engines toward relevant sources, improving performance and reducing unnecessary exploration.

An example of this is found in the Solid ecosystem, which uses hypermedia descriptions to expose the contents of Solid pods [6]. Although this mechanism supports data discovery, it is not expressive enough for detailed query planning. To overcome this, Solid integrates a type index specification[1], which maps RDF classes to resources—offering a more declarative approach to data localization [7].

Utilizing structural assumptions in environments such as Solid has led to marked improvements in query execution

[1]

performance [5], [8]. In some cases, optimization has reached a level where the HTTP layer is no longer the bottleneck; instead, inefficiencies stem from suboptimal query plans dictated by current heuristics. However, for complex queries or in less structured environments, HTTP overhead remains a critical constraint [8].

This motivates the hypothesis that many HTTP requests performed during LTQP yield non-contributory results. Thus, reducing unnecessary dereferencing is a promising optimization avenue. In pursuit of this, we explore the use of *RDF data shapes* as an advanced form of structural assumption.

RDF data shapes, traditionally used for data validation [9], offer a formal and expressive way to describe the expected structure of RDF resources. Although their primary role lies in validation, some studies have also explored their use in optimizing query execution [10]. Building on this, we propose a novel construct termed the *shape index*, which functions as a summarization mechanism over decentralized datasets [11], [12], [13]. The shape index facilitates intelligent source selection, and we foresee its applicability extending to link queue prioritization and dynamic query planning.

This paper presents our preliminary efforts to leverage shape-based metadata for more efficient data discovery and pruning of irrelevant links during LTQP. By integrating RDF data shapes as a structural assumption, we aim to significantly narrow the search space and alleviate one of the most persistent challenges in querying decentralized linked data environments.

II. Related Work

The problem of efficient query processing over decentralized and fragmented RDF data has been extensively studied in the context of Linked Data and Web-based semantic systems. Traditional approaches such as Link Traversal Based Query Processing (LTQP) [3] rely on the dynamic discovery and dereferencing of RDF documents during query execution. While this strategy promotes flexibility, it often incurs high latency and unpredictable performance due to the excessive number of HTTP requests and the inherent unreliability of web servers.

To improve source selection in such decentralized environments, multiple proposals have introduced indexing strategies. One prominent method is the *type index*, used in Solid-based personal data stores, where data types are annotated to guide query engines in selecting relevant sources [5]. While type indexes help prune the search space, they offer only coarse-grained guidance and cannot exploit structural information embedded in data schemas.

Our work builds on this foundation by introducing a more expressive alternative in the form of *shape indexes*. These leverage RDF data shape languages such as SHACL and ShEx [14], [15] to describe the structure and constraints of datasets. Related works in RDF validation and containment [16], [17] have demonstrated the feasibility of reasoning over shapes, but their application to query source selection is novel.

Other notable contributions include strategies for precomputing source descriptions [**?**], federated SPARQL query planning [**?**], and heuristics for dataset relevance estimation [**?**] that exploit metadata or statistical summaries. However, these typically assume centralized control or preprocessed indexes, limiting their applicability in dynamic or user-controlled data ecosystems.

Work by Hartig [3] highlighted the challenges of query execution over Web data using traversal methods, emphasizing the cost of unpredictable network interactions. Similarly, Taelman et al. [18] provided a modular framework for experimentation in this domain through the Comunica engine, which supports a variety of link traversal and federated querying scenarios.

In contrast to approaches that depend on global precomputed indexes or domain-specific assumptions, our shape index approach is designed to be domain-agnostic and compatible with decentralized publishing models. It complements existing Linked Data practices such as the Linked Data Platform (LDP) [2], while enabling more informed decision-making during query execution.

Finally, benchmarking efforts such as SolidBench [5] have played a key role in evaluating the performance of different traversal and indexing methods under realistic conditions. Our implementation and evaluation framework builds on these efforts to enable reproducibility and fair comparison against state-of-the-art techniques.

III. Shape Index and Query-Shape Containment

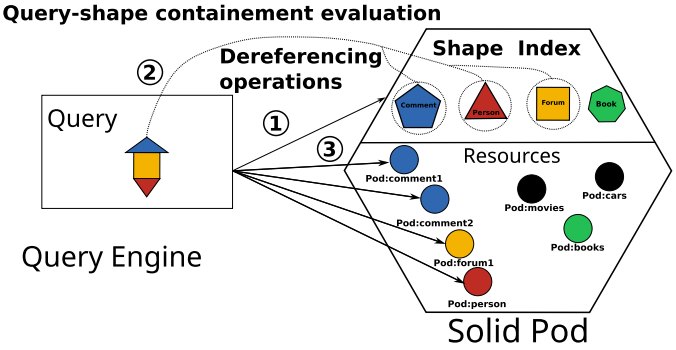

Fig. 1. First, the shape index is dereferenced, then the *query-shape containment* operations are performed in the query engine and lastly, only the relevant resources are dereferenced.

While the RDF model is schema-less by design, most RDF datasets still exhibit consistent structural patterns due to their modeling constraints and real-world semantics [19]. These patterns often function as implicit schemas, and prior work has successfully leveraged such structural regularities to improve query performance [19], [20]. Building on these insights, we adapt these techniques for Link Traversal Query Processing (LTQP) by proposing the use of explicit data structure definitions—schemas offered by the data providers themselves—to enhance source selection and reduce redundant data access.

A. Shape Index

To reduce the overhead of superfluous HTTP dereferencing and improve source selection accuracy within linked data subdomains, we introduce the notion of a *shape index*. The central idea is to exploit metadata about the structure of RDF resources within a domain to inform and guide the query engine. Concretely, a shape index is defined as a set of associations between RDF data shapes and sets of RDF documents or resources, scoped within a specific domain. [3]

Unlike conventional shape mapping [21] or SHACL target declarations [22], which bind shapes to specific parts of RDF graphs, the shape index operates at a higher level by linking shapes directly to RDF documents. Moreover, the proposed index offers a streamlined alternative to the more complex Shape Trees standard [4], focusing primarily on query optimization rather than data validation or access control.

Each shape index specifies its scope using a domain definition (a set of IRIs or patterns) and includes a completeness flag. If a shape index is marked *complete*, this implies that every RDF resource within the domain conforms to at least one shape listed in the index. Additionally, any data instance matching a listed shape must be located within one of the associated RDF documents. These constraints are key to enabling advanced reasoning during the query planning and traversal process.

B. Query-Shape Containment

A major benefit of shape indexes is their capacity to allow pre-traversal filtering of relevant resources through a process we refer to as *query-shape containment*. The goal is to determine, prior to issuing dereference requests, whether the query is structurally contained within the known shapes of the domain, thereby allowing the engine to prune irrelevant resources early in execution.

This task draws inspiration from the classical *query containment problem* [23], [24], [25], which investigates whether the result of one query is always a subset of another regardless of the data. We extend this concept by translating RDF data shapes into equivalent SPARQL `SELECT` queries (Q_s) [14], [26], [27], [28]. The user query is then decomposed into multiple star-shaped subqueries (Q_{star}), which correspond to the central pattern and its immediate neighbors.

The containment check determines whether each Q_{star} is semantically included in a corresponding Q_s derived from the shape index. If containment holds, the query engine concludes that dereferencing the documents associated with Q_s is sufficient to retrieve all relevant data for Q_{star}. This pre-check allows the traversal process to focus only on productive resources and skip the rest, significantly improving performance.

Importantly, this method is dynamic and compatible with liberal reachability criteria such as c_{all} [2] or more refined policies like those in Solid [5]. Even under these expansive policies, our approach limits resource fetching based on containment outcomes, thereby mitigating long execution times typically associated with unconstrained traversals.

Figure 1 visualizes the proposed approach for a single domain. First, the engine discovers and dereferences the shape index—potentially located at the root of a Solid pod for easy access. [5] Then, query-shape containment is evaluated. Based on the containment mappings, the engine updates its traversal policy: resources not contributing to any matched Q_s are pruned, even if the index is incomplete.

When the shape index is marked as *complete*, the engine can guarantee that all matching data is covered. If not all Q_{star} subqueries are contained in any Q_s, the engine switches strategy. In this case, it may selectively dereference only those resources partially matching the user query—i.e., those whose shapes share a partial homomorphism with some Q_{star}.

In contrast, if the index is *incomplete*, then a full guarantee of query answerability cannot be ensured. However, the index can still be utilized for guided data discovery, similar to the Solid *type index*. The key difference is that RDF data shapes describe expected property structures, while type indexes provide only class-level declarations. Although dereferencing the RDF class IRIs might yield further information about entity properties, this is not standard practice in current linked data platforms [5].

Evaluating the comparative expressive power and utility of RDF shapes versus RDF class definitions remains an open research question and is left to future work. [6]

C. Illustrative Scenario

To better understand the effectiveness of our proposed approach, consider a concrete example where a user aims to retrieve a set of related data elements. Specifically, the user's goal is to extract the unique identifiers and textual content of comments in a distributed network, alongside the corresponding forum identifiers where these comments were posted, and the names of the moderators managing those forums.

As illustrated in Figure 1, this query naturally decomposes into three distinct but interconnected star-shaped query patterns: one for retrieving comments ($Q_{comments}$), another for forums (Q_{forums}), and the third for moderators ($Q_{moderators}$). These subqueries are interconnected through shared variables, forming a complete query through a join operation: $Q = Q_{comments} \bowtie Q_{forums} \bowtie Q_{moderators}$.

In a traditional Linked Data query execution, the engine is unaware of the actual data contained within each RDF document until it is dereferenced. As such, the engine must dereference every resource that is deemed reachable under a given reachability criterion, which can be highly inefficient.

[3] From a perspective where the domain is composed of URLs leading to linked data documents and the codomain is composed of the documents with their RDF content.

[4] https://shapetrees.org/TR/specification/

[5] In this work, we do not take into consideration confidentiality restrictions.

[6] There exist comparisons between the shape and class definition approaches in the context of data validation [29] but it is left to be determined if their frame of comparison is compatible with our current problem and foreseen opportunities.

However, the integration of a shape index can significantly optimize this process.

For example, if the query engine encounters a domain that is confirmed by a *complete* shape index to contain only book-related data, it can skip dereferencing any of those documents, as they are known not to contribute relevant results to the current query. Similarly, in a domain where both comments and movie reviews are published, but the query only concerns comments, the engine can restrict its dereferencing operations solely to those documents associated with comment data.

Furthermore, in a domain with a shape index—whether complete or incomplete—that declares support for comments, forums, and individuals (moderators being a type of individual), the query engine can use the containment relations to ignore irrelevant parts of the dataset, such as movie or book shapes, and prioritize only those matching the query's star patterns.

This targeted dereferencing leads to a more efficient and scalable traversal process by ignoring documents that are known in advance to be irrelevant. In essence, shape indexes enable the engine to traverse subdomains selectively and intelligently, improving both the speed and efficiency of query execution across decentralized data sources.

IV. Preliminary Evaluation

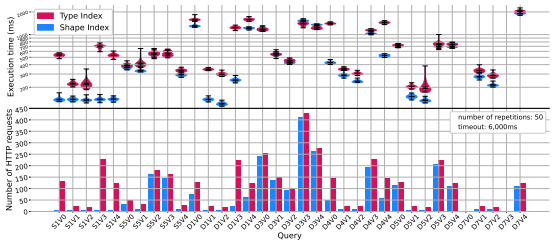

Fig. 2. The execution time using shape indexes consistently performs better than or equal to the state-of-the-art type index approach, with improvements reaching up to 80% (e.g., D1V3 and S1V3) and a significant reduction in HTTP requests. Queries are labeled using a pattern where the prefix (e.g., S1) indicates the scenario (short, dynamic, etc.), followed by the version identifier (e.g., V0). Some queries like D7V0 and D7V3 exceeded the timeout threshold and were excluded.

The testing excluded SPARQL queries involving property paths and unions, due to current limitations in the Comunica engine's implementation of these SPARQL 1.1 features. [7] Each query was run 50 times with a fixed timeout of 60 seconds. As seen in Figure 3, our method delivers up to an 80% reduction in execution time and as much as a 97% decrease in the number of HTTP requests.

This reduction arises because the shape index avoids unnecessary dereferencing of resources, unlike the type index approach that often queries each document blindly. Our approach only requires a few additional dereferences to retrieve the shape index itself, which are typically small and quickly retrievable.

Interestingly, we observed that there isn't always a one-to-one correlation between reduced HTTP requests and lower execution times. For instance, although D1V3 achieved a halving in request count, S1V3 showed a more pronounced impact on execution speed with a relatively smaller reduction in requests. This suggests that while reduced network I/O helps, internal query planning and engine overhead still play significant roles in overall performance [5].

In some queries, our method performed similarly or marginally worse than existing techniques. For example, D3V3 and D3V4 exhibited a 9% increase in average execution time, likely due to the overhead of the containment calculations. However, these trade-offs are minimal compared to the overall efficiency gains, especially in network-heavy environments.

We also noted that execution variance decreased when using the shape index, likely due to the predictable dereferencing strategy that avoids long-tail latency in HTTP requests [3]. This could be particularly beneficial in real-world scenarios involving unstable or latency-prone networks.

V. Experimental Results

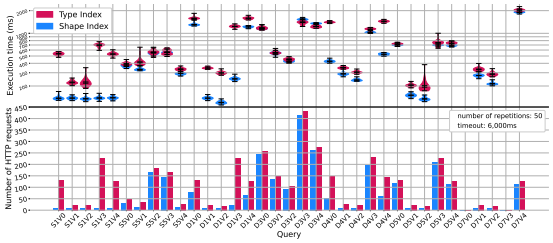

Fig. 3. Execution times and HTTP requests for shape index vs. type index. Shape indexes reduce execution time significantly (up to 80% in D1V3 and S1V3), and HTTP requests (up to 97%). Query naming follows the pattern: query type (e.g., S1 for short) followed by version number (e.g., V0). Missing values (e.g., D7V0, D7V3) indicate timeout.

We implemented the proposed shape index mechanism and integrated it into the Comunica query engine [18], following a modular and open-source design. [8] Our experimental setup utilized the SolidBench benchmarking framework [5], comparing the performance of our method against the state-of-the-art *type index* approach and standard Linked Data Platform (LDP) assumptions. [9]

To ensure fair evaluation, only a subset of queries supported by the current Comunica engine was used. Specifically, queries involving SPARQL property paths or unions—currently unsupported—were omitted. [10] Each query was executed 50 times, with a hard timeout of 60 seconds (6000 ms) per execution.

As illustrated in Figure 3, the use of shape indexes led to a marked improvement in both execution time and number of HTTP requests. For certain queries such as D1V3 and S1V3, we observed up to 80% faster execution and a reduction in

7

8

9

10

HTTP requests by 97%. These improvements stem from the shape index's ability to filter out non-relevant RDF documents preemptively, reducing network overhead and resource parsing.

Even in the worst cases (e.g., D3V3 and D3V4), the performance degradation was minimal, with only a 9% increase in execution time relative to the baseline. Additionally, the variance across runs was generally lower for the shape index, indicating more consistent performance across different network conditions. This can be attributed to fewer and more predictable HTTP requests, mitigating delays caused by server latency or intermittent failures [3].

Interestingly, the correlation between reduced HTTP requests and execution time is not strictly linear. For instance, while S1V3 showed a 97% drop in HTTP requests, the corresponding execution time improvement was comparatively modest. This suggests that while network requests dominate performance, internal query planning and containment computation may also contribute to latency.

Overall, our approach consistently outperformed or matched existing strategies across all metrics. The integration of shape-aware filtering into the source selection process allows for a more targeted and efficient traversal of RDF documents, making it especially well-suited for decentralized and data-fragmented environments like Solid.

VI. Conclusion

In this work, we introduced a novel shape index-based mechanism designed to enhance source selection in Link Traversal-Based Query Processing (LTQP). Through rigorous evaluation and benchmarking, we demonstrated that our shape-aware approach significantly reduces both execution time and the number of HTTP requests required during query evaluation. This improvement is particularly valuable in decentralized RDF environments, such as those found in Solid or similar document-fragmented linked data architectures.

Unlike traditional type index-based methods, our approach leverages shape information to selectively dereference only those documents likely to contain query-relevant data. This not only enhances query responsiveness but also reduces computational and network overhead—an essential feature in scenarios where bandwidth and latency constraints are critical. The results validate that even with the additional step of retrieving shape definitions, our method consistently outperforms or matches the state-of-the-art, delivering more predictable and efficient performance across a diverse set of queries.

Another key advantage of this method is its non-intrusive nature—it operates without requiring any modification to the underlying RDF data model or the architecture of existing linked data publishing workflows. [11] This makes it a viable option for large-scale adoption with minimal friction for data publishers.

In addition to its performance benefits, the proposed shape index mechanism has the potential to improve overall data quality in distributed RDF systems. By requiring publishers to formally specify data structures through shapes, inconsistencies and ambiguities in data modeling can be reduced, enhancing both semantic clarity and query accuracy.

Our findings indicate that precision in data source targeting is not only feasible but also beneficial at runtime. While our current solution addresses many fundamental challenges, it opens up new avenues for improving scalability, expressiveness, and interoperability in the semantic web landscape.

VII. Future Work

While our approach demonstrates promising improvements in LTQP performance, several open challenges and future directions remain to be explored.

- **Handling Sensitive or Private Data:** One of the critical challenges in practical deployment is the representation and usage of shapes for private or access-controlled data. Future research should investigate how shape indexes can be selectively disclosed or abstracted without compromising sensitive information.
- **Measuring the Cost of Shape Containment:** Although containment checks significantly aid in source selection, they introduce computational overhead. A detailed profiling and optimization of containment algorithms—especially for highly expressive shape languages—will be essential to mitigate latency in more complex query scenarios. [12]
- **Impact of Internal Store Size vs. Network Traffic:** Our current evaluation did not isolate the effects of reducing HTTP requests versus minimizing the number of triples loaded into the internal store. Future work should investigate how these two dimensions individually influence query performance and scalability.
- **Combining with Other Summarization Techniques:** While shape indexes provide structural insights, combining them with probabilistic data summaries (e.g., histograms, bloom filters) could further optimize query planning. Exploring such hybrid techniques might offer greater flexibility and precision in heterogeneous RDF environments.
- **Adapting Query Planners for Shape Awareness:** Current query engines are not optimized to fully exploit shape indexes. Enhancing query planners to be shape-aware—e.g., adjusting join orders or cardinality estimations based on shape metadata—could significantly boost efficiency and reliability in large-scale deployments.
- **Stability in Unreliable Networks:** Since our findings suggest that reduced variance in execution time is a byproduct of limited HTTP requests, further exploration is needed to evaluate performance in unstable or high-latency network conditions.

[11] This approach does not require changes to the existing RDF data model.

[12] For more on the expressiveness and computational considerations of RDF shape languages, refer to [14], [15], [16], [17], [30].

- **Incremental and Dynamic Indexing:** As RDF data is often updated in real-time, support for dynamic updates and incremental shape index generation is crucial. Future work should focus on designing lightweight synchronization protocols that keep shape indexes consistent with evolving data stores.
- **User-Centric Tools for Shape Design:** Creating, validating, and maintaining shapes can be a barrier to adoption. Developing intuitive tools and interfaces that assist publishers in defining reusable and interoperable shapes could significantly increase the practical utility of shape indexes.

By addressing these directions, we aim to evolve the shape index approach from a performance optimization technique to a foundational component of semantic web infrastructure, capable of enabling more intelligent, scalable, and secure data integration and query execution.

REFERENCES

[1] A. Mendelzon, G. Mihaila, and T. Milo, "Querying the world wide web," in *Fourth International Conference on Parallel and Distributed Information Systems*, 1996, pp. 80–91.

[2] O. Hartig and J.-C. Freytag, "Foundations of traversal based query execution over linked data," in *Conference on Hypertext and Social Media*, ser. HT '12. New York, NY, USA: ACM, 2012, p. 43–52.

[3] O. Hartig and M. T. Özsu, "Walking without a map: Optimizing response times of traversal-based linked data queries (extended version)," 2016.

[4] O. Hartig, "Zero-knowledge query planning for an iterator implementation of link traversal based query execution," in *The Semantic Web: Research and Applications*. Berlin, Heidelberg: Springer, 2011, pp. 154–169.

[5] R. Taelman and R. Verborgh, "Link traversal query processing over decentralized environments with structural assumptions," in *Proceedings of the 22nd International Semantic Web Conference*, Nov. 2023.

[6] R. T. Fielding, "Architectural styles and the design of network-based doftware architectures," Ph.D. dissertation, University of California, 2000.

[7] R. Taelman and R. Verborgh, "Declaratively describing responses of hypermedia-driven web apis," in *Knowledge Capture Conference*, ser. K-CAP '17. New York, NY, USA: Association for Computing Machinery, 2017.

[8] R. Eschauzier, R. Taelman, and R. Verborgh, "How does the link queue evolve during traversal-based query processing?" in *Proceedings of the 7th QuWeDa*, ser. CEUR Workshop Proceedings, Oct. 2023.

[9] J.-E. L. Gayo, E. Prud'hommeaux, I. Boneva, and D. Kontokostas, *Validating RDF Data: Applications*. Cham: Springer International Publishing, 2018, pp. 195–231.

[10] K. Rabbani, M. Lissandrini, and K. Hose, "Optimizing sparql queries using shape statistics," 2021.

[11] H. Stuckenschmidt, R. Vdovjak, G.-J. Houben, and J. Broekstra, "Index structures and algorithms for querying distributed rdf repositories," in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 631–639. [Online]. Available: https://doi.org/10.1145/988672.988758

[12] R. Goldman and J. Widom, "Dataguides: Enabling query formulation and optimization in semistructured databases," in *Proceedings of the 23rd International Conference on Very Large Data Bases*, ser. VLDB '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, p. 436–445.

[13] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, and J. Umbrich, "Data summaries for on-demand queries over linked data," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 411–420. [Online]. Available: https://doi.org/10.1145/1772690.1772733

[14] Delva, Thomas and Dimou, Anastasia and Jakubowksi, Maxime and Van den Bussche, Jan, "Data provenance for SHACL," in *Proceedings 26th International Conference on Extending Database Technology (EDBT 2023)*, vol. 26, 2023, pp. 285–297. [Online]. Available: http://doi.org/10.48786/edbt.2023.23

[15] S. Staworko, I. Boneva, J.-E. L. Gayo, S. Hym, E. G. Prud'hommeaux, and H. Solbrig, "Complexity and Expressiveness of ShEx for RDF," in *18th International Conference on Database Theory (ICDT 2015)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 31. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 195–211.

[16] I. Boneva, J.-E. L. Gayo, and E. G. Prud'hommeaux, "Semantics and validation of shapes schemas for rdf," in *The Semantic Web – ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2017, p. 104–120.

[17] S. Staworko and P. Wieczorek, "Containment of shape expression schemas for rdf," *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2018.

[18] R. Taelman, J. Van Herwegen, M. Vander Sande, and R. Verborgh, "Comunica: a modular sparql query engine for the web," in *Proceedings of the 17th International Semantic Web Conference*, Oct. 2018.

[19] T. Neumann and G. Moerkotte, "Characteristic sets: Accurate cardinality estimation for rdf queries with multiple joins," *2011 IEEE 27th International Conference on Data Engineering*, pp. 984–994, 2011.

[20] M. Meimaris and G. Papastefanatos, "Hierarchical characteristic set merging for optimizing sparql queries in heterogeneous rdf," *ArXiv*, vol. abs/1809.02345, 2018.

[21] J.-E. L. Gayo, E. Prud'hommeaux, I. Boneva, and D. Kontokostas, *Shape Expressions*. Cham: Springer International Publishing, 2018, pp. 55–117.

[22] ——, *SHACL*. Cham: Springer International Publishing, 2018, pp. 119–194. [Online]. Available: https://doi.org/10.1007/978-3-031-79478-0_5

[23] R. C. Foto Afrati, *Query Containment and Equivalence*. Springer Cham, 2019, ch. 2, pp. 21–59.

[24] M. Spasić and M. V. Janičić, "Solving the SPARQL query containment problem with SpeCS," *Journal of Web Semantics*, vol. 76, p. 100770, Apr. 2023.

[25] M. W. Chekol, J. Euzenat, P. Genevès, and N. Layaïda, "Sparql query containment under schema," *Journal on Data Semantics*, vol. 7, no. 3, p. 133–154, May 2018. [Online]. Available: http://dx.doi.org/10.1007/s13740-018-0087-1

[26] W3C. (2013) Sparql queries to validate shape expressions (informative). [Online]. Available: https://www.w3.org/2013/ShEx/toSPARQL.html

[27] J.-E. L. Gayo, E. Prud'hommeaux, H. Solbrig, and I. Boneva, "Validating and describing linked data portals using shapes," 2017.

[28] J. Corman, F. Florenzano, J. L. Reutter, and O. Savković, "Validating shacl constraints over a sparql endpoint," in *The Semantic Web – ISWC 2019*. Cham: Springer International Publishing, 2019, pp. 145–163.

[29] B. De Meester, P. Heyvaert, D. Arndt, A. Dimou, and R. Verborgh, "RDF graph validation using rule-based reasoning," *Semantic Web Journal*, vol. 12, no. 1, pp. 117–142, 2021.

[30] S. Lieber, A. Dimou, and R. Verborgh, "Statistics about data shape use in RDF data," in *Proceedings of the 19th International Semantic Web Conference: Posters, Demos, and Industry Tracks*, ser. CEUR Workshop Proceedings, vol. 2721, Nov. 2020, pp. 330–335. [Online]. Available: http://ceur-ws.org/Vol-2721/paper584.pdf