# SHACL Shape Extraction for Evolving Knowledge Graphs using QSE

## Introduction

Storing complex and interconnected data is quite a challenge for data engineers. While relational systems are commonly used for structured data, tasks which involve a lot of relationships and lack a fixed schema may find a solution in graph databases. These serve as a foundation for constructing knowledge graphs, which have a broad range of applications, in industry and in academia and can also be used as a basis for machine learning. As in many applications, the data quality is vital and is the basis for good results later on. As a validation language SHACL can be used for RDF. Previously, a program called QSE (quality shape extraction) has been released, which extracts SHACL-shapes automatically from large datasets. The user can define two important parameters, which omit seldomly used shapes: support and confidence. Based on these generated node and property shapes, existing data can be validated. An extension to this program is called Shactor, a web-based tool which visualizes the extraction process and provides useful statistics. [1] [2]

Since knowledge graphs are not static, there exist different versions of a graph, maybe with minimal changes. QSE and Shactor are specialised on the shapes extraction, but not comparing them between different versions of a graph. This work aims at shapes extraction, specifically with evolving knowledge graphs.

## Goals

The main goal of this thesis is to develop a web-based tool which allows shapes extraction for different versions of a graph. This web-app will be heavily based on Shactor, since it already implements useful features, such as graph upload. Users can define different versions of a graph. For the first version, the user has the same options as in Shactor. He/She can choose between and exact or approximate QSE approach, define support and confidence and can specify classes and extract shapes. The user can create multiple shape extractions with different parameters. Later, when he/she uploads a new version of the graph, the user can compare the shapes with previously generated shapes. The web-app shows the user, which shapes stayed the same, which are new and which were deleted. Also, there will be information why shapes were added or deleted.

The second goal of this thesis is to adapt the QSE-algorithm, so that it is faster for another version of the graph. An approach is to calculate the difference between two versions of a graph and create shapes for the later version of the graph based on this difference. Another idea is to execute QSE for the first version and then use the shapes and interim results from the first run in the second run. The performance of these versions of QSE will be compared to find the best solution. This comparison will be made with the same parameters and will be based on different datasets, where the versions have different degrees of change.

## Current state of research

Presently, the QSE algorithm can be run using the command line, allowing the definition of multiple parameters but lacking user-friendliness. The extension, Shactor, addresses this issue by offering a user interface through a web-based tool. While this program allows users to extract shapes for a single version of a graph, it does not support the extraction of shapes from multiple versions Additionally, it only facilitates the exact approach and connection via a graph existing as a file.

There has been a lot of research on evolving knowledge graphs and defining versions of a graph. GraphDB already offers a feature for data versioning and history. [3] [4] [5] [6] [7]
However, in the context of my research project, adopting this form of versioning is currently not possible because not all graphs use the same form of versioning. Standardization across all graphs

would be essential, otherwise the application would only be applicable to specific graphs utilizing a particular versioning technique.

Regarding the second goal, there exist some solutions to differentiate RDF graphs. [8]

Also, there has been some general research on how data can be validated in knowledge graphs. [9] [10] [11]

Generally, there are other papers on knowledge graphs, which might be useful later. [12] [13] [14] [15]

## Research Questions

After defining the goals it becomes evident that there is a need for data engineers to ensure the quality of knowledge graphs across multiple versions. Since there is always a manual part in quality assurance, there is the need for a user-friendly web-based tool to evaluate the differences between extracted shapes of knowledge graphs. Considering the frequent changes in knowledge graphs, it would be a lot of work to run and compare extracted SHACL-shapes for every version. The state-of-the art solution would be to download the extracted shapes from Shactor and compare them with text-comparison tools.
Furthermore, the similarities between different versions of knowledge graphs raise the question, if knowledge can be reused between different runs and how this can happen. This is relevant for data engineers since it has the potential to significantly reduce execution time.

RQ1. How can extracted shapes from different versions of a graph be compared in a user-friendly way?

RQ2. Can the QSE-algorithm be more efficient by initially calculating the difference between two graphs instead of running the algorithm twice for both versions?

RQ3. Which (interim) results from previous versions can be reused to enhance the execution in following versions?

## Methods

Design science is selected as a scientific method. In the relevance cycle a simplified form of a systematic literature review will be used. During the design cycle, prototyping will be used for RQ1. For RQ1, a web-based tool will be built, the usability will be evaluated with qualitative expert interviews.
For RQ2 and RQ3, experimentation will be used to find methods to enhance the algorithm's speed. This will be quantitatively evaluated by measured the execution time.

## Literature

[1] K. Rabbani, M. Lissandrini, und K. Hose, „SHACTOR: Improving the Quality of Large-Scale Knowledge Graphs with Validating Shapes“, in *Companion of the 2023 International Conference on Management of Data*, in SIGMOD '23. New York, NY, USA: Association for Computing Machinery, 2023, S. 151–154. doi: 10.1145/3555041.3589723.
[2] K. Rabbani, M. Lissandrini, und K. Hose, „Extraction of Validating Shapes from Very Large Knowledge Graphs“, *Proc. VLDB Endow.*, Bd. 16, Nr. 5, S. 1023–1032, Jän. 2023, doi: 10.14778/3579075.3579078.
[3] M. Frommhold, R. N. Piris, und N. Arndt, „Towards Versioning of Arbitrary RDF Data“.
[4] A. Lohfink und D. McPhee, „Resource-Level Versioning in Administrative Geography RDF Data“.
[5] M. Tasnim, D. Collarana, D. Graux, F. Orlandi, und M.-E. Vidal, „Summarizing Entity Temporal Evolution in Knowledge Graphs“.

[6] R. Pernischová, D. Dell'Aglio, M. Horridge, und A. Bernstein, „Toward Predicting Impact of Changes in Evolving Knowledge Graphs".
[7] Mohamat Ulin Nuha, „Data Versioning for Graph Databases". [Online]. Verfügbar unter: https://repository.tudelft.nl/islandora/object/uuid:3cdbd161-3e6b-463a-957d-00ec0942917a/datastream/OBJ/download
[8] T. Berners-Lee und D. Connolly, „Delta: an Ontology for the Distribution of Di erences between RDF Graphs", Jän. 2004.
[9] „Tsaneva and Sabou - Hybrid Human-Machine Evaluation of Knowledge Graphs.pdf", Google Docs. Zugegriffen: 10. November 2023. [Online]. Verfügbar unter: https://drive.google.com/file/u/0/d/1jzH1KhSn6UxxhayoZV5rCXQF1nsCI5a6/view?pli=1&usp=embed_facebook
[10] A. Schmickl, „5 steps to detect inconsistencies in evolving knowledge graphs", Medium. Zugegriffen: 10. November 2023. [Online]. Verfügbar unter: https://alenaschmickl.medium.com/5-steps-to-find-inconsistencies-in-evolving-knowledge-graphs-6f3f88c0ab7b
[11] J. H. Brenas und A. Shaban-Nejad, „Proving the Correctness of Knowledge Graph Update: A Scenario From Surveillance of Adverse Childhood Experiences", *Front Big Data*, Bd. 4, S. 660101, Mai 2021, doi: 10.3389/fdata.2021.660101.
[12] S. Takan, „Knowledge graph augmentation: consistency, immutability, reliability, and context", *PeerJ Comput Sci*, Bd. 9, S. e1542, Aug. 2023, doi: 10.7717/peerj-cs.1542.
[13] L. Schmelzeisen, C. Dima, und S. Staab, „Wikidated 1.0: An Evolving Knowledge Graph Dataset of Wikidata's Revision History". arXiv, 9. Dezember 2021. Zugegriffen: 10. November 2023. [Online]. Verfügbar unter: http://arxiv.org/abs/2112.05003
[14] K. Belhajjame und M.-Y. Mejri, „Online maintenance of evolving knowledge graphs with RDFS-based saturation and why-provenance support", *Journal of Web Semantics*, Bd. 78, S. 100796, 2023, doi: https://doi.org/10.1016/j.websem.2023.100796.
[15] L. Meijer, „Bi-VAKs: Bi-Temporal Versioning Approach for Knowledge Graphs", 2022, Zugegriffen: 10. November 2023. [Online]. Verfügbar unter: https://repository.tudelft.nl/islandora/object/uuid%3A63aeab75-64a5-4b59-9cb0-241b603bd00d