Interactive Analysis of Knowledge Graph Validation Results with the SHACL Dashboard

Johannes Mäkelburg[1], Zenon Zacouris[1], Jin Ke[1] and Maribel Acosta[1]

[1]Technical University of Munich, Munich, Germany

Abstract

Validating knowledge graphs (KGs) ensures their quality and reliability in real-world applications. The Shapes Constraint Language (SHACL) has emerged as a recommended standard for validating RDF KGs, by defining structured constraints. Many organizations leverage SHACL validation and its reports to detect problems, guide corrections, and improve data quality. Yet, large-scale KGs often produce extensive validation reports, making manual analysis infeasible. To address this challenge, we present SHACL Dashboard, a novel online tool for visualization and multidimensional analysis of SHACL validation reports. It provides an interactive user interface featuring detailed violation summaries, analytical plots, and fine-grained insights into individual constraints. These functionalities enable users to efficiently understand validation results, identify problematic areas, and take precise corrective actions on their data. A demo version of the SHACL Dashboard is available online at https://purl.org/shacl-dashboard.

Keywords

SHACL, Visualization, Data Quality, Knowledge Graph, Constraints, Validation

1. Introduction

The Shapes Constraint Language (SHACL) [1], recommended by the W3C, is a language for expressing constraints on RDF knowledge graphs (KGs). By defining structural patterns and valid values, SHACL assesses data quality in RDF KGs. The outcome of the SHACL validation process is a validation report, which provides a detailed account of conformance and lists all detected violations [1]. In this way, the validation report offers a fine-grained view of all constraint violations, making it an essential tool for identifying and addressing erroneous data to improve overall data quality.

SHACL has gained significant attention in the industry, leading to its incorporation into commercial triple stores [2, 3] and its application in the validation of enterprise data (e.g., [4]). However, while validation reports are widely acknowledged for their utility, they face serious challenges when applied to large-scale KGs. For instance, the validation of a real-world RDF graph with over 30 million triples [5] yielded more than 10 million violations, which resulted in a violation report with over 84 million triples (even larger than the validated graph). Such massive reports are simply impossible to analyze manually and demand further processing techniques to extract meaningful and actionable insights.

In this paper, we present SHACL Dashboard: a novel online tool designed to tackle the aforementioned challenge by enabling interactive visualization of SHACL validation reports over large-scale RDF KGs. Our tool can process validation reports from any SHACL validator that follows the standard [1]. By incorporating innovative features such as automated prioritization of violations and an interactive user interface, SHACL Dashboard empowers users to focus on the most critical issues, dynamically explore validation reports, and efficiently analyze data quality issues. These highlights make the tool accessible to a broader range of users, guiding them to quickly understand validation results and take precise, timely corrective actions to improve data quality.

The remainder of this paper is structured as follows. In Section 2, we discuss related work. Section 3 details the implementation of the SHACL Dashboard, covering its architecture, layout, and key features.

ISWC 2025 Companion Volume, November 2–6, 2025, Nara, Japan

johannes.maekelburg@tum.de (J. Mäkelburg); zenon.zacouris@tum.de (Z. Zacouris); jin.ke@tum.de (J. Ke); maribel.acosta@tum.de (M. Acosta)

0009-0001-3821-7817 (J. Mäkelburg); 0009-0008-7806-2507 (Z. Zacouris); 0009-0001-8516-8894 (J. Ke); 0000-0002-1209-2868 (M. Acosta)

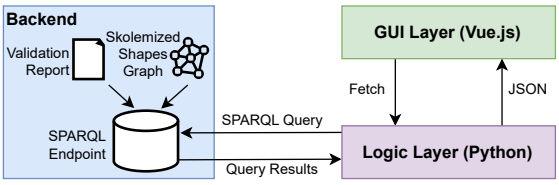

Figure 1: SHACL Dashboard 3-layered architecture.

```
SELECT ?nodeShape ?propertyShape COUNT(?violation)
WHERE {
  ?nodeShape a shacl:NodeShape .                    #tp1
  ?nodeShape shacl:property  ?propertyShape .       #tp2
  ? violation   shacl:sourceShape ?propertyShape . }  #tp3
```

Figure 2: SPARQL query joining constraint definitions (#tp1, #tp2) with violations (#tp3).

In Section 4, we present several use cases to demonstrate its capabilities. Finally, Section 5 concludes our work and outlines future work.

This demo complements our resource paper [6] by showcasing a live implementation of the SHACL Dashboard and demonstrating how users can analyze validation reports through concrete use cases.

2. Related Work

Since the introduction of SHACL, several tools have been proposed to manage the creation and manipulation of shapes and to manage the validation reports. Typically, they support one or more of the following functionalities: constraint visualization, shape visualization, validation report representation, violation prioritization, aggregated statistics and insights, and SPARQL-based exploration.

Constraint Tools: OntoPad [7] and UnSHACLed [8] support constraint visualization through graphical editors tailored to users unfamiliar with SHACL syntax.

Shapes Tools: SHACTOR [9] and RDFminer [10] focus on shape extraction and analysis, offering shape visualization along with basic support for violation prioritization and aggregated statistics. VocEditor [11] targets validation but only provides a validation report representation.

Validation Tools: SHACL Playground[1], SHACL4P [12], RDFShape [13], Zazuko SHACL Playground[2], and RDF Playground [14] primarily support validation report representation. SHACL4P additionally allows limited prioritization, while RDFShape adds shape visualization and violation prioritization. RDF Playground and SHACL Playground include SPARQL-based exploration.

To the best of our knowledge, the SHACL Dashboard is the first tool to support large-scale, query-based, and interactive analysis of SHACL validation reports while integrating prioritization logic, statistical summaries, and shape-level heatmaps in a unified tool.

3. SHACL Dashboard

We present the SHACL Dashboard, a tool to facilitate the analysis of SHACL validation reports generated from evaluating RDF datasets, referred to as the *data graph*. In SHACL, the constraints are defined in a separate RDF dataset called the *shapes graph*. Within this graph, a node shape represents a collection of constraints describing the expected structure and properties for a group of nodes in the data graph. Each node shape may include several *property shapes*, which define constraints on specific attributes of these nodes. These property shapes contain *constraint* components – conditions that data must satisfy, such as the minimum number of values for a property (sh:minCount) or the required datatype of an attribute (sh:datatype). The validation process applies these constraints to targeted entities in the data graph, referred to as *focus nodes*. The result of this process is a *validation report*, which summarizes the compliance of the data graph with the defined constraints.

Figure 1 illustrates the component architecture of the SHACL Dashboard, which is further detailed in the following subsections. A demo version of the SHACL Dashboard is available online at https://purl.org/shacl-dashboard. The source code for the SHACL Dashboard, including the frontend components,

[1]SHACL Playground: https://shacl.org/playground/
[2]Zazuko SHACL Playground: https://shacl-playground.zazuko.com/

SPARQL queries, data underlying the evaluation tables and figures, as well as documentation and instructions for extending the dashboard, is available on GitHub[3] and is licensed under the AGPL-3.0 license.

Backend In SHACL, both the shapes graph and the validation reports are modeled using RDF. To enable joining shapes with validation results, the shapes graph should be skolemized to replace blank nodes with unique identifiers, ensuring consistent references across both graphs. To facilitate large-scale analysis, these RDF graphs can be loaded into a triple store and queried via a SPARQL endpoint. Therefore, the backend of the SHACL Dashboard is built around a SPARQL endpoint. In our current implementation, we use OpenLink Virtuoso v7.2.13[4], leveraging its querying capabilities to efficiently handle large RDF graphs. The SPARQL endpoint provides the data required by the logical layer, by executing real-time queries to retrieve the necessary data to support different types of analyses such as violation distributions and constraint counts.

Logic Layer The logical layer of our SHACL Dashboard implements SPARQL queries to access the backend. An example of such queries is shown in Figure 2. We defined a total of 63 queries in this layer, extracting useful information via the endpoint to support frontend data analysis and key metrics for visualization. The query results are then serialized into JSON objects, which are passed to the GUI. This layer is implemented in Python v.3.9.12 and integrates with the frontend via Flask-based APIs, enabling seamless communication and service interaction between them.

Graphical User Interface (GUI) Layer The GUI, implemented using Vue.js 3.5.13, comprises several views, with which the user can analyze the validation reports from different perspectives. In this paper, we focus on the home view, the shapes view, and the shape insights, each offering dynamic exploration capabilities. Users can sort and filter tables, hover over charts to inspect detailed counts, and click on shapes or properties to access more detailed views. Navigation between views is fully dynamic and powered by preloaded SPARQL queries executed through the logic layer. For example, selecting a shape in the shapes view automatically updates the corresponding plots in the shape insights. These features enable users to progressively refine their analysis, from high-level summaries to the exact combinations responsible for most violations.

4. Demonstration

This section illustrates how the SHACL Dashboard can support users in analyzing validation reports through three concrete use cases over the DBpedia KG, which contains several types of quality issues [15]. We use a subset of constraints from Rabbani et al. [16], and the validation report computed by Ke et al. [17]. The following examples show how users can use the dashboard to detect high-impact problems, understand violation patterns, and pinpoint problematic constraint–shape pairs.

Prioritizing High-Impact Fixes One of the most valuable insights for the validation report is identifying which parts of the validated data and shapes are responsible for the majority of violations. However, this information is not directly available in the validation report and requires several queries that retrieve and aggregate details about the violations.

The SHACL Dashboard simplifies this process by displaying a concise summary of the report at the top of the *Home View* (Figure 3). The corresponding plots can be accessed by selecting the *Home View* from the top navigation bar, where summary statistics and violation distributions are presented. Plot ❶ displays key statistics such as the total number of violations, the percentage of violated shapes, paths, and constraint components, and the number of focus nodes (or entities) with violations, as well as their names. Whereas ❷ provides the corresponding distributions of each key statistic. Together, these two components give users an intuitive overview of where most errors in the data occur. In our DBpedia use case, our tool shows that violations are highly concentrated on a single path and constraint component. The most frequently violated path is rdf:type, and the most common violated constraint component is

[3]SHACL Dashboard Source Code: https://github.com/DE-TUM/shacl-dashboard
[4]Virtuoso Open-Source Edition: https://vos.openlinksw.com/owiki/wiki/VOS

Figure 3: Home View of the SHACL Dashboard. Top: Summary statistics including the number of focus nodes, shapes, paths, and violated constraints. Bottom: Visual distributions of violations across detail types.

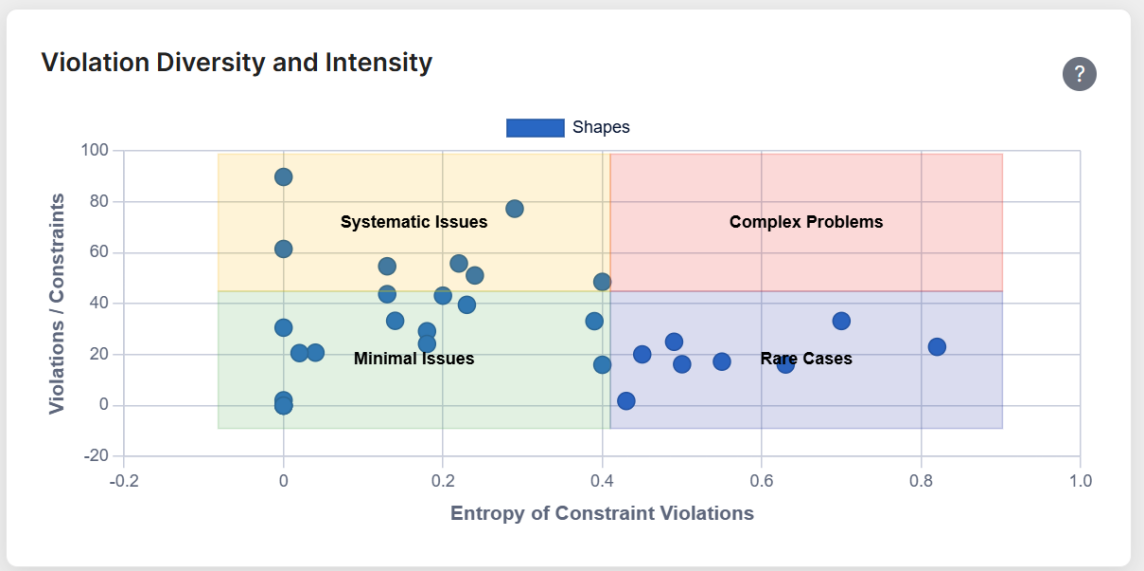

Figure 4: Violation Diversity and Intensity chart from the shapes view. Each shape is plotted according to the entropy of its constraint violations (diversity) and the average number of violations per constraint (intensity).

sh:InConstraintComponent. This combination indicates that the class assignments in the data graph are particularly problematic. This helps users quickly focus on the most problematic areas, enabling efficient resolution of large portions of violations with minimal effort.

Characterizing Violation Patterns After identifying potential high-impact fixes in the home view, a deeper analysis can be conducted in the shapes view to better understand the nature of these violations. The shapes view presents violations grouped by node shapes, allowing the user to systematically determine problematic shapes. SHACL allows the validation of various types of constraints, e.g., minCount, maxCount, datatype, class, etc. During analysis, an important question arises: Do the same constraints consistently lead to violations? The "Violation Diversity and Intensity" (Figure 4) addresses this question by comparing the entropy of constraint violations – i.e., whether violations are spread across multiple constraints – with the ratio of violations per constraint. To access this plot, the user needs to select the shapes icon in the sidebar and scroll to the Violation Diversity and Intensity section. This categorization allows users to distinguish between shapes that are easier to resolve, e.g., those with low entropy and low violations per constraint, and shapes that require more attention, e.g., those with high entropy or a high ratio of violations per constraint. In our DBpedia use case, our tool shows that

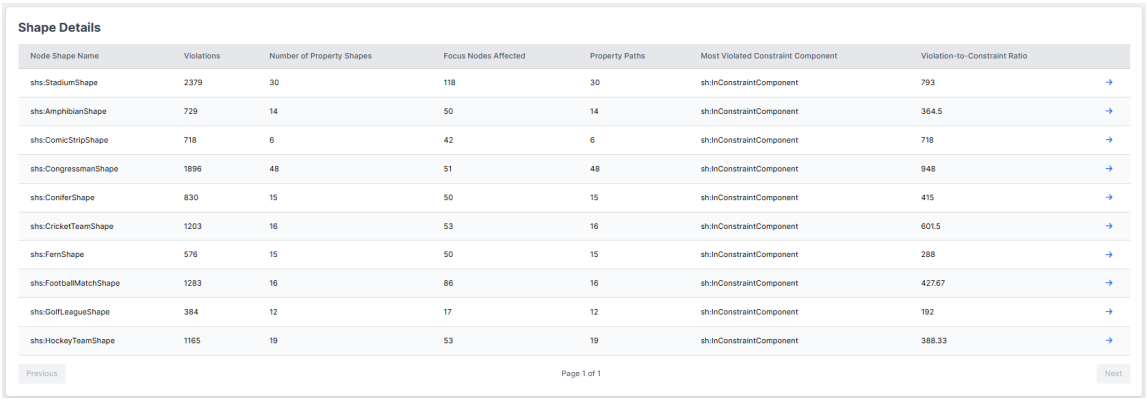

Figure 5: Table in the shapes view showing all property shapes defined under a selected node shape. Each row shows key statistics such as number of focus nodes, distinct violated constraints, and total violations.

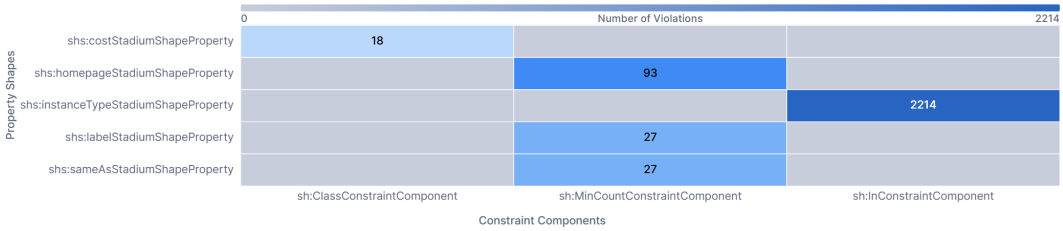

Figure 6: Heatmap in the shape insights. After selecting the StadiumShape shape in the shapes view, this visualization shows the number of violations per constraint–shape pair.

most shapes have violations with low entropy, e.g., the errors are concentrated on a small number of constraints. This means they fall into the categories of Minimal Issues or Systematic Issues, depending on the number of violations per constraint. The difference between the two categories is that the ratio of violations per constraint is high for Systematic Issues, e.g., many repeated violations of the same constraints, and low for Minimal Issues, e.g., only a few isolated violations. In our example, most of the shapes are located in the category Minimal Issues. This allows users to distinguish between isolated data issues and broader modeling problems, guiding targeted corrective actions.

Pinpointing Critical Constraint–Shape Pairs The most detailed analysis of the violations in one shape can be made within the shape insights page. Users can access this view by selecting a specific shape from the table at the bottom of the shapes view Figure 5. Figure 6 presents a heatmap illustrating the spread of violations across constraint types and shapes, providing an overview of how violations are distributed. In our DBpedia use case, our tool shows that the violations are spread across five different combinations. The most common combination (2214 violations) in the shs:StadiumShape corresponds to the combination of the constraint component sh:InConstraintComponent and the property shape shs:instanceTypeStadiumShapeProperty. In contrast, the remaining four combinations together account for only 165 violations, indicating a strong concentration of issues in a single constraint–shape pair. This helps users pinpoint the exact constraint–shape pairs responsible for most violations, streamlining debugging and refinement.

5. Conclusion

This work presents SHACL Dashboard, a novel online visualization tool designed for large-scale SHACL validation reports. Unlike the state of the art, our dashboard enables multidimensional analysis and extraction of valuable insights conveyed by the validation reports, presenting the results in a unified user interface. It offers three main views that provide a comprehensive analysis of SHACL validation reports. These views allow users to explore validation results at both high-level summaries and detailed insights, enabling efficient identification of problematic areas and a deeper understanding of constraint

violations. Regarding future work, we plan to enhance the SHACL Dashboard by introducing additional views for focus node and property path analysis while deploying it in an industrial setting with a previously implemented SHACL-based validation tool [4].

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1608 – 501798263. We thank TUM BPC Data Engineering students for their work that inspired this research.

Declaration on Generative AI

During the preparation of this work, the author(s) used X-GPT-4 in order to: Grammar and spelling check, as well as improve writing style. 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] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), 2017. URL: https://www.w3.org/TR/shacl/.

[2] Ontotext, Shacl-ing the data quality dragon: I – the problem and the tools, 2023. URL: https://www.ontotext.com/blog/shacl-ing-the-data-quality-dragon-i-the-problem-and-the-tools/, accessed: 2025-01-24.

[3] Stardog, Data quality constraints, 2023. URL: https://docs.stardog.com/data-quality-constraints, accessed: 2025-01-24.

[4] J. Mäkelburg, C. John, M. Acosta, Automation of electronic invoice validation using knowledge graph technologies, in: European Semantic Web Conference, Springer, 2024, pp. 253–269.

[5] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, Semantic web 6 (2015) 167–195.

[6] J. Mäkelburg, J. Ke, Z. Zacouris, M. Acosta, SHACL Dashboard: Analyzing Data Quality Reports over Large-Scale Knowledge Graphs, in: The Semantic Web – ISWC 2025: Proceedings of the 24th International Semantic Web Conference (Resources Track), Springer, 2025. To appear.

[7] N. Arndt, A. Valdestilhas, G. Publio, A. C. Arriaga, K. Höffner, T. Riechert, A visual SHACL shapes editor based on ontopad, in: SEMANTiCS (Posters & Demos), volume 2941 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021.

[8] S. Lieber, B. De Meester, P. Heyvaert, F. Brückmann, R. Wambacq, E. Mannens, R. Verborgh, A. Dimou, Visual notations for viewing rdf constraints with unshacled, Semantic Web 13 (2022) 757–792.

[9] K. Rabbani, M. Lissandrini, 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, SIGMOD '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 151–154. URL: https://doi.org/10.1145/3555041.3589723. doi:10.1145/3555041.3589723.

[10] R. Felin, P. Monnin, C. Faron, A. G. B. Tettamanzi, "RDFminer: An Interactive Tool for the Evolutionary Discovery of SHACL Shapes", in: The Semantic Web: ESWC 2024 Satellite Events, Springer Nature Switzerland, Cham, 2025, pp. 207–211.

[11] A. Valdestilhas, G. Publio, A. C. Arriaga, T. Riechert, Voceditor-an integrated environment to visually edit, validate and versioning rdf vocabularies, in: 2021 IEEE 15th International Conference on Semantic Computing (ICSC), IEEE, 2021, pp. 473–476.

[12] F. J. Ekaputra, X. Lin, Shacl4p: Shacl constraints validation within protégé ontology editor, in: 2016 International Conference on Data and Software Engineering (ICoDSE), IEEE, 2016, pp. 1–6.

[13] J. E. L. Gayo, D. Fernández-Álvarez, H. García-González, RDFShape: An RDF Playground Based on Shapes, in: International Workshop on the Semantic Web, 2018.
[14] B. Inostroza, R. Cid, A. Hogan, Rdf playground: An online tool for learning about the semantic web, in: Companion Proceedings of the ACM Web Conference 2023, 2023, pp. 111–114.
[15] M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, F. Flöck, J. Lehmann, Detecting linked data quality issues via crowdsourcing: A dbpedia study, Semantic web 9 (2018) 303–335.
[16] K. Rabbani, M. Lissandrini, K. Hose, Extraction of validating shapes from very large knowledge graphs, Proceedings of the VLDB Endowment 16 (2023) 1023–1032.
[17] J. Ke, Z. Zacouris, M. Acosta, Efficient validation of shacl shapes with reasoning, Proceedings of the VLDB Endowment 17 (2024) 3589–3601.