Opportunities for Shape-Based Optimization of Link Traversal Queries

Bryan-Elliott Tam[1,*], Ruben Taelman[1], Pieter Colpaert[1] and Ruben Verborgh[1]

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

Abstract

Data on the web is naturally unindexed and decentralized. Centralizing web data, especially personal data, raises ethical and legal concerns. Yet, compared to centralized query approaches, decentralization-friendly alternatives such as Link Traversal Query Processing (LTQP) are significantly less performant and understood. The two main difficulties of LTQP are the lack of apriori information about data sources and the high number of HTTP requests. Exploring decentralized-friendly ways to document unindexed networks of data sources could lead to solutions to alleviate those difficulties. RDF data shapes are widely used to validate linked data documents, therefore, it is worthwhile to investigate their potential for LTQP optimization. In our work, we built an early version of a source selection algorithm for LTQP using RDF data shape mappings with linked data documents and measured its performance in a realistic setup. In this article, we present our algorithm and early results, thus, opening opportunities for further research for shape-based optimization of link traversal queries. Our initial experiments show that with little maintenance and work from the server, our method can reduce up to 80% the execution time and 97% the number of links traversed during realistic queries. Given our early results and the descriptive power of RDF data shapes it would be worthwhile to investigate non-heuristic-based query planning using RDF shapes.

Keywords

Linked data, Link Traversal Query Processing, Query containment, RDF data shapes, Descentralized environments

1. Introduction

The World Wide Web is a naturally decentralized database. Centralizing large web segments in single endpoints provides easier query interfaces and faster query execution times. However, data centralization can lead to practices that raise ethical and legal concerns, making the exploration of decentralization-friendly query paradigms a relevant research topic. The query languages webSQL [1] and SPARQL propose mechanisms to capture decentralized web data with conjunctive queries. However, webSQL relies on web indexing [1]. Indexing processes can be expensive, particularly on the scale of the web, and necessitate frequent updates, furthermore, they can be restrictive by excluding some sources thus hindering the natural serendipity of the web. SPARQL solutions rely on the publication of linked data. Linked data in their structure particularly with the presence of IRI gives the opportunity to find more related information without indexes. However, most query processing over linked data is performed in centralized and federated setups, leaving indexing-independent approaches largely experimental.

Link Traversal Query Processing (LTQP) [2] is a method to query unindexed networks of linked data documents. The method consists of answering a query using an evolving triple store. This evolving triple store is continuously updated with data acquired by the query engine through the recursive dereferencing of IRIs from the store. The process is started with a set of IRIs provided by the user to the engine. While LTQP enables live exploration of environments without prior indexing, it leads to some difficulties. One of them is the pseudo-infinite search domain derived from the size of the World Wide

AMW 2024: 16th Alberto Mendelzon International Workshop on Foundations of Data Management, September 30th–October 4th, 2024, Mexico City, Mexico

[*] Corresponding author.

bryanelliott.tam@ugent.be (B. Tam); ruben.taelman@ugent.be (R. Taelman); pieter.colpaert@ugent.be (P. Colpaert); ruben.verborgh@ugent.be (R. Verborgh)

https://www.rubensworks.net (R. Taelman); https://pietercolpaert.be (P. Colpaert); https://ruben.verborgh.org (R. Verborgh)

0000-0003-3467-9755 (B. Tam); 0000-0001-5118-256X (R. Taelman); 0000-0001-6917-2167 (P. Colpaert); 0000-0002-8596-222X (R. Verborgh)

Web [3]. Additionally, HTTP requests can be very slow and unpredictable making their execution the bottleneck of the method [3]. Reachability criteria [2] are a partial answer to this problem by defining completeness based on the traversal of URIs contained in the internal data source of the engine instead of on the acquisition of all the results or the traversal of the whole web. Another difficulty is the lack of a priori information about the sources rendering query planning challenging. To alleviate this problem, the current state-of-the-art consists of using carefully crafted heuristics for joins ordering [4]. The limitations of the heuristics approach are usually of little importance because the main bottleneck is the high number of HTTP requests.

Earlier LTQP research has focused on the open web. More recently, LTQP research has shifted its focus to environments where the structure of data publication provides useful information for query optimization. This line of research uses *structural assumptions* [5] to guide query engines [6] towards relevant data sources. Structural assumptions act as contracts between the data provider and the query engines stipulating that within a certain subdomain of the web, information meeting a specific constraint can be found. The use of structural assumptions has been studied in Solid [5]. The method involves the utilization of the solid storage hypermedia description [7] to locate all the resources of a pod. This hypermedia description is not expressive enough to capture the content of the resources of a pod, thus, for query-aware optimizations, the type index specification [1] is additionally used. The type index formulation proposes a more declarative approach [8] by mapping RDF classes with sets of resources. By using those structural assumptions it is possible to reduce the query execution time of realistic queries to the extent where the bottleneck is not the execution of HTTP requests but the suboptimal heuristic-based query plan [9, 5]. Yet, for multiple queries the high number of HTTP requests remains the main bottleneck [9]. It is reasonable to hypothesize that a significant portion of those HTTP requests lead to the dereferencing of documents containing data that do not contribute to the result of the query. Hence, investigating more descriptive structural assumptions is a relevant research endeavor.

In this article, we propose to use RDF data shapes as the main mechanism for a structural assumption in the form of a shape index. RDF data shapes are mostly used in data validation [10] thus, they provided a good formalization to describe the structure of data. Additionally, to a lesser extent, they have been used for query optimizations [11]. The shape index is an early effort for data summarisation of decentralized datasets [12, 13, 14] within networks of unindexed linked data documents. The current focus of the index is source selection. However, we foresee opportunities to use a similar approach for link queue ordering and query planning. This paper presents our preliminary work on data discovery and link pruning thus tackling the problem of the large search space of LTQP queries in linked data environments with structure.

2. Shape Index and Query-Shape Containment

The RDF specification does not enforce schemas on data. However, the data published often adhere to an implicit schema due to the nature of its modeled object and the formulation of RDF [15]. From those observations, implicit schemas have been used with success for query optimization [15, 16]. We propose to adapt those methods for LTQP by using explicit data schemas provided by the data provider in our source selection process.

2.1. Shape Index

Our method introduces the concept of a *shape index* to reduce query execution time by minimizing unnecessary dereferencing of RDF documents within web subdomains (sets of URLs or URL patterns). [2] We define a shape index as a set of mappings between RDF data shapes and sets of resources. This mapping concept shares similarities with shape mapping [17] and target declarations [18]. However,

[1] https://solid.github.io/type-indexes/

[2] 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.

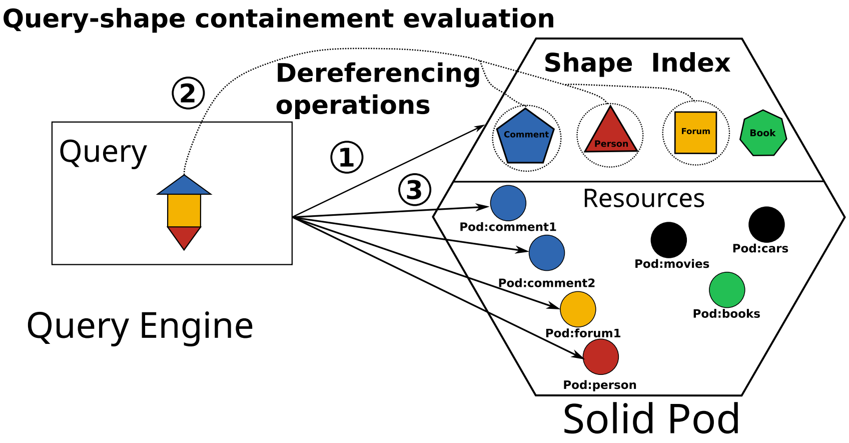

Figure 1: First, the shape index is dereferenced, then the *query-shape containment* operations are performed in the query engine and lastly, only the relevant resources are dereferenced.

instead of mapping shapes to RDF subgraphs, the shape index maps shapes to sets of documents. The shape index also shares commonalities with shape trees, [3] however, it is designed to be a simpler formulation focused on query optimization. The shape index has a range of applications defined in a domain and a flag indicating if the index is *complete*. A shape index is complete when every resource in the domain is associated with a shape within the shape index. In a shape index when a shape is mapped to a set of RDF resources then the shape *must* validate those resources. Furthermore, every set of triples respecting the shape in the domain *must* be located inside one resource of the set.

2.2. Query-Shape Containment

In order to determine before the traversal of a whole subdomain which resources are useful and which can be pruned, the query engine solves a *query-shape containment* problem over the shape of the index analogous to the classic query containment problem [19, 20, 21]. The query containment problem consists of determining independent of the data source if the results of a query will be a subset of the results of another query. We propose to express RDF data shapes into SELECT SPARQL queries (Q_s) [22, 23, 24, 25] and apply similar resolution methods to query containment problems. Due to the explicit domain definition of the index, this approach is adaptative, thus, the query engine can start its processing with permissive reachability criteria such as c_{all} [2] or the Solid state-of-the-art reachability criteria [5] and not suffer from the associated longer execution time during the traversal of environments containing a shape index. The source selection process is schematized for a single (sub)domain in Figure 1. The process starts with the discovery of the shape index in the current (sub)domain. In the case of Solid, the index can be at the root of the pod to be easily discoverable. [4] After the dereferencing of the index, the analysis is started inside the query engine. The analysis consists of interpreting the binding results (homomorphism and "partial" homomorphism) of the *query-shape containment* problem. The algorithm divides the query from the user into multiple star patterns with their dependent star pattern (Q_{star}). After the division of the query, the queries are pushdown [12, 26] to the level of source selection to evaluate if the Q_{star} are contained inside the Q_s of the shape index. If all the Q_{star} are contained in a Q_s or have no binding with any Q_s the reachability criterion is adapted to ignore all the resources not linked to a Q_s even if the shape index is *incomplete*. If the shape index is *complete* and not all the Q_{star} are contained in a Q_s the reachability criterion can be adapted to visit every resource in relation to a Q_s with a partial binding with a Q_{star}. In a similar case with an *incomplete* shape index, the query engine can only use the shape index for data discovery. This case is similar to the usage of the type index but with a more reaching ability to match a query with the index because shapes in their definition describe the properties (RDF predicates) of the entities whereas the type index only provides

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

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

the classes IRIs. It is possible to dereference the class IRIs to get information about the properties (if available), however, it is not the current practice [5]. A comparison of the RDF data shapes and RDF class approach due to their potential similarities is delegated to future works. [5]

2.3. A Concrete Example

We conclude this section with a concrete example. Let's assume that a user wants to retrieve the IDs and contents of the comments in a network along with the forums ID where they have been posted and the name of the moderator of the forums. This query is schematized in Figure 1. The query *can* be represented by three star pattern queries, the comment $Q_{comments}$, the forum Q_{forums}, and the moderator $Q_{moderators}$. The full query is formed by the join of those star patterns, where the joins respect the dependencies defined by shared variables $Q = Q_{comments} \bowtie Q_{forums} \bowtie Q_{moderators}$. When traversing the network the query engine cannot know the content of the documents encountered, therefore, the engine *must* deference every reachable document as defined by a reachability criterion. The presence of a shape index can change the state of affairs. If the engine encounters a domain containing exclusively book data as indicated by a complete shape index, the engine can skip the documents of the domain. If a domain has comment and movie review data declared by a complete shape index, the query engine can safely limit its dereferencing operations to the set of documents related to comments without affecting results completeness. The engine can restrict its dereferencing operations because at least one star pattern is contained in the comment shape and none in the movie review and book shapes. If the engine encounters a domain regardless of the completeness of the index, declaring comment, forum, and individual (moderators are individuals/people) data, among others, then the documents associated with the non-query-relevant part of the domain can be ignored with the same containment logic presented earlier. Thus, we can consider that the traversal proceeds domain by domain ignoring documents known a priori to not content query-relevant data.

3. Preliminary Results

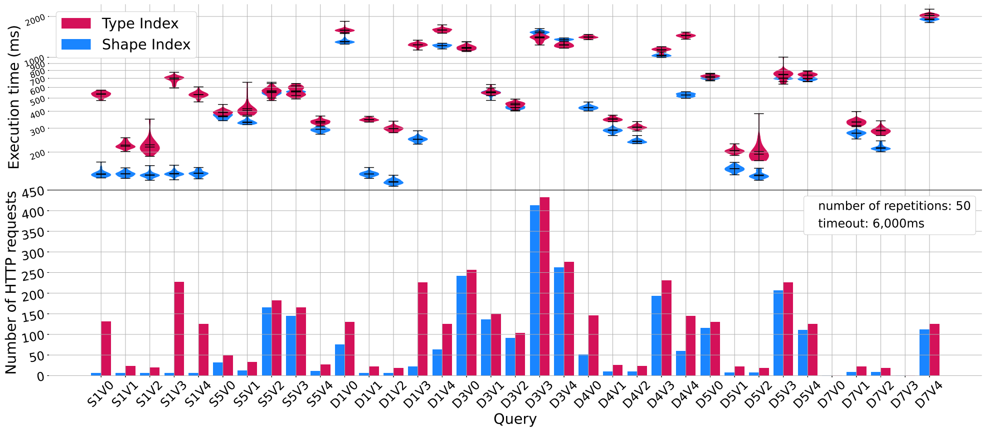

Figure 2: The execution time with shape indexes is consistently lower (up to 80% with D1V3 and S1V3) or equal to that with the type indexes (except for D3V3 and D3V4), and always uses fewer HTTP requests. The queries are denoted with first the initial of the query template (e.g., S1 for interactive-short-**1**), and the version of the concrete query (e.g., V0). Values not present in the plot (D7V0 and D7V3) indicate that the query timeout before the end of the execution.

An open-source implementation of the algorithm and an integration in the query engine Comunica

[5] There exist comparisons between the shape and class definition approaches in the context of data validation [27] but it is left to be determined if their frame of comparison is compatible with our current problem and foreseen opportunities.

[28] is available online. [6] We use the benchmark Solidbench [5] to compare our approach with the current state-of-the-art (the type index and the LDP specification [7] as structural assumptions) [5]. We used the supported subset of SolidBench queries, skipping the currently unimplemented SPARQL property paths [8] and unions. We executed each query 50 times with a timeout of 1 minute (6,000 ms). Figure 2 shows that the reduction can be as high as 80% (D1V3 and S1V3) for execution time and 97% (S1V3) for the number of HTTP requests. Our approach reliably executes fewer HTTP requests compared to the state-of-the-art. This is an expected result because no queries target (implicitly) each file of a user. The shape index approach requests a subset of the request of the type index approach (without sacrificing query results) with the addition of the request to get the shape definitions which leads in general to the dereferencing of a small number of short documents. There is not a direct correlation between the reduction of execution time and HTTP requests (e.g., the ratio between our approach and the state-of-the-art of the number of HTTP requests by the execution time for D1V3 is 0.5 compared to 0.15 for S1V3). This hints at the results from the state-of-the-art [5] proposing that the query plan is the bottleneck for some queries in this environment, however, the overhead of the containment calculation could also be a contributing factor to the current results. In the worst cases, our approach has similar query execution to the state-of-the-art except for D3V3 and D3V4 with an increase of 9% of the mean of the execution time. The variance of the execution with a shape index tends to be lower compared to the type index. A possible explanation for this observation is that the execution time of HTTP requests is unpredictable [3] leading to an increase in variance. This observation not only has potential implications for the reliability of multiple executions in terms of execution time but also in terms of the performance of single executions in unstable networks where the server might take longer times to respond.

4. Conclusion

The shape index approach shows that more precise source selection in LTQP can significantly reduce query execution time. Although it is still an early effort, we believe that a solution inspired by our approach could be beneficial for the query and publication of fragmented document-based linked data. Our solution does not require extensive computational power from the data publisher during queries and updates [9] of data sources. Additionally, using a shape index holds promise to improve the data quality of fragmented document-based linked data. There are still multiple questions left to be answered such as how to handle private data, what is the overhead and the complexity of the method [10], does the reduction of HTTP request or the reduction on the size of the internal triple store has more impact on the performance, can the shape index alone or with other data summarisation structures be used to improve query planning without sacrificing query execution times.

Acknowledgements

Supported by SolidLab Vlaanderen (Flemish Government VV023/10) and the Research Foundation Flanders (FWO) under grant number S006323N.

[6] The algorithm implementation is available at the following link https://github.com/constraintAutomaton/query-shape-detection and the integration in the Comunica query engine at the following link https://github.com/constraintAutomaton/comunica-feature-link-traversal/tree/feature/shapeIndex. The implementation of the benchmark and complementary results such as the analysis of the statistical significance are available at the following link https://github.com/constraintAutomaton/amw_shape_index_results.

[7] https://www.w3.org/TR/ldp/

[8] https://www.w3.org/TR/sparql11-query/#propertypaths

[9] Considering no change in the data model.

[10] Given the expressiveness of RDF data shapes language [22, 29, 30] and practice in shape definitions [31, 29, 32].

References

[1] A. Mendelzon, G. Mihaila, T. Milo, Querying the world wide web, in: Fourth International Conference on Parallel and Distributed Information Systems, 1996, pp. 80–91. doi:10.1109/PDIS.1996.568671.

[2] O. Hartig, J.-C. Freytag, Foundations of traversal based query execution over linked data, in: Conference on Hypertext and Social Media, HT '12, ACM, New York, NY, USA, 2012, p. 43–52. doi:10.1145/2309996.2310005.

[3] O. Hartig, 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, Springer, Berlin, Heidelberg, 2011, pp. 154–169.

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

[6] R. Verborgh, R. Taelman, Guided link-traversal-based query processing, 2020. arXiv:2005.02239.

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

[8] R. Taelman, R. Verborgh, Declaratively describing responses of hypermedia-driven web apis, in: Knowledge Capture Conference, K-CAP '17, Association for Computing Machinery, New York, NY, USA, 2017. doi:10.1145/3148011.3154467.

[9] R. Eschauzier, R. Taelman, R. Verborgh, How does the link queue evolve during traversal-based query processing?, in: Proceedings of the 7th QuWeDa, CEUR Workshop Proceedings, 2023.

[10] J.-E. L. Gayo, E. Prud'hommeaux, I. Boneva, D. Kontokostas, Validating RDF Data: Applications, Springer International Publishing, Cham, 2018, pp. 195–231. doi:10.1007/978-3-031-79478-0_6.

[11] K. Rabbani, M. Lissandrini, K. Hose, Optimizing sparql queries using shape statistics, 2021. doi:10.5441/002/EDBT.2021.59.

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

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

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

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

[16] M. Meimaris, G. Papastefanatos, Hierarchical characteristic set merging for optimizing sparql queries in heterogeneous rdf, ArXiv abs/1809.02345 (2018).

[17] J.-E. L. Gayo, E. Prud'hommeaux, I. Boneva, D. Kontokostas, Shape Expressions, Springer International Publishing, Cham, 2018, pp. 55–117. doi:10.1007/978-3-031-79478-0_4.

[18] J.-E. L. Gayo, E. Prud'hommeaux, I. Boneva, D. Kontokostas, SHACL, Springer International Publishing, Cham, 2018, pp. 119–194. URL: https://doi.org/10.1007/978-3-031-79478-0_5. doi:10.1007/978-3-031-79478-0_5.

[19] R. C. Foto Afrati, Query Containment and Equivalence, Springer Cham, 2019, pp. 21–59. doi:https://.doi.org/10.1007/978-3-031-01871-8.

[20] M. Spasić, M. V. Janičić, Solving the SPARQL query containment problem with SpeCS, Journal of

Web Semantics 76 (2023) 100770. doi:10.1016/j.websem.2022.100770.

[21] M. W. Chekol, J. Euzenat, P. Genevès, N. Layaïda, Sparql query containment under schema, Journal on Data Semantics 7 (2018) 133–154. URL: http://dx.doi.org/10.1007/s13740-018-0087-1. doi:10.1007/s13740-018-0087-1.

[22] 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), volume 26, 2023, pp. 285–297. URL: http://doi.org/10.48786/edbt.2023.23.

[23] W3C, Sparql queries to validate shape expressions (informative), 2013. URL: https://www.w3.org/2013/ShEx/toSPARQL.html.

[24] J.-E. L. Gayo, E. Prud'hommeaux, H. Solbrig, I. Boneva, Validating and describing linked data portals using shapes, 2017. arXiv:1701.08924.

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

[26] Y. Yang, M. Youill, M. Woicik, Y. Liu, X. Yu, M. Serafini, A. Aboulnaga, M. Stonebraker, Flexpushdowndb: Hybrid pushdown and caching in a cloud dbms, Proc. VLDB Endow. 14 (2021) 2101–2113.

[27] B. De Meester, P. Heyvaert, D. Arndt, A. Dimou, R. Verborgh, RDF graph validation using rule-based reasoning, Semantic Web Journal 12 (2021) 117–142. doi:10.3233/SW-200384.

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

[29] S. Staworko, I. Boneva, J.-E. L. Gayo, S. Hym, E. G. Prud'hommeaux, H. Solbrig, Complexity and Expressiveness of ShEx for RDF, in: 18th International Conference on Database Theory (ICDT 2015), volume 31 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2015, pp. 195–211. doi:10.4230/LIPIcs.ICDT.2015.195.

[30] I. Boneva, J.-E. L. Gayo, 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, Springer-Verlag, Berlin, Heidelberg, 2017, p. 104–120. doi:10.1007/978-3-319-68288-4_7.

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

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