The Title of Your Contribution

Ruben Eschauzier[1], Ruben Taelman[1]

[1]Department of Electronics and Information Systems, Ghent University – imec

Abstract
Some cool abstract goes here!

Keywords
LaTeX class, paper template, paper formatting, CEUR-WS

This paper addresses (Issue #2) of the W3C Dataspaces Community Group.

1. Introduction

While centralizing data into warehouses or lakes can benefit query engine performance, it is antithetical to the core principles of *dataspaces*. Dataspaces promote a decentralized ecosystem where data remains at the source, managed by independent participants under agreed-upon governance models. This architectural shift introduces significant technical challenges for query engines, which must now discover and query data across a vast network of sources, each enforcing its own usage policies.

Centralized querying requires collecting large volumes of proprietary or sensitive data in a single source, which conflicts with the requirements for fine-grained sovereignty and minimized data replication. Conversely, traditional *federated* approaches [1, 2, 3] support decentralization but typically assume a static federation of a small number (10–100) of uniform, expressive endpoints (e.g., SPARQL endpoints) [4]. Enforcing complex usage policies [5] on such heavyweight interfaces significantly impacts performance [6], and these engines struggle to scale to the thousands of sources characteristic of a dataspace.

In contrast, realistic dataspace environments involve a massive number of permissioned sources exposing data through highly *heterogeneous interfaces* [7, 8]. Rather than a uniform layer of SPARQL endpoints, a dataspace participant may expose data via generic HTTP documents, constrained APIs, derived views, or specialized query services. Due to the scale and autonomy of the dataspace, the complete set of relevant sources and their specific interface capabilities is possibly unknown during query time. Such an environment requires a *hybrid* federation approach capable of querying across diverse data interfaces , discovered at runtime, without prior centralization.

Link Traversal-based Query Processing (LTQP) is an approach that meets these needs by discovering data sources during execution via hypermedia links found in earlier results. Starting from seed references (e.g., a self-description or catalog entry), it follows links asynchronously. While LTQP is traditionally defined as an approach for querying over federated *hypermedia* documents by following links at runtime, data within a dataspace is often exposed through diverse access interfaces that may not support standard hypermedia traversal and querying.

We propose *Hybrid LTQP*, a generalization of link traversal that supports the diverse access methods found in dataspaces—including SPARQL endpoints, materialized views, and Triple Pattern Fragments (TPF) [9]. To leverage these varied levels of expressivity, a hybrid engine must offload computation to the server whenever possible. For instance, a SPARQL endpoint can execute complex sub-queries locally. By leveraging internal indexing and local data knowledge, the endpoint can produce results far more efficiently than a client-side LTQP engine. Offloading these tasks to capable endpoints significantly

The Third International Workshop on Semantics in Dataspaces, co-located with the Extended Semantic Web Conference, June 01, 2025, Portorož, Slovenia
ruben.eschauzier@ugent.be (R. Eschauzier); ruben.taelman@ugent.be (R. Taelman)
0000-0002-6475-806X (R. Eschauzier); 0000-0001-5118-256X (R. Taelman)

reduces overall query latency. However, building such a system introduces several unresolved challenges. In this paper, we review the state-of-the-art in hybrid querying, analyze related work, and outline open issues for future research.

2. Related Work

While Hybrid LTQP has previously been discussed in literature, the exact definition of hybrid link traversal querying varies between papers.

2.1. Hybrid Traversal - Local vs Remote

A branch of hybrid link traversal querying uses precomputed indexes, either located within the decentralized environment or computed locally to speed-up query execution or source discovery. The first of such approaches, uses precomputed indexes to rerank sources based on relevance to the query's triple pattern and joins. and retrieves them to execute queries over. [10]. Other works [11] propose to cache information on sources to help in prioritization of sources.

Another approach is to store entire RDF graphs in local stores to quickly execute parts of the query. For this approach, non-blocking indexed-based operators [12] are used to leverage the locally computed indexes for faster query execution. A major limitation is the freshness of the data in the store, which can quickly go stale for high velocity data sources. Various methods, like freshness (or coherence)-aware query processing exist [13], however it remains a limitation.

2.2. Hybrid Traversal - Closed Systems

Other approaches integrate into the dataspace itself. ESPRESSO [14] is such an approach, it is a framework around the Solid dataspace specifically that defines several apps in the dataspace to perform certain $optimization$ tasks. In the Solid environment, data is stored into personal data vaults. These vaults use the Linked Data Platform [15] to arrange the data into a folder-like structure. To avoid traversing the entire pod to query it, ESPRESSO uses the *Brewmaster* application to locally create indexes of the pod and uses the *CoffeeFilter* application to search its contents. In addition, ESPRESSO uses an overlay network to distribute end-user queries to relevant data resources across Solid servers, where each Solid server is mapped to a federated database node in the overlay network.

Unlike approaches based on link traversal or service discovery within Solid, ESPRESSO relies on an explicitly configured overlay network, with no mechanism for dynamically discovering participating servers or query endpoints. Futhermore, currently ESPRESSO only supports distributed keyword-based search, it does not support SPARQL queries over the sources.

While these approaches do hybrid link traversal, they significantly deviate from our interpretation of hybrid traversal. This in contrast to the setting described in this paper where the discovered sources themselves are heterogeneous, serendiptiously discovered, and cannot be assumed to be present for every source within the dataspace.

2.3. Hybrid Link Traversal over Heterogeneous Sources

State-of-the-art LTQP engines, most notably Comunica [16, 17], demonstrate hybrid capabilities by dynamically federating over discovered SPARQL endpoints, TPF interfaces, and hypermedia documents. However, these systems are currently limited by a *lowest common denominator* strategy: they fail to leverage the full expressivity of sophisticated interfaces. For instance, upon discovering a SPARQL endpoint, existing engines restrict interaction to simple *triple pattern* lookups.

This forces the client to retrieve high volumes of intermediate results and perform joins locally, effectively negating the optimization and latency benefits of server-side execution.

3. Challenges with Hybrid Link Traversal

In this section, we will outline specific identified problems with hybrid link traversal making implementation difficult.

3.1. Establishing Exclusive Groups

In classical federated query processing, *exclusive groups* are sets of triple patterns within a query that can *only* be answered by a single source [1]. instead of retrieving results for individual triple patterns and joining them locally, the engine can delegate the entire group as a single sub-query (e.g., a Basic Graph Pattern) to the remote endpoint. This technique significantly reduces the number of remote requests and processed intermediate results [18, 1].

The identification of exclusive groups is a strict requirement for this sub-query delegation. If a query engine cannot verify that a group of patterns belongs *exclusively* to a single source, it cannot safely dispatch them as a conjunctive sub-query (BGP). Doing so risks *incomplete results* because the endpoint will execute a server-side join. If the endpoint lacks data for one of the patterns, or if valid join partners reside on a different server, the server-side join execution will filter out bindings that could have been successfully joined with data from another source. Thus, the exclusive grouping strategy fails to produce the valid distributed join results.

Federation engines like *FedX* [1], *HiBISCuS* [19], and *FedUp* [18] rely on either prior knowledge on the content of the federation members, or issue queries (such as ASK) to determine these groups.

For Hybrid Link Traversal to use SPARQL endpoints to compute joins locally, the engine should also identify exclusive groups. The problem is the unbounded nature of LTQP [20]. As during LTQP the query engine does not have access to all sources it will query over, exclusive groups can never be established. Even if we move to a finite web [20], the engine would have to first traverse all reachable [20] sources before exclusive groups can be determined. This is antithetical to the streaming nature of LTQP engines [16, 21] and will delay time to first result unacceptably. Thus, traditional approaches for determining exclusive groups are not applicable.

3.1.1. Research Avenues

Discoverable Indexes: Indexes can be effectively used to catalogue the data sources present in a dataspace. These indexes could be deployed as services, similar to approaches like ESPRESSO [14], pointing towards available source URIs and potentially exposing summary statistics. Using such indexes to quickly determine relevant sources, LTQP engines can apply traditional federation techniques to establish exclusive groups and perform source selection.

Caching: Similar to discoverable indexes, caches provide prior knowledge about the data sources present in a dataspace. Provided the cache maintains a degree of freshness and completeness, the query engine can perform a 'simulated traversal' offline over the cached data to identify relevant sources. Any gaps in this knowledge can then be supplemented by live traversal prior to or during execution.

Authoritativeness Assumptions: Establishing exclusive groups in a decentralized dataspace requires a notion of *source authority* [22]. Under standard Link Traversal, any reachable source can assert triples about any subject (e.g., a third-party source asserting `<P1> foaf:knows <P2>`), creating a "wild west" of data where no single source can be deemed the exclusive authority for a topic. This lack of authority prevents the query engine from determining if a set of patterns can be safely delegated to a single endpoint.

A possible solution is restricting the scope of validity such that a data source is the *sole authority* for triples where the subject corresponds to the source's own URI (or a URI within its controlled namespace). By enforcing this *Subject Authority Constraint*, the query engine can statically determine that all triples matching `<SourceA> ?p ?o` reside exclusively at *Source A*. This allows the engine to safely treat such patterns as an exclusive group, enabling the delegation of complex sub-queries (e.g., BGPs) to hybrid sources like SPARQL endpoints without the risk of incomplete results.

3.2. Integrating Dynamically Delegated Sub-queries into the Query Plan

When exclusive groups are identified and subqueries are dynamically pushed to expressive server-side interfaces, the resulting intermediate results must fit into the existing query plan.

In the traditional optimize-then-execute approach, used by Comunica-link-traversal [16, 17] and SQUIN [21] (except in [23]), the query plan is constructed before execution and evaluated over dynamically discovered sources.

Thus, an exclusive group can only be integrated if the precomputed plan already produces an intermediate result that exactly matches the triple patterns in that group.

As an example, consider a query plan with the join structure

$$(T_1 \bowtie T_2) \bowtie (T_3 \bowtie T_4).$$

If the engine discovers an exclusive group consisting of the triple patterns $\{T_2, T_3\}$, this group cannot be integrated into the plan. The plan never produces an intermediate result of the form $T_2 \bowtie T_3$, since these patterns occur in different subtrees. As a result, the exclusive group does not fit into the existing join structure and cannot be pushed as a single subquery.

3.2.1. Research Avenues

A clear path forward is the use of adaptive query processing techniques. These techniques should enable low-cost plan switching without discarding intermediate results, as the large number of sources in a dataspace would otherwise lead to plan thrashing. Algorithms such as Eddies [24], SteMs [25] (as used in [23]), and STAIRs [26] are viable candidates due to their tuple-level adaptivity. However, the literature provides no analysis of their performance characteristics in the context of LTQP, nor guidance on the design of adaptive routing strategies that determine how tuples are forwarded.

3.3. Discovery of Data in SPARQL endpoints

3.4. Alternative View Deduplication

Declaration on Generative AI

During the writing of this paper, the author(s) used DeepL and GPT-4o in order to: Grammar, translation and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] A. Schwarte, P. Haase, K. Hose, R. Schenkel, M. Schmidt, Fedx: Optimization techniques for federated query processing on linked data, in: The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I 10, Springer, 2011, pp. 601–616.

[2] M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte, T. Tran, Fedbench: A benchmark suite for federated semantic data query processing, in: The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I 10, Springer, 2011, pp. 585–600.

[3] L. Heling, M. Acosta, Federated sparql query processing over heterogeneous linked data fragments, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1047–1057.

[4] M.-H. Dang, J. Aimonier-Davat, P. Molli, O. Hartig, H. Skaf-Molli, Y. Le Crom, Fedshop: a benchmark for testing the scalability of sparql federation engines, in: International Semantic Web Conference, Springer, 2023, pp. 285–301.

[5] B. Esteves, H. J. Pandit, V. Rodríguez-Doncel, Odrl profile for expressing consent through granular access control policies in solid, in: 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, 2021, pp. 298–306.
[6] A. Padia, T. Finin, A. Joshi, et al., Attribute-based fine grained access control for triple stores, in: 3rd Society, Privacy and the Semantic Web-Policy and Technology workshop, 14th International Semantic Web Conference, 2015.
[7] A. Halevy, M. Franklin, D. Maier, Principles of dataspace systems, in: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2006, pp. 1–9.
[8] E. Curry, Dataspaces: fundamentals, principles, and techniques, in: Real-time Linked Dataspaces: Enabling Data Ecosystems for Intelligent Systems, Springer, 2019, pp. 45–62.
[9] R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, P. Colpaert, Triple pattern fragments: a low-cost knowledge graph interface for the web, Journal of Web Semantics 37 (2016) 184–206.
[10] G. Ladwig, T. Tran, Linked data query processing strategies, in: International Semantic Web Conference, Springer, 2010, pp. 453–469.
[11] M. M. Sabri, A hybrid framework for online execution of linked data queries, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 515–519.
[12] G. Ladwig, T. Tran, Sihjoin: Querying remote and local linked data, in: Extended Semantic Web Conference, Springer, 2011, pp. 139–153.
[13] J. Umbrich, A hybrid framework for querying linked data dynamically (2012).
[14] M. Ragab, Y. Savateev, R. Moosaei, T. Tiropanis, A. Poulovassilis, A. Chapman, G. Roussos, Espresso: a framework for empowering search on decentralized web, in: International Conference on Web Information Systems Engineering, Springer, 2023, pp. 360–375.
[15] S. Speicher, J. Arwe, A. Malhotra, Linked data platform 1.0, 2015. URL: https://www.w3.org/TR/ldp/, w3C Recommendation.
[16] R. Taelman, J. Van Herwegen, M. Vander Sande, R. Verborgh, Comunica: a modular sparql query engine for the web, in: International Semantic Web Conference, Springer, 2018, pp. 239–255.
[17] R. Taelman, R. Verborgh, Link traversal query processing over decentralized environments with structural assumptions, in: International Semantic Web Conference, Springer, 2023, pp. 3–22.
[18] J. Aimonier-Davat, B. Nédelec, M.-H. Dang, P. Molli, H. Skaf-Molli, Fedup: querying large-scale federations of sparql endpoints, in: Proceedings of the ACM Web Conference 2024, 2024, pp. 2315–2324.
[19] M. Saleem, A.-C. Ngonga Ngomo, Hibiscus: Hypergraph-based source selection for sparql endpoint federation, in: European semantic web conference, Springer, 2014, pp. 176–191.
[20] O. Hartig, J.-C. Freytag, Foundations of traversal based query execution over linked data, in: Proceedings of the 23rd ACM conference on Hypertext and social media, 2012, pp. 43–52.
[21] O. Hartig, Squin: a traversal based query execution system for the web of linked data, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 2013, pp. 1081–1084.
[22] B. Bogaerts, B. Ketsman, Y. Zeboudj, H. Aamer, R. Taelman, R. Verborgh, Distributed subweb specifications for traversing the web, Theory and Practice of Logic Programming 24 (2024) 394–420.
[23] O. Hartig, M. T. Özsu, Walking without a map: Ranking-based traversal for querying linked data, in: International Semantic Web Conference, Springer, 2016, pp. 305–324.
[24] R. Avnur, J. M. Hellerstein, Eddies: Continuously adaptive query processing, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 261–272.
[25] V. Raman, A. Deshpande, J. M. Hellerstein, Using state modules for adaptive query processing, in: Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405), IEEE, 2003, pp. 353–364.
[26] A. Deshpande, J. M. Hellerstein, Lifting the burden of history from adaptive query processing, in: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, 2004, pp. 948–959.