Towards Applications on the Decentralized Web using Hypermedia-driven Query Engines

Ruben Taelman
IDLab, Department of Electronics and Information Systems, Ghent University – imec, Ghent, Belgium

The Web is facing unprecedented challenges related to the control and ownership of data. Due to recent privacy and manipulation scandals caused by the increasing centralization of data on the Web into increasingly fewer large data silos, there is a growing demand for the re-decentralization of the Web. Enforced by user-empowering legislature such as GDPR and CCPA, decentralization initiatives such as Solid are being designed that aim to break personal data out of these silos and give back control to the users. While the ability to choose where and how data is stored provides significant value to users, it leads to major technical challenges due to the massive distribution of data across the Web. Since we cannot expect application developers to take up this burden of coping with all the complexities of handling this data distribution, there is a need for a software layer that abstracts away these complexities, and makes this decentralized data feel as being centralized. Concretely, we propose personal query engines to play the role of such an abstraction layer. In this article, we discuss what the requirements are for such a query-based abstraction layer, what the state of the art is in this area, and what future research is required. Even though many challenges remain to achieve a developer-friendly abstraction for interacting with decentralized data on the Web, the pursuit of it is highly valuable for both end-users that want to be in control of their data and its processing and businesses wanting to enable this at a low development cost.

1. INTRODUCTION

The Web as originally envisioned by Tim Berners-Lee [Berners-Lee 1989] was highly decentralized. Yet, in recent years, the Web has becoming increasingly centralized due to large Web companies gobbling up increasingly more data into fewer large data silos. While this centralization provides economic value to these handful of businesses, it makes it more difficult for smaller data-oriented businesses to gain a foothold, and it opens the door towards misuse of personal information and censorship. Due to the recurrence of the latter problems in recent years, privacy-related legislature such as the European General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) have been put into place. Since most businesses follow a centralized data silo model, a major data reformation is taking place. To provide a vehicle for this data reformation, various initiatives are being spawned that aim to *re-decentralize* data on the Web. These initiatives include Solid [Verborgh 2022], Bluesky [Bluesky 2023], and Mastodon [Zignani et al. 2018]. While Solid aims to enable a domain-agnostic decentralization ecosystem, Bluesky and Mastodon are focus on the social networking domain. Initiatives such as Solid enable everyone to have a personal data vault, where the user can choose where and how this data vault is hosted, and who can access what data within this vault. While the contemporary mode of data management leads to data silo owners being in control over data, these decentralization initiatives give full control of data to the users. Furthermore, instead of data being captured within a few large centralized data silos, decentralization initiatives lead to data being spread over a huge number of small personalized data vaults across the Web. Yet, these distributed data vaults are not isolated, but they are interlinked with each other. For example, links between vaults can exist to represent friendships [Brickley and Miller 2018] and hierarchies of comments on comments. Hence, this interlinking of data forms *Decentralized Knowledge Graphs* [Berners-Lee 2009]. This massive data distribution leads to a radical shift in how businesses manage data, which is coupled with new technical challenges, but also opens up new opportunities [Crum et al. 2024]. While most innovations are focused on centralized big data solutions, there is a need for decentralized small data solutions, which come with additional complexities due to its distribution aspect.

The Web enables applications to be built and in a multitude of forms, such as browser-based, native (desktop/mobile), back-end, ... Most of these applications today are centralized, which means that they interact with a single data silo. Large data silos are typically implemented through some form of database, which can be relational, document-oriented, graph-based, ... These databases could even run in the cloud or be distributed, but they are usually accessable through a central access point or Web API. Common among these databases, is that they offer access to the data in the form of a declarative query language, such as SQL, GraphQL [Foundation], Cypher [Francis et al. 2018], SPARQL [Harris et al. 2013], ... Through this query language, databases can abstract away complexities for searching and updating data, such as using multiple internal indexes, or joining between tables. As such, application developers only need to be concerned with writing queries and passing them to the database management system, after which the query results can be used within the application.

Applications on the decentralized Web do not just have to interact with a single data silo, but they have to integrate data from across a potentially huge number of data vaults that are spread over the Web. Similar to databases, data from across these vaults needs to be joined together, and there could be a need for intermediary Web-based indexes to speed up the search process. Just like we do not expect application developers to directly interact with database internals such as tables and indexes, neither can we expect these developers to find and integrate data from across vaults manually. Hence, there is a need for hiding these complexities surrounding decentralized data management behind an abstraction layer. Similar to data silos, this abstraction layer can exist in the form of a declarative query language, which gives application developers a familiar way to interact with decentralized data. Such queries then need to be processed by reusable engines that convert these queries into multiple Web requests across data vaults. To cope with the heterogeneity of data vaults caused by the different needs and facilities of Web users, query engines needs to be hypermedia-driven for discovering and using vault capabilities during query processing.

In this article, we discuss which requirements exist for systems to query the decentralized Web. Furthermore, we discuss which requirements can already be met with existing techniques, and what future research is still needed.

2. A QUERY ENGINE FOR THE DECENTRALIZED WEB

In practise, different types of Web-driven applications can exist, each of which interact with data on the Web. First, the most obvious type is a traditional *user-facing browser-based application*, that runs in the Web browser using browser-provided APIs, typically just for the duration of the user's session. Second, we consider the *user-facing browserless application*, which concerns applications running outside of the browser, such as natively on your mobile device, a personal computer, or a voice-enabled system. Third, there is the *non-interactive application*, which does not offer any direct user interaction, such as autonomous Web crawlers or other types of agents. In order to build such applications across decentralized data, there is a need for query engines that can cope with the complexities surrounding decentralization. Below, we discuss these requirements, which are generalized from requirements specific to the Solid environment [Taelman 2024].

2.1 Execution of declarative queries

In order to provide a fixed and well-defined contract between the application developer and the query engine, there is a need for a declarative query language. Declarativity is key here, as it enables the application developer to specify *what* data is needed or needs to be updated, without having to specify *how* and from where the data must be fetched and combined. As such, the responsibility of determining the "*how*" is shifted to the query engine, which becomes responsible for finding data and computing results. Furthermore, this query language must be sufficiently expressive to handle a wide domain of questions that applications over decentralized data may require.

2.2 Live discovery of data sources

Due to the potentially unbounded number of data sources over which data may be spread in a decentralized environment, it is unfeasible to always assume application developers have prior knowledge of sources relevant for each query. Furthermore, data within a decentralized environment may be constantly evolving, and may even move across data sources over time. Therefore, we can not assume application developers to be able to define all relevant sources or their interconnectivity. As such, a query engine must be capable of discovering query-relevant sources on its own. This could involve deriving sources from hints provided by the application developer, such as the current user for using-facing applications, caches, and the current query. If these sources contain pointers to other sources, the query engine may decide to recursively follow these pointers if they are deemed relevant to the query. When multiple queries are executed in sequence, the query engine may exploit cache-based techniques, on the condition that data sources explicitly indicate their cacheability, e.g. through HTTP caching headers.

2.3 Handling API heterogeneity

Since decentralized environments may contain data sources in different forms, there will not be a single API for data sources that is optimal for all different use cases. For example, data served on low-end machine might benefit from a low-complexity API, while data served on a high-end might offer more capabilities through its API. The capabilities of this API may even evolve over time if server capabilities or server load change. As such, query engines will need to cope with the heterogeneous nature of APIs within decentralized environments. In order for query engine to dynamically detect the capabilities of each API and how to use them during query processing, APIs will need to be self-descriptive using hypermedia description languages such as Hydra [Lanthaler and Gütl 2013].

2.4 Handling data model heterogeneity

Data exposed by different data sources may not only be heterogeneous in their API, but also in terms of their data model. For example, some data may be exposed as RDF [Cyganiak et al. 2014], while other data may be exposed using the property graph model [Rodriguez and Neubauer 2010]. Besides the data model, sources using the same data model may also offer data using different schemas. For example, RDF-based data sources may offer data using different RDF vocabularies. In order to combine data from across these heterogeneous data models and schemas during query execution, there will be a need for model and schema alignment techniques [Kifer 2008].

2.5 Authentication

Besides open data, data sources within decentralized environments may also contain private and personal data that require privacy assurances such as authentication and authorization. Query engines that back user-facing applications can therefore require authentication mechanisms to enable the query engine to authenticate itself to sources on behalf of the user. Non-interactive applications on the other hand can benefit from agent-based authentication. This allows the query engine to determine a personalized view over the decentralized environment, which means that the query engine can access anything the user can access for the duration of the application session.

2.6 Personalized configuration

Next to authentication, query engines should allow users to customize how the engine operates, how it handles data sources, and how it combines intermediary results. This can for example include defining priorities of data sources, e.g. preferring UK-based data sources over USA-based data sources. This is similar to Web browsers being personalizable, such as configuring default languages and automatic translations, how to open specific files, and how to cache content.

2.7 User-perceived performance

Decentralized applications should be usable with a sufficient level of user-perceived performance [Nielsen 1993]. As such, when a user performs an action that translates to a query execution, the user should be served with at least a partial response based on query results in the order of seconds, so that user attention is kept within the application.

2.8 Introspection of results

In order to provide sufficient levels of trust to the user about how results were computed, there is a need to inspect the provenance trail of query results. This should allow users to inspect from what sources the results originate from, and how they were combined. This enables auditing of results, which may be necessary for privacy-concerning legislation such as GDPR and CCPA, and the more recent European AI act.

3. SOLUTIONS AND CHALLENGES

In this section, we discuss to what respect the envisioned query engines over decentralized environments can already be achieved. We discuss each requirement introduced above separately in terms of what the state of the art is, and what challenges remain.

3.1 Execution of declarative queries

A multitude of declarative languages have been introduced that may be used to access decentralized environments. Most of these languages enable read and write capabilities. For example, SQL has been proposed for querying over the Web many years ago [Mendelzon et al. 1996]. The main downside of SQL is that it assumes a centralized data model and schema, which makes integration in a decentralized environment difficult. A query language that has been specifically designed for handling graph-based distributed data is the W3C-recommended SPARQL language [Harris et al. 2013]. While the SPARQL language is most commonly used for accessing centralized SPARQL endpoints [Feigenbaum et al. 2013], it is also possible to use it for querying multiple sources [Schwarte et al. 2011; Hartig 2013]. Besides SPARQL, there are other graph-based query languages such as Cypher [Francis et al. 2018], GraphQL [Foundation] and GQL [Deutsch et al. 2022]. To enable improved levels of developer experience, Object Relational Mapping-like libraries may be built on top of these languages to simplify common tasks [Verborgh and Taelman 2020].

3.2 Live discovery of data sources

While plenty of research has been carried out in the area of federated querying [Schwarte et al. 2011], existing federation approaches assume prior knowledge of data sources, which means that additional sources can not be discovered on the fly. A less known querying paradigm based on the Linked Data principles [Berners-Lee 2009] is called Link Traversal Query Processing (LTQP) [Hartig 2013], which allows data sources to be discovered on the fly *during* query execution by following Linked Data links between data sources. While LTQP has been proven to work over the Solid environment [Taelman and Verborgh 2023], it still suffers from performance limitations caused by the large number of links being followed and ineffective query planning. More precise link discovery techniques, better link handling [Eschauzier et al. 2023], collaborative querying [Tam 2023], and adaptive query planning [Hanski 2023] can help solve these problems.

3.3 Handling API heterogeneity

The requirement of handling heterogeneous APIs has seen very limited attention within research. Within the RDF data model, a formal language [Cheng and Hartig 2020] has been created, and some works have focused on performance improvements [Heling and Acosta 2022; Montoya et al. 2018], Most of these works however assume prior knowledge of API capabilities, whereas only limited work has been done on interpreting these capabilities on the fly [Taelman et al. 2018].

3.4 Handling data model heterogeneity

Handling heterogeneous data models has received some more research attention, such as code-generation based approaches [Karpathiotakis et al. 2016] and schema alignment techniques [Kifer 2008]. The main open problem here lies in how to efficiently apply these alignment techniques to incomplete and partial data, which are being discovered on the fly.

3.5 Authentication

Various techniques for querying while taking into account authentication and authorization have been investigated [De Capitani di Vimercati et al. 2011; Gabillon and Letouzey 2010; Capadisli 2022]. As such, what remains for this requirement are mainly concrete implementations.

3.6 Personalized configuration

In order to provide trust and control to users, there is a need for query engines to be configurable. There has been a vast amount of research around personal recommendation systems [Ko et al. 2022] to enable users to prefer certain data sources or query results over others. Furthermore, techniques that model browsing behaviour [Park and Fader 2004] may help to guide discovery and optimize caching within query engines [Eschauzier 2024].

3.7 User-perceived performance

Performance has been one of the primary concerns within query optimization research. Most of this research focuses on reducing total query execution time, which measures the time it takes from the start of execution, until the final result is produced. This metric can be too strict for measuring the user-perceived performance, since the display of content placeholders or partial results can desire user needs even before results are complete [Sebrechts 2019]. As such, more work is needed into not just optimizing total query execution time, but also enabling partial and early results [Acosta et al. 2017].

3.8 Introspection of results

Enabling trust and result audit can be achieved through exposing provenance trails of how query results came to be. Unfortunately, this area of research has received little research attention. Current techniques focus on query rewriting approaches [Hernández et al. 2021], but native support in query engines is lacking.

4. CONCLUSIONS

Reusable query engines can help reducing development effort and time for applications over decentralized data. Due to the radical shift in how data needs to be processed compared to centralized applications, specific requirements arise, which are not fully met today as shown in this article. While several systems have been introduced that meet partial requirements [Taelman et al. 2018; Taelman and Verborgh 2023; Ragab et al. 2023; Vandenbrande et al. 2023], none of these systems offer full solutions [Taelman 2024].

In the future, we foresee query engines playing a significant role in the decentralization ecosystem. For user-facing browser-based application, browser-specific and highly personalizable query engines can be built in, which may be exposed through a standardized Web browser API. For user-facing browserless and non-interactive applications, query engines may be available through open and commercial libraries, some of which could even be derived from browser engines, similar to how many server-side JavaScript runtime environments such as Node.js and Deno are based on browser engines. Even for recent advances in Generative AI, these query engines provide significant value in terms of accuracy and trustworthiness through RAG-based techniques [Lewis et al. 2020].

ACKNOWLEDGMENTS

This research was supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10). Ruben Taelman is a postdoctoral fellow of the Research Foundation – Flanders (FWO) (1202124N).

REFERENCES

ACOSTA, M., VIDAL, M.-E., AND SURE-VETTER, Y. 2017. Diefficiency metrics: measuring the continuous efficiency of query processing approaches. In *International Semantic Web Conference*. Springer, 3–19.

BERNERS-LEE, T. 2009. Linked data. Design Issues.

BERNERS-LEE, T. J. 1989. Information management: A proposal. Tech. rep.

BLUESKY. 2023. Bluesky. URL: https://blueskyweb.xyz/.

BRICKLEY, D. AND MILLER, L. 2018. Foaf (friend of a friend). URL: www.foaf-project.org.

CAPADISLI, S. 2022. Web access control. Editor's draft, Solid. July.

CHENG, S. AND HARTIG, O. 2020. Fedqpl: A language for logical query plans over heterogeneous federations of rdf data sources. In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*. 436–445.

CRUM, E., TAELMAN, R., BUELENS, B., ERTAYLAN, G., AND VERBORGH, R. 2024. Personalized medicine through personal data pods. In *Proceedings of the 15th International SWAT4HCLS Conference*.

CYGANIAK, R., WOOD, D., AND LANTHALER, M. 2014. Rdf 1.1: Concepts and abstract syntax. Recommendation, W3C. Feb.

DE CAPITANI DI VIMERCATI, S., FORESTI, S., JAJODIA, S., PARABOSCHI, S., AND SAMARATI, P. 2011. Authorization enforcement in distributed query evaluation. *Journal of Computer Security 19,* 4, 751–794.

DEUTSCH, A., FRANCIS, N., GREEN, A., HARE, K., LI, B., LIBKIN, L., LINDAAKER, T., MARSAULT, V., MARTENS, W., MICHELS, J., ET AL. 2022. Graph pattern matching in gql and sql/pgq. In *Proceedings of the 2022 International Conference on Management of Data*. 2246–2258.

ESCHAUZIER, R. 2024. Personalized query engine optimization for link traversal-based query processing over structured decentralized environments. In *European Semantic Web Conference*. Springer.

ESCHAUZIER, R., TAELMAN, R., AND VERBORGH, R. 2023. How does the link queue evolve during traversal-based query processing? In *Proceedings of the 7th International Workshop on Storing, Querying and Benchmarking Knowledge Graphs*.

FEIGENBAUM, L., TODD WILLIAMS, G., GRANT CLARK, K., AND TORRES, E. 2013. SPARQL 1.1 protocol. Rec., W3C. Mar.

FOUNDATION, T. G. Graphql. october 2021 edition.

FRANCIS, N., GREEN, A., GUAGLIARDO, P., LIBKIN, L., LINDAAKER, T., MARSAULT, V., PLANTIKOW, S., RYDBERG, M., SELMER, P., AND TAYLOR, A. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data*. 1433–1445.

GABILLON, A. AND LETOUZEY, L. 2010. A view based access control model for sparql. In *2010 Fourth International Conference on Network and System Security*. IEEE, 105–112.

HANSKI, J. 2023. Optimisation of link traversal query processing over distributed linked data through adaptive techniques. In *European Semantic Web Conference*. Springer, 266–276.

HARRIS, S., SEABORNE, A., AND PRUD'HOMMEAUX, E. 2013. SPARQL 1.1 query language. Recommendation, W3C. Mar.

HARTIG, O. 2013. An overview on execution strategies for linked data queries. *Datenbank-Spektrum 13,* 2, 89–99.

HELING, L. AND ACOSTA, M. 2022. Federated sparql query processing over heterogeneous linked data fragments. In *Proceedings of the ACM Web Conference 2022*. 1047–1057.

HERNÁNDEZ, D., GALÁRRAGA, L., AND HOSE, K. 2021. Computing how-provenance for sparql queries via query rewriting. *Proceedings of the VLDB Endowment 14,* 13, 3389–3401.

KARPATHIOTAKIS, M., ALAGIANNIS, I., AND AILAMAKI, A. 2016. Fast queries over heterogeneous data through engine customization. *Proceedings of the VLDB Endowment 9,* 12, 972–983.

KIFER, M. 2008. Rule interchange format: The framework. In *International Conference on Web Reasoning and Rule Systems*. Springer, 1–11.

KO, H., LEE, S., PARK, Y., AND CHOI, A. 2022. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics 11,* 1, 141.

LANTHALER, M. AND GÜTL, C. 2013. Hydra: A vocabulary for hypermedia-driven Web apis. In *Proceedings of the 6th Workshop on Linked Data on the Web.*

LEWIS, P., PEREZ, E., PIKTUS, A., PETRONI, F., KARPUKHIN, V., GOYAL, N., KÜTTLER, H., LEWIS, M., YIH, W.-T., ROCKTÄSCHEL, T., ET AL. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems 33*, 9459–9474.

MENDELZON, A. O., MIHAILA, G. A., AND MILO, T. 1996. Querying the world wide web. In *Fourth International Conference on Parallel and Distributed Information Systems*. IEEE, 80–91.

MONTOYA, G., AEBELOE, C., AND HOSE, K. 2018. Towards efficient query processing over heterogeneous rdf interfaces. In *2nd Workshop on Decentralizing the Semantic Web, DeSemWeb 2018*. CEUR Workshop Proceedings.

NIELSEN, J. 1993. Response times: the three important limits. *Usability Engineering*.

PARK, Y.-H. AND FADER, P. S. 2004. Modeling browsing behavior at multiple websites. *Marketing Science 23,* 3, 280–303.

RAGAB, M., SAVATEEV, Y., MOOSAEI, R., TIROPANIS, T., POULOVASSILIS, A., CHAPMAN, A., AND ROUSSOS, G. 2023. Espresso: A framework for empowering search on decentralized web. In *International Conference on Web Information Systems Engineering*. Springer, 360–375.

RODRIGUEZ, M. A. AND NEUBAUER, P. 2010. Constructions from dots and lines. *arXiv preprint arXiv:1006.2361*.

SCHWARTE, A., HAASE, P., HOSE, K., SCHENKEL, R., AND SCHMIDT, M. 2011. Fedx: Optimization techniques for federated query processing on linked data. In *International semantic web conference*. Springer, 601–616.

SEBRECHTS, I. 2019. Usability of distributed data sources for modern web applications.

TAELMAN, R. 2024. Requirements and challenges for query execution across decentralized environments. In *Companion Proceedings of the ACM Web Conference 2024*.

TAELMAN, R., VAN HERWEGEN, J., VANDER SANDE, M., AND VERBORGH, R. 2018. Comunica: a modular sparql query engine for the web. In *Proceedings of the 17th International Semantic Web Conference*.

TAELMAN, R. AND VERBORGH, R. 2023. Link traversal query processing over decentralized environments with structural assumptions. In *Proceedings of the 22nd International Semantic Web Conference*.

TAM, B.-E. 2023. Introducing collaborative link traversal query processing in the context of structured decentralized environments. In *ISWC2023, the International Semantic Web Conference*.

VANDENBRANDE, M., JAKUBOWSKI, M., BONTE, P., BUELENS, B., ONGENAE, F., AND VAN DEN BUSSCHE, J. 2023. Pod-query: Schema mapping and query rewriting for solid pods. In *ISWC2023, the International Semantic Web Conference*.

VERBORGH, R. 2022. Re-decentralizing the Web, for good this time. In *Linking the World's Information: A Collection of Essays on the Work of Sir Tim Berners-Lee*, O. Seneviratne and J. Hendler, Eds. ACM.

VERBORGH, R. AND TAELMAN, R. 2020. LDflex: a read/write Linked Data abstraction for front-end Web developers. In *Proceedings of the 19th International Semantic Web Conference*.

ZIGNANI, M., GAITO, S., AND ROSSI, G. P. 2018. Follow the mastodon: Structure and evolution of a decentralized online social network. In *Twelfth International AAAI Conference on Web and Social Media*.

Ruben Taelman is a Postdoctoral Researcher at IDLab, Ghent University – imec, Belgium. His research concerns the decentralized publication and querying of Linked Data on the Web, and investigating the trade-offs that exist between server and client. Next to that, he is actively applying his research by developing reusable software that can be used by developers and other researchers.