Is SHACL Suitable for Data Quality Assessment?

Carolina Cortés[1], Lisa Ehrlinger[1], Lorena Etcheverry[2] and Felix Naumann[1]

[1]Hasso Plattner Institute, Potsdam, Germany
[2]Universidad de la República, Montevideo, Uruguay

Abstract

Knowledge graphs have been widely adopted in both enterprises, such as the Google Knowledge Graph, and open platforms like Wikidata, to represent domain knowledge and support artificial intelligence applications. They model real-world information as nodes and edges. To embrace flexibility, knowledge graphs often lack enforced schemas (i.e., ontologies), leading to potential data quality issues, such as semantically overlapping nodes. Yet ensuring their quality is essential, as issues in the data can affect applications relying on them. To assess the quality of knowledge graphs, existing works propose either high-level frameworks comprising various data quality dimensions without concrete implementations, define tools that measure data quality with ad-hoc SPARQL queries, or promote the usage of constraint languages, such as the Shapes Constraint Language (SHACL), to assess and improve the quality of the graph. Although the latter approaches claim to address data quality assessment, none of them comprehensively tries to cover all data quality dimensions. In this paper, we explore this gap by investigating the extent to which SHACL core can be used to assess data quality in knowledge graphs. Specifically, we defined SHACL shapes for 69 data quality metrics proposed by Zaveri et al. [1] and implemented a prototype that automatically instantiates these shapes and computes the corresponding data quality measures from their validation results. All resources are provided for repeatability.

Keywords

Knowledge Graphs, Data Quality Assessment, RDF Validation, SHACL

1. Introduction

Knowledge graphs (KGs) have been increasingly used to represent domain knowledge and support artificial intelligence applications, such as information retrieval and question answering [2]. As a result, ensuring the quality of KGs is crucial for applications that rely on their input [3]. Major technology companies, such as Google (Google Knowledge Graph [4]) and Amazon (Alexa Knowledge Graph [5]), have developed proprietary KGs [2], while collaborative KGs, such as Wikidata [6], DBpedia [7] and YAGO [8], have emerged as open-source alternatives. This growing adoption of KGs is reflected in the growth of the Linked Open Data (LOD) cloud, which has enabled the publication of numerous datasets across domains, with 1 656 resources as of Nov. 2024 [9].

Knowledge graphs represent information about the world as nodes and edges [10]. They are typically constructed and enriched using diverse sources and (semi-)automated techniques, with some also supporting human curation [3, 10]. While several graph data models are available [11], this paper focuses on the Resource Description Framework (RDF) [12], which structures data as triples consisting of a subject, a predicate, and an object. RDF-based graphs can include a semantic schema, such as a vocabulary or ontology, that defines their expected structure. However, to ensure flexibility and support the evolution of KGs over time, these schemata are generally not enforced [13], which can introduce data quality issues. Furthermore, the quality of ontologies directly affects the quality of the data, as poorly defined classes or properties can lead to misclassified or ambiguous data in the graph, and inconsistent ontology axioms can propagate errors in reasoning over the graph [14]. Consider the case of Wikidata, which contains several classes that are difficult to distinguish, such as "geographical location", "location", and "geographic region", and also classes and instances appear mixed, such as "scientist", which is both a subclass of "researcher" and an instance of "profession" [2]. We focus on the quality assessment of the data graph, but the proposed methods can also be applied to ontologies.

WOP2025: 16th Workshop on Ontology Design and Patterns, November 3, 2025, Nara, Japan
carolina.cortes@hpi.de (C. Cortés); lisa.ehrlinger@hpi.de (L. Ehrlinger); lorenae@fing.edu.uy (L. Etcheverry); felix.naumann@hpi.de (F. Naumann)

Data quality (DQ) is generally defined as "fitness for use" [15], which highlights the importance of considering the context in which the data is utilized when assessing its quality. DQ is typically evaluated across various *dimensions*, such as Completeness, Consistency, and Understandability [16]. These dimensions can be quantified using specific metrics, known as *DQ metrics*, which are designed to measure different aspects of each dimension [16]. The process of data quality assessment (DQA) involves obtaining numerical values, referred to as *DQ measures*, that characterize various aspects of DQ. While many classifications exist for both DQ dimensions and metrics, there is no universally accepted standard [17]. As a result, DQA remains a challenging task in practice.

Several works address the quality of KGs from a *high-level* perspective, presenting definitions and metrics [1, 18] or conceptual frameworks to assess the quality of KGs [19, 20] without a concrete implementation. Other works approach the topic of KG quality from a *bottom-up* approach, by developing tools to assess DQ (e.g., [21, 22, 23, 24]). Most of these works define ad-hoc SPARQL (SPARQL Protocol and RDF Query Language) queries to obtain measures for DQ dimensions.

In recent years, constraint languages, such as the Shapes Constraint Language (SHACL) [25] and Shapes Expression Language (ShEx) [26], have emerged to enable RDF graph validation by specifying constraints as shapes [10]. Despite syntactic differences, both languages enable the definition of constraints on nodes and their value nodes (i.e., values reachable via properties or paths), allowing for the detection of data violations [27]. These languages enable a *low-level* perspective on DQ, focusing on concrete error detection through constraint checks. Unlike ad-hoc SPARQL-based approaches, constraint languages like SHACL offer a formal way to express validation rules.

Research gap and contributions. Since constraint languages for RDF were first introduced, different works have focused on the generation of shapes, either from the data or its metadata (e.g., ontologies) [28, 29, 30, 31, 32, 33, 34, 35, 36]. Only some works attempt to bridge the gap between the generation of shapes and their potential to assess and improve the quality of the data [28, 29, 35, 36]. In particular, Luthfi et al. [36] approaches DQ from the dimensions perspective, defining shapes for Completeness.

This paper explores how SHACL core can be leveraged for DQA. In particular, we want to investigate the extent to which we can connect the *high-level* view on DQ dimensions with the *low-level* view on DQ, which entails identifying constraint violations using SHACL shapes. Thus, the contributions of this paper are as follows:

1. Evaluation of the suitability of SHACL core for DQA by defining shapes for the set of 69 DQ metrics defined by Zaveri et al. [1].

2. A prototype that (i) automatically instantiates the defined SHACL shapes, and (ii) computes DQ metrics based on the shape validation results.

Outline. Section 2 summarizes related work. Section 3 presents the definition of SHACL shapes for a single dimension and metric of each group defined by [1]. The complete list of all defined SHACL shapes for all dimensions is provided in the appendix of the extended version of this paper [37]. Section 4 describes the implemented prototype for SHACL-based DQA, followed by a discussion of the suitability of SHACL for DQA in Section 5. Finally, Section 6 concludes the paper with an outlook on future work.

2. Related Work

Data quality in knowledge graphs. Several works address the quality of KGs from a high-level perspective. In particular, Zaveri et al. [1] identified a set of DQ dimensions and metrics through a systematic literature review and compared different DQA tools. Issa et al. [18] defined a set of DQ metrics for the Completeness dimension, and also analyzed different tools capable of assessing KG Completeness. Nayak et al. [19] analyzed different tools for assessment, profiling, and improvement of linked data and proposed a DQ refinement lifecycle. Chen et al. [20] proposed a DQA framework by mapping KG application requirements to DQ dimensions from [1], and extending them with two new dimensions: Robustness and Diversity. However, none of these works present a concrete implementation.

In addition, several tools have been proposed over the years to assess the quality of KGs and linked data (which we consider a type of KG, though not all KGs follow Linked Data principles). Some of these tools focus on specific DQ dimensions [38, 39], while others try to cover a wider range of them [21, 23, 40, 41, 24, 42, 43, 44]. Moreover, some tools focus on assessing the quality of SPARQL endpoints [42, 39, 38], while others focus on the quality of the data itself [21, 23, 40, 41, 24, 43, 44]. Assessment is primarily done via ad-hoc SPARQL queries [23, 40, 38, 24, 42, 22, 44], while some introduce more complex techniques [21, 43]. Only [24] considers the usage of a constraint language. Additionally, some of these tools haven't been maintained for some time or aren't available (See Appendix A of [37] for more details).

Constraint languages. Constraint languages validate a set of conditions over RDF graphs. The Shapes Constraint Language (SHACL), a W3C recommendation [25], enables this via shapes that specify constraints on the graph. The *shapes graph* holds these shapes, and the *data graph* is the RDF graph being validated. When a shape is evaluated on a node (the *focus node*), SHACL uses *node shapes*, to constrain the node itself, and *property shapes* to constrain *value nodes* reached from the *focus node* via a property or path in the graph. *Constraint components* define conditions to validate focus and value nodes. For example, *MinCountConstraintComponent* defines the *minCount* property, which can be used to specify a minimum number of values for a given property. The validation process takes a *data graph* and a *shapes graph* as input and produces a validation report, which provides insights on how to fix the errors causing the violations.

While constraint languages provide a formal way to define and validate constraints, writing them is time-consuming and requires domain expertise [27]. To address this, several works aim to automatically [28, 29, 30, 31, 32, 33, 34] or semi-automatically [35, 36] generate shapes from existing data [29, 30, 28, 34] or related artifacts [32, 33, 31], such as ontologies. Data-driven approaches usually cover basic constraints (e.g., required properties, ranges, cardinality) [33] but often produce many unreliable shapes, which are filtered using support/confidence [29] or trustworthiness scores [34]. Artifact-based methods, on the other hand, can generate richer constraints by leveraging formal restrictions like OWL axioms.

Few of these works try to bridge the gap between the generation of shapes and how these can help assess and improve DQ. Spahiu et al. [28] present an approach to generate SHACL shapes from semantic profiles created with a profiling tool, which are then used to assess the quality of different versions of a dataset over time. Rabbani et al. [29] present the tool SHACTOR, which not only generates shapes from a KG, but also allows the user to generate SPARQL queries that retrieve the triples that produce low support and confidence shapes. These triples are considered to be erroneous and can be removed from the graph to improve its quality. Luthfi et al. [36] consider the survey [18] and define shape patterns for different aspects of Completeness, testing them on Wikidata and DBpedia. To instantiate the shape patterns, they consider specific information provided by Wikidata, such as property constraints. Yang et al. [35] propose a SHACL-based DQ validation process for Completeness, Accuracy, and Consistency, using shapes tailored to a health ontology. While constraint examples are given, full shape definitions are not publicly accessible, and the method is specific to a particular ontology and dataset.

Although the approaches discussed above claim to address DQ in some way, none of them comprehensively attempts to cover all DQ dimensions.

3. SHACL shapes for DQ dimensions

This paper builds on the comprehensive DQ metrics survey by Zaveri et al. [1], which groups dimensions into four categories. We follow their structure, defining SHACL core shapes for each metric or explaining SHACL core's limitations when shapes cannot be defined.[1] We present the results of this study in tables 1–4, which illustrate the feasibility of implementing metrics from [1] with SHACL core. Symbols ✓,

[1] A discussion on SHACL extensions and the justification for the focus on SHACL core can be found in Section 5.3. Across the paper we use SHACL interchangeably with SHACL core, unless stated otherwise.

p, and x denote the degree to which metric implementation was possible (full, partial, or not at all, respectively). For the shape definition, we made some realistic assumptions (A1–3):

(A1) All entities are explicitly typed: for each entity e representing a real-world object, the triple ⟨e, rdf:type, c⟩ exists in the graph.

(A2) The ontology used is sufficiently defined: each class c and property p has triples ⟨c, rdf:type, rdfs:Class⟩ and ⟨p, rdf:type, rdf:Property⟩, respectively. Each property p, includes domain and range triples ⟨p, rdfs:domain, d⟩ and ⟨p, rdfs:range, r⟩. Other property characteristics (e.g., irreflexivity, asymmetry) are also defined. For every instance i of a class *c* defined in the ontology, ⟨i, rdf:type, owl:NamedIndividual⟩ exists.

(A3) Relevant domain knowledge, typically provided by domain experts, is available for shape instantiation, e.g., expected number of values for certain properties, gold standard value sets, or definitions of when data is considered to be up to date.

We acknowledge that these assumptions may not always hold in real-world KGs. If (A1) is violated, untyped instances are excluded from validation. Likewise, when (A2) is not satisfied, schema characteristics such as range, domain, or property constraints (e.g., symmetry, irreflexivity) cannot be checked. One possible direction to relax these requirements is to mine such characteristics automatically or leverage profiling statistics to approximate range and domain information, thereby supporting the construction of an "initial" ontology. Regarding (A3), this is inherent to DQA, as some quality dimensions rely on domain knowledge, limiting their assessment. Tables 1–4, mark the use of assumptions with $*_X$, where X is the assumption number. In the following subsections, we summarize SHACL core coverage per group, discuss a representative DQ metric for a single dimension of each group, and, if applicable, provide the corresponding shape in Turtle syntax. For the implemented shapes, we also indicate the DQ measure type: *binary measure* (1 if no violations, 0 otherwise), *ratio measure* (violation-based ratios), or *composite measure* (aggregated score across shape instances for specific properties or classes).

3.1. Accessibility

The *Accessibility* group includes dimensions related to accessing, retrieving, and verifying the authenticity of data: Availability, Licensing, Interlinking, Security, and Performance [1]. In the following, we exemplarily present the shape definition for a metric of the dimension Performance. Table 1 summarizes which metrics from the entire *Accessibility* group can be implemented using SHACL core. The remaining shapes' definitions can be found in Appendix B of [37].

Performance refers to how efficiently a system that hosts a large dataset can process data. For this dimension, Zaveri et al. [1] defined four metrics. We showcase the assessment of Performance with metric P1 in SH1. P1 was defined by [45] and checks for slash-URIs in datasets with over 500 000 triples. According to W3C recommendations, slash-URIs are preferred to identify entities in large graphs because hash ("#") URIs can cause performance issues [46]. We follow the W3C Best Practices [47] and apply this rule to entities' URIs. Here, SH1 targets subjects of triples with predicate `rdf:type`, applying a regex pattern that does not allow # to appear in URIs. Note that the regex pattern identifies any hash occurrence (not just at the end of the URI) to cover cases such as https://www.example.org#Example1/, which still loads all entities with https://www.example.org# as base namespace. Therefore, the validation result will output nodes whose URI contains a hash in any part of the URI, not just at the end.

Shape 1: Performance - Use of Hash URIs in Entities

```
ex:UsageHashURIsShape a sh:NodeShape ;
  sh:targetSubjectsOf rdf:type;
  sh:or (
    [ sh:path rdf:type; sh:hasValue rdfs:Class; ] [ sh:path rdf:type; sh:hasValue rdf:Property; ]
    [ sh:path rdf:type; sh:hasValue owl:NamedIndividual; ] [ sh:pattern "^[^#]*$"; ]
  ).
```

For this metric, assumption (A1) is required; without it, URIs of untyped entities cannot be checked. Assumption (A2) is also needed to restrict the constraint to entities, excluding classes, properties, and named individuals. Instances of `owl:NamedIndividual` are excluded to avoid mixing ontology-level individuals with graph entities, since validation runs over a graph containing both instance data and schema definitions. This "filtering" approach is used across the whole study when the metric verifies a constraint across all entities in the graph. The DQ measure derived from the validation result is a *ratio measure*, calculated with $\frac{\#\text{ violations}}{\#\text{ entities}}$.

Dimension	Metric Id	Metric	Implemented with SHACL core
Availability	A1	Accessibility of the SPARQL endpoint and the server	x
	A2	Accessibility of the RDF dumps	*p*
	A3	Dereferenceability of the URI	x
	A4	No misreported content types	x
	A5	Dereferenced forward-links	x
Licensing	L1	Machine-readable indication of a license in the VoID description	✓
	L2	Human-readable indication of a license in the documentation	x
	L3	Specifying the correct license	x
Interlinking	I1	Detection of good quality interlinks	x
	I2	Existence of links to external data providers	✓
	I3	Dereferenced back-links	x
Security	S1	Usage of digital signatures	✓
	S2	Authenticity of the dataset	✓
Performance	P1	Usage of slash-URIs	✓ *1*2
	P2	Low latency	x
	P3	High throughput	x
	P4	Scalability of a data source	x

Table 1
SHACL core coverage of the *Accessibility* group metrics.

3.2. Intrinsic

The *Intrinsic* category groups dimensions that are independent of the user's context, and assess whether data accurately (syntactically and semantically), compactly, and completely represents the real world, and whether it is logically consistent. The dimensions in this category are Syntactic Validity, Semantic Accuracy, Consistency, Conciseness, and Completeness [1]. In the following, we exemplarily present the shape definition for a metric of the dimension Consistency. Table 2 summarizes which metrics from the *Intrinsic* group can be implemented using SHACL core. The remaining shapes' definitions can be found in Appendix B of [37].

Consistency means that a knowledge base contains no (logical or formal) contradictions according to its knowledge representation and inference mechanisms [1]. For this dimension, Zaveri et al. [1] identified 10 metrics. We showcase the assessment of Consistency with metric CN5, which checks the correct use of inverse-functional properties. For this metric, Zaveri et al. [1] discuss two ways to check inverse-functional properties: (i) verifying the uniqueness of their values and (ii) defining a SPARQL constraint for such properties. While SHACL core cannot cover (ii), we defined SH2 to address (i), which ensures that no two different subjects share the same value, by verifying that each object has only one incoming link.

Shape 2: Consistency - Uniqueness of inverse functional properties

```
ex:InverseFunctionalPropertyShape a sh:NodeShape ;
  sh:targetObjectsOf PROPERTY_URI;
  sh:property [ sh:path [ sh:inversePath PROPERTY_URI ]; sh:maxCount 1; ].
```

SH2 shape is meant to be instantiated by replacing the placeholder `PROPERTY_URI` with specific inverse-functional properties defined in the ontology/vocabulary. Shape instantiation is needed in cases where metrics apply to particular classes or properties: instead of defining a separate shape for each one, we define a generic shape with a placeholder and instantiate it with the relevant URI(s) before validation. Moreover, in some cases, shapes may need to be instantiated with domain knowledge (e.g. SH3 in Section 3.3).

The validation report outputs a violation for each property value that is used more than once. Additionally, the DQ measure is a *composite measure*, so we compute an individual score for each property as: 1 if no violations are found, 0 otherwise. The final metric score is then aggregated with the formula: $\frac{\texttt{\# inverse-functional properties correctly used}}{\texttt{\# inverse-functional properties used to instantiate the shape}}$. We consider an inverse-functional property correctly used if the value of its individual score is 1.

Dimension	Metric Id	Metric	Implemented with SHACL core
Syntactic validity	SV1	No syntax errors of the documents	x
	SV2	Syntactically accurate values	p $*_3$
	SV3	No malformed datatype literals	✓
Semantic accuracy	SA1	No outliers	x
	SA2	No inaccurate values	p $*_3$
	SA3	No inaccurate annotations, labellings or classifications	p $*_3$
	SA4	No misuse of properties	x
	SA5	Detection of valid rules	x
Consistency	CN1	No use of entities as members of disjoint classes	✓ $*_2$
	CN2	No misplaced classes or properties	✓ $*_1*_2$
	CN3	No misuse of *owl:DatatypeProperty* or *owl:ObjectProperty*	✓ $*_2$
	CN4	Members of *owl:DeprecatedClass* or *owl:DeprecatedProperty* not used	✓ $*_1*_2$
	CN5	Valid usage of inverse-functional properties	p $*_2$
	CN6	Absence of ontology hijacking	x
	CN7	No negative dependencies/correlation among properties	p $*_3$
	CN8	No inconsistencies in spatial data	x
	CN9	Correct domain and range definition	✓ $*_2$
	CN10	No inconsistent values	p $*_2$
Conciseness	CS1	High intensional conciseness	x
	CS2	High extensional conciseness	p $*_3$
	CS3	Usage of unambiguous annotations/labels	x
Completeness	CP1	Schema completeness	p $*_1*_2$
	CP2	Property completeness	p $*_3$
	CP3	Population completeness	p $*_3$
	CP4	Interlinking completeness	p $*_1*_2$

Table 2
SHACL core coverage of the *Intrinsic* group metrics.

3.3. Contextual

Contextual dimensions are those that depend on the specific task at hand or on the context. This group includes four dimensions: Relevancy, Trustworthiness, Understandability, and Timeliness. We exemplarily present the shape definition for a metric of the dimension Timeliness. Table 3 summarizes which metrics from the entire *Contextual* group can be implemented using SHACL core. The remaining shapes' definitions can be found in Appendix B of [37].

Timeliness measures how current (or up-to-date) data is in relation to a specific task. For this dimension, Zaveri et al. [1]present two metrics. We illustrate the assessment of Timeliness with T1, which verifies the freshness of the dataset based on currency and volatility. This metric uses the formula $max\{0, 1 - \frac{\text{currency}}{\text{volatility}}\}$, where volatility refers to the length of time the data remains valid, and currency describes the age of the data at the time it is delivered to the user. In this case, we are not able to calculate the formula, but we can use SHACL core to identify outdated nodes. Therefore, for the definition of SH3, we assume there's some temporal annotation in the data, for example, using properties like `dcterms:date` or `dcterms:temporal` from the Dublin Core vocabulary. The defined shape identifies outdated entities by constraining the value of the temporal property to be after a certain point in time, indicating that the node is up-to-date. For this shape, we need to consider all assumptions, given that we are targeting entities. Additionally, we require a vocabulary or ontology to determine which properties are used to annotate entities with temporal facts, and domain knowledge indicating when entities are considered up-to-date. The validation report for this shape outputs a violation for each of the entities whose `dcterms:date` value is older than DATE_RANGE_MIN_BOUND. The DQ measure in this case is a *ratio measure*, calculated as $\frac{\texttt{\#violations}}{\texttt{\#entities}}$.

Shape 3: Timeliness - Outdated entities

```
ex:TimelinessEntitiesShape a sh:NodeShape ;
  sh:targetSubjectsOf rdf:type;
  sh:or ( [ sh:path rdf:type; sh:hasValue rdfs:Class; ]
          [ sh:path rdf:type; sh:hasValue rdf:Property; ]
          [ sh:path rdf:type; sh:hasValue owl:NamedIndividual; ]
          [ sh:path dcterms:date; sh:minInclusive "DATE_RANGE_MIN_BOUND"; ]  ).
```

Dimension	Metric Id	Metric	Implemented with SHACL core
Relevancy	R1	Relevant terms within meta-information attributes	x
	R2	Coverage	p $*_3$
Understandability	U1	Human-readable labelling of classes, properties and entities as well as presence of metadata	p $*_1*_2$
	U2	Indication of one or more exemplary URIs	✓
	U3	Indication of a regular expression that matches the URIs of a dataset	✓ $*_1*_2$
	U4	Indication of an exemplary SPARQL query	x
	U5	Indication of the vocabularies used in the dataset	✓
	U6	Provision of message boards and mailing lists	x
Trustworthiness	TW1	Trustworthiness of statements	x
	TW2	Trustworthiness through reasoning	p $*_1*_2$
	TW3	Trustworthiness of statements, datasets and rules	p $*_1*_2$
	TW4	Trustworthiness of a resource	x
	TW5	Trustworthiness of the information provider	p $*_2*_3$
	TW6	Trustworthiness of information provided (content trust)	✓ $*_1$ $*_2$ $*_3$
	TW7	Reputation of the dataset	x
Timeliness	T1	Freshness of datasets based on currency and volatility	p $*_1$ $*_2$ $*_3$
	T2	Freshness of datasets based on their data source	p $*_3$

Table 3
SHACL core coverage of the *Contextual* group metrics.

3.4. Representational

The *Representational* group addresses design aspects of the data. The dimensions in this category are Representational Conciseness, Interoperability, Versatility, and Interpretability [1]. We exemplarily present the shape definition for a metric of the dimension Versatility. Table 4 summarizes which metrics from the entire *Representational* group can be implemented using SHACL core. The remaining shapes' definitions can be found in Appendix B of [37].

Versatility refers to the availability of data in multiple representations and its support for internationalization [1]. For this dimension, Zaveri et al. [1] present two metrics. We showcase the assessment of Versatility with V2, defined in [45], which checks whether data is available in multiple languages by verifying the use of language tags in literals used for entity labels and descriptions. This metric can be partially covered by SHACL core, as it allows us to check for the presence of language tags on labels and descriptions. However, it does not verify whether the literal values are actually written in the specified language, which would require semantic analysis beyond SHACL's capabilities. Therefore, for this metric, we defined shapes SH4 and SH5, which check that labels and descriptions in entities have language tags. For both shapes, we consider assumptions (A1) and (A2), as we verify the use of language tags in entity labels and descriptions. Moreover, constraints are only checked for entities that have a label or description.

Shape 4: Versatility - Languages in entities labels

```
ex:DifferentLanguagesLabelsShape a sh:NodeShape ;
  sh:targetSubjectsOf rdfs:label;
  sh:or (
    [sh:path rdf:type; sh:hasValue rdfs:Class;]
    [sh:path rdf:type; sh:hasValue rdf:Property;]
    [sh:path rdf:type; sh:hasValue owl:
         NamedIndividual;]
    [sh:path rdfs:label; sh:datatype rdf:langString;]
  ).
```

Shape 5: Versatility - Languages in entities descriptions

```
ex:DifferentLanguagesDescriptionsShape a sh:NodeShape;
  sh:targetSubjectsOf rdfs:comment;
  sh:or (
    [sh:path rdf:type; sh:hasValue rdfs:Class;]
    [sh:path rdf:type; sh:hasValue rdf:Property;]
    [sh:path rdf:type; sh:hasValue owl:NamedIndividual
         ;]
    [sh:path rdfs:comment; sh:datatype rdf:langString
         ;]
  ).
```

The validation report for SH4 and SH5 outputs a violation for each entity that has a label or description without a language tag, respectively. In both cases, the DQ measure is a *ratio measure*, calculated using the formulas $\frac{\texttt{\# violations}}{\texttt{\# entities with labels}}$ and $\frac{\texttt{\# violations}}{\texttt{\# entities with descriptions}}$, respectively.

Dimension	Metric ID	Metric	Implemented with SHACL core
Representational conciseness	RC1	Keeping URIs short	✓ $*_1*_2$
	RC2	No use of prolix RDF features	✓ $*_1*_2$
Interoperability	ITO1	Re-use of existing terms	✓ $*_3$
	ITO2	Re-use of existing vocabularies	p $*_3$
Versatility	V1	Provision of the data in different serialization formats	✓
	V2	Checking whether data is available in different languages	p $*_1*_2$
Interpretability	ITP1	Use of self-descriptive formats	✓ $*_1*_2$
	ITP2	Detecting the interpretability of data	p $*_3$
	ITP3	Invalid usage of undefined classes and properties	✓ $*_2$
	ITP4	No misinterpretation of missing values	✓ $*_1*_2$

Table 4
SHACL core coverage of the *Representational* group metrics.

4. Prototype

This section presents our prototype for DQA using SHACL core. It is built with the Python libraries *rdflib* and *PySHACL* and provides a Streamlit dashboard to visualize results. The code is available on GitHub at [48].

Overview of the DQ assessment process. Our DQA process takes as input the data graph to be evaluated, optionally a metadata file (either the VoID or DCAT description of the graph), and an ontology, along with a set of vocabularies used in the data graph. Without an ontology, vocabularies, or a metadata file, fewer shapes are instantiated, and some, such as those targeting `void:Dataset` or those checking for class/property labels, are not validated, as they rely on these artifacts. A configuration file allows users to customize preferred properties needed for instantiating SHACL shapes.

Figure 1 shows the architecture of the prototype. The DQA process begins by Step (1) profiling the graph, ontology, and vocabularies to obtain the necessary information for shape instantiation and metric calculation. For example, SH2 is instantiated with properties marked as `owl:InverseFunctionalProperty` in the ontology; while to compute the measure associated with SH1, we retrieve the total number of entities.

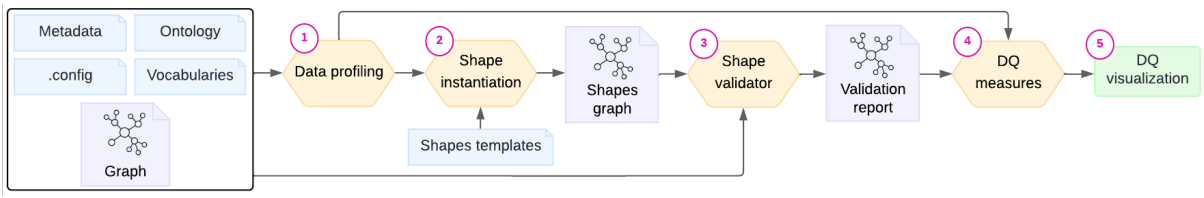

Figure 1: SHACL-based data quality assessment architecture.

After obtaining this information, we (2) instantiate the shapes' templates and generate the shapes graph. Our prototype stores SHACL shapes as reusable templates with variables that are replaced with actual values during instantiation – a process where the template is populated with specific properties from either the graph profiling results or the configuration file of the dataset. For example, shape SH1 uses the type property specified in the configuration file to create a concrete shape from its template form. Moreover, shape SH2 is instantiated with inverse-functional properties used in the dataset, obtained from the graph profiling results. Before shape validation, we perform a pre-processing step, based on assumption (A2), to enrich the data graph with necessary triples (described in detail in Section 4 of [37]). Then, we use the validator (3) provided by the PySHACL library to validate the shapes against the data graph, ontology, vocabularies, or metadata file, depending on the shape. Once the validation is completed, we (4) calculate the DQ measures from the validation result, which are later stored in a CSV file. Finally, the results can be visualized in a dashboard (5).

Shape instantiation details. For the 69 metrics defined in Zaveri et al. [1], we defined 64 *shapes*. The mapping is not one-to-one: some metrics have no shape, others (e.g., V2 in Section 3.4) have multiple. We identified 38 shapes that could be instantiated without domain expert input, and excluded five more: one requiring merged graphs (SH11) and 4 using non-standard vocabularies (S13, S14, S53, S55) (see Appendix B of [37]). Of the 38, 11 rely on vocabulary or ontology terms, and are instantiated only with those found in the data graph (e.g., SH2 uses only inverse-functional properties present in the data). This avoids generating unnecessary shapes, as many data graphs may partially reuse vocabularies. The exceptions are the shapes for metrics CN2 and CP1: CN2 checks for property/class misuse by instantiating with all available classes and properties since misused ones do not appear in profiling results, while CP1 uses all defined classes in the vocabulary to check if they are used. See Appendix C of [37] for additional aspects of shape instantiation aimed at improving runtime efficiency.

Evaluation. We evaluated the prototype with three datasets from the LOD Cloud [49]: *Temples of the Classical World* (15 326 triples and 1 363 entities), *DBTunes - John Peel Sessions* (271 369 triples and 76 056 entities), and *Drugbank* (3 646 181 triples and 316 555 entities). Validation time ranged from 40 seconds to 3 hours for 500–740 instantiated shapes, depending on the dataset. Longer times are likely due to both more triples and more shapes. As the prototype aimed to test the SHACL-based DQA approach, performance aspects were left outside its scope, and no runtime experiments were conducted since validation relied on an external library. Section 4 of [37] presents detailed results for the *Temples of the Classical World* dataset, while results for other datasets are available on GitHub at [48]

5. Suitability of SHACL core to assess RDF data quality

We now discuss the suitability of SHACL core to assess RDF DQ. Section 5.1 highlights where SHACL core falls short in covering certain DQ dimensions – metrics not mentioned are considered covered (see Section 3 and Appendix B of [37]). We then examine SHACL core's strengths and limitations (Section 5.2), and discuss SHACL core's extensions (Section 5.3).

5.1. Coverage of DQ dimensions by SHACL core

For each of the 18 DQ dimensions discussed in Zaveri et al. [1], we present SHACL core coverage as the normalized sum of coverage values assigned to each DQ metric: 1 for full coverage, 0.5 for partial coverage, and 0 for no coverage. The resulting sum is divided by the total number of metrics to obtain an overall coverage percentage. Figure 2 shows these results, where two dimensions with 100% coverage stand out: Representational Conciseness and Security. In these cases, the implemented metrics check for the presence or use of specific properties or classes.

Seven dimensions present a coverage between 50-90% : Timeliness, Understandability, Consistency, Completeness, Syntactic Validity, Interoperability, Interpretability, and Versatility. Regarding Timeliness, SHACL core partially covers both of the proposed metrics. While it cannot directly compute the formulas these metrics rely on, like T1's formula or T2's time-distance calculation, it can still check

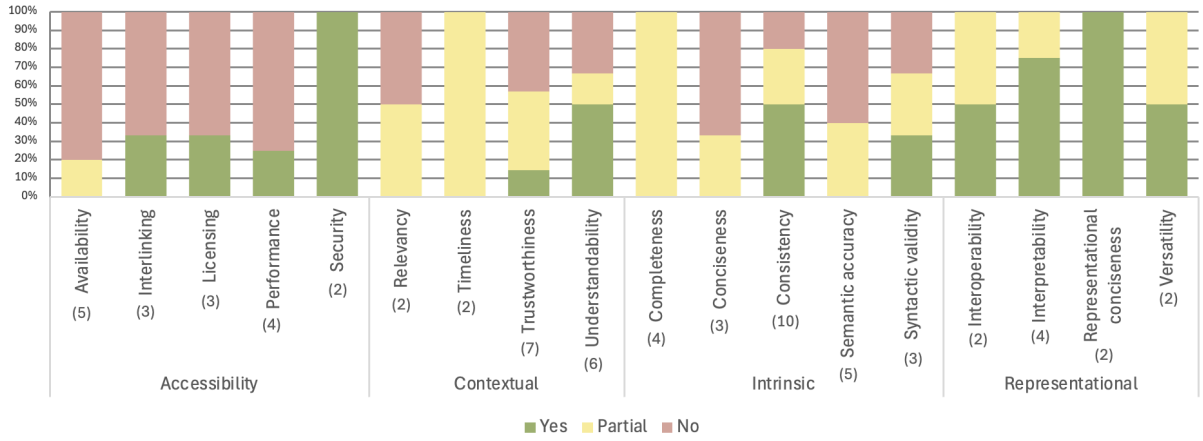

Figure 2: Coverage of DQ dimensions by SHACL core (including number of metrics for each dimension).

related aspects. For example, for T1, we defined a shape that detects outdated entities via temporal annotations, and for T2, the shape verifies if the dataset is up to date, even though we cannot compute distances with SHACL core.

In the context of Understandability, Versatility, and Interpretability, SHACL only partially covers metrics U1, V2, and ITP2 due to its limitations to do "semantic" checks. It can confirm the presence of labels (U1) and language tags (V2, ITP2), but it cannot assess the readability of labels or whether literals align with their specified language. Additionally, for Understandability, SHACL core cannot cover metrics U4 and U6. U4 is not covered because there's no standard method to declare example SPARQL queries. While datasets might use properties like `rdfs:comment` to include such queries, this is not standard and would require language processing to verify. U6 involves checking external resources (e.g., mailing lists), which SHACL core cannot handle since it only works on RDF graphs.

In terms of Consistency, several metrics rely on SPARQL queries or reasoning, which are beyond the capabilities of SHACL core (specifically metrics CN5, CN6, and CN10). Additionally, SHACL core does not cover CN8, as it requires specialized knowledge related to the representation of spatial data.

In the case of Completeness, all metrics are partially covered. CP1 measures schema completeness, but SHACL core can only check class usage, not property usage, since SHACL core cannot target triples in the graph to check whether a property is used or not. CP2 defines two aspects for measuring property completeness, where the second one uses property distribution statistics to assess completeness, which SHACL core does not support. CP3 measures population completeness; while SHACL core can check for cardinality and allowed values, it cannot leverage semantic constructs explicitly stated in the graph (e.g., recognizing equivalent entities defined using `owl:sameAs`). Finally, CP4 (interlinking completeness) is partially covered: SHACL core cannot verify linkset completeness because it involves checking the existence of links between instances of equivalent classes. However, SHACL node shapes require a specific target, while this check verifies the existence of certain triples in the graph and is not associated with any particular node.

When it comes to Syntactic Validity, SHACL core cannot cover SV1 since it requires checking syntax errors of RDF documents, while SHACL works after parsing the RDF document. SV2 is only partially covered because some aspects of this metric involve complex techniques like clustering.

For Interoperability, SHACL core does not cover all metrics due to its limited target capabilities. In particular, ITO2 requires checking vocabulary usage. While SHACL can check the usage of classes, it cannot check property usage, as this would entail checking if a property is used in any triple, but SHACL cannot target the predicate position in triples.

Finally, we turn to dimensions that are covered below 50% (Availability, Interlinking, Licensing Performance, Relevancy, Trustworthiness, Conciseness, and Semantic accuracy).

For Availability, most of the metrics cannot be covered with SHACL core, as most require accessing resources on the web, such as checking the dereferenceability of URIs or detecting broken links.

In the case of Interlinking, SHACL core cannot cover I1, as it requires computing graph-based measures (e.g., interlinking degree), which cannot be expressed as constraints over nodes and their properties. I3 also cannot be covered: SHACL cannot dereference URIs (task 1), nor verify whether there is any triple with the resource as the object (in-links) (task 2). For task 2, SHACL core also falls short because, while it can check for the existence of values for specific properties using `sh:path`, it cannot verify the existence of any triple where the resource is the object, regardless of the predicate.

As for Licensing, the metric L2 cannot be covered with SHACL core since it entails verifying the existence of a human-readable license in the documentation of the dataset, which usually is an HTML document. For metric L3, it also cannot be covered with SHACL core, as it requires checking license clauses and comparing licenses between datasets, which involves natural language processing.

Regarding Performance, SHACL core cannot cover metrics (P2 - P4), as they describe system-level behaviors (e.g., low-latency or high throughput), rather than constraints on nodes and properties.

For Relevancy, R1 is not covered, as it requires identifying relevant data via ranking or crowd-sourcing, which SHACL, as a constraint language, does not support. R2 is only partially covered, since it entails measuring the coverage (i.e., number of entities) and level of detail of entities (i.e., number of properties) in the dataset, to ensure that the data is appropriate for the task at hand. While SHACL core does not provide a way of counting entities and properties, we were able to define a shape that states the expected properties (level of detail) for instances of a certain class. In the validation results, we obtain entities lacking the expected level of detail.

Regarding Trustworthiness, SHACL core fully covers one metric, partially covers three, and cannot cover the remaining three. Metrics TW2 and TW3 require annotating the data with trust values, for example, using a trust ontology. While SHACL core cannot annotate the data, it can check the presence of these annotations, so these are partially covered. TW5 can only be partially covered since one of the aspects of this metric states checking the trustworthiness of the information provider using decision networks, which SHACL core cannot handle. To conclude, the metrics that cannot be covered for this dimension require trust value computations (TW1, TW4, and TW7) or human input (TW7), both beyond SHACL's capabilities.

For Conciseness, SHACL core is not able to cover the metric CS1, as it requires identifying semantically equivalent properties or classes. CS2 is only partially covered, as one of its approaches requires identifying duplicated entities (i.e., entities with different URIs but similar or identical property values), which involves cross-entity comparisons and similarity measures, both unsupported by SHACL core. Finally, CS3 cannot be covered by SHACL core, as detecting ambiguous labels and annotations requires semantic interpretation beyond SHACL's capabilities.

Finally, for Semantic Accuracy, three metrics (SA1, SA4, and SA5) are not covered, since they rely on outlier detection (SA1), profiling (SA4), and association rule generation via induction and analogy methods (SA5) - all of which are unsupported by SHACL core. The other two (SA2 and SA3) are only partially covered. SA2 specifies three checks to detect inaccurate values; the third (i.e., validating functional dependencies) is not supported by SHACL core, as it requires comparing property values for triples that do not share the same subject. Metric SA3 checks for inaccurate labels and classifications. We defined a SHACL shape to verify labels and types against a list of allowed values, but we cannot compute the original metric defined in Zaveri et al. [1] since it requires similarity measures, and SHACL core supports only exact value matching.

5.2. Strengths and Limitations of SHACL core

SHACL core has many strengths for specific aspects of DQA. For example, it can perform syntactic validation, including pattern validation, datatype checks, and verifying whether values fall within specified ranges or lists. It can validate consistency between the data and vocabulary or ontology definitions, such as ensuring correct use of a property's domain or range. It can also support best practices for publishing Linked Data, such as the use of hash URIs for entities, labels in resources (i.e., entities, classes, and properties), and some characteristics of URI designs (i.e., short URIs and no parameters). Moreover, SHACL can verify the correct application of property characteristics such

as irreflexive, functional, asymmetric, and inverse-functional. Finally, it is well-suited for defining the expected structure of class instances, specifying required properties, their types, and other basic constraints [13].

However, its applicability to DQA is still limited in several ways:

Lack of External Access. SHACL core operates solely within the RDF graph and cannot access external web resources or perform system-level checks. This restricts its applicability to metrics related to Availability and Performance, which often require testing endpoints, dereferencing URIs, or measuring response times.

Node-Centric Scope. Constraints in SHACL core are evaluated for individual *focus nodes* and their immediate property values. This node-centric approach limits the ability to compute network-level measures, which are essential for evaluating the Interlinking dimension (covered by SHACL-SPARQL).

No Cross-Entity Comparison. SHACL core lacks mechanisms to compare property values across different entities in the graph. As a result, it cannot assess functional dependencies or detect duplicated entities, limiting its applicability for Semantic Accuracy and Conciseness (covered by SHACL-SPARQL).

No Arithmetic or Dynamic Expressions. SHACL core lacks support for arithmetic operations and dynamic expressions. For instance, it cannot evaluate conditions like `age = now() - birthDate`. This limits the applicability of SHACL core for metrics that depend on computed values, such as Timeliness and Trustworthiness (covered by SHACL-SPARQL).

Limited Semantic Awareness. SHACL core lacks mechanisms to verify semantic aspects of the data, which are crucial for dimensions like Semantic Accuracy, Understandability, and Versatility. Additionally, it cannot take advantage of semantic declarations present in the graph, such as entity alignments defined through properties like `owl:sameAs`.

Restricted Targeting Mechanism. SHACL provides only a small set of predefined target types (e.g., `sh:targetClass`, `sh:targetNode`). This makes it difficult to define metrics that need to assess patterns over arbitrary triples or general predicate-based constraints (covered by SHACL-SPARQL).

5.3. SHACL extensions

The SHACL language consists of two main components: SHACL core and SHACL-SPARQL. Additionally, there exist non-standard SHACL extensions such as the *DASH Data Shapes Vocabulary* [50] and *SHACL Advanced Features* [51], all of which rely heavily on SPARQL. This paper focuses on SHACL core for several reasons. First, studies that extract constraints from RDF graphs primarily use SHACL core components. Notably, research by Spahiu et al. [28] and Rabbani et al. [29] indicate that SHACL can enhance DQ. Thus, we aim to determine if SHACL core alone is sufficient for this purpose. Second, while SHACL can be implemented in various programming languages (e.g., *jena-shacl* in Java and *PySHACL* in Python), each with their own optimizations, once SPARQL-based constraints are used, performance depends on the underlying query engine. Furthermore, while the SHACL recommendation details its extension via SPARQL, other languages could be used. Lastly, we choose to focus on SHACL core because it is typically easier to understand and use than writing complete SPARQL queries, allowing for more intuitive formulation of constraints accessible to domain experts [52]. Appendix D of [37] provides more details on SHACL extensions.

6. Conclusion and Future Work

In this paper, we assess the suitability of SHACL for DQA by defining shapes for the 69 data quality (DQ) metrics identified by Zaveri et al. [1], whenever possible. We also developed a prototype to automatically instantiate and validate the defined shapes, and compute DQ measures from the validation results.

Our findings indicate that SHACL is well-suited for syntactic validation (pattern validation, datatype checks, and value range enforcement), structural validation of class instances, ensuring correct use of properties and classes, and enforcing linked data best practices. This makes SHACL particularly useful for assessing dimensions such as Syntactic Validity, Interpretability, Security, Representational Conciseness, and specific aspects of Consistency, Versatility, and Understandability. However, SHACL core has

several limitations: it cannot access external resources, perform cross-entity comparisons, support network-based measures, or assess data semantics. This limits its use for Availability, Conciseness, Interlinking, and Semantic Accuracy. Its lack of dynamic calculations and limited targeting also limits its applicability for Interoperability, Trustworthiness, and Timeliness.

The DQ metrics defined by Zaveri et al. [1] may benefit from refinement, as some are overly specific (e.g., those that require spatial data representations) and difficult to generalize, limiting their practical use in KG DQA. Moreover, new metrics (e.g., bias [53]) are not covered in the survey; we suggest extending this work to test whether SHACL core can cover them.

Assessing DQ goes beyond checking constraints, encompassing the data source, the system processing the data, downstream tasks, and human factors [54]. SHACL core focuses on data graphs, limiting its applicability for comprehensive DQA. Future work should explore hybrid approaches combining SHACL's constraint checking with tools that access external resources and LLMs to overcome its limited semantic awareness. It would also be useful to identify which limitations are inherent to SHACL core design that can be addressed through extensions.

Declaration on Generative AI

During the preparation of this work, Grammarly was used for spell checking.

References

[1] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer, Quality assessment for Linked Data: A Survey, Semantic Web 7 (2012) 63–93. doi:10.3233/SW-150175.

[2] F. M. Suchanek, M. Alam, T. Bonald, L. Chen, P. H. Paris, J. Soria, Yago 4.5: A large and clean knowledge base with a rich taxonomy, Proceedings of the International Conference on Information retrieval (SIGIR) (2024) 131–140. doi:10.1145/3626772.3657876.

[3] Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO, Semantic Web 9 (2018) 77–129. doi:10.3233/SW-170275.

[4] A. Singhal, Introducing the knowledge graph: things, not strings, https://blog.google/products/search/introducing-knowledge-graph-things-not/, 2012. Accessed: 2025-07-29.

[5] A. Developer, Alexa entities reference, https://developer.amazon.com/en-US/docs/alexa/custom-skills/alexa-entities-reference.html, 2023. Accessed: 2025-07-29.

[6] D. Vrandečić, M. Krötzsch, Wikidata: A free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.

[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: The Semantic Web, Springer, 2007, pp. 722–735.

[8] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: A core of semantic knowledge, in: Proceedings of the 16th international conference on World Wide Web, ACM, 2007, pp. 697–706.

[9] G. Tuozzo, Navigating the lod subclouds: Assessing linked open data quality by domain, in: Companion Proceedings of the ACM on Web Conference, ACM, 2025, pp. 2141–2148. doi:10.1145/3701716.3717569.

[10] A. Hogan, C. Gutierrez, M. Cochez, G. de Melo, S. Kirrane, A. Polleres, R. Navigli, A.-C. N. Ngomo, S. M. Rashid, L. Schmelzeisen, S. Staab, E. Blomqvist, C. d'Amato, J. E. L. Gayo, S. Neumaier, A. Rula, J. Sequeda, A. Zimmermann, Knowledge Graphs (2022). doi:10.1007/978-3-031-01918-0.

[11] R. Angles, The Property Graph Database Model, in: Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, 2018.

[12] D. Wood, M. Lanthaler, R. Cyganiak, RDF 1.1 Concepts and Abstract Syntax, 2014.

[13] A. Polleres, R. Pernisch, A. Bonifati, D. Dell'Aglio, D. Dobriy, S. Dumbrava, L. Etcheverry, N. Ferranti, K. Hose, E. Jiménez-Ruiz, et al., How does knowledge evolve in open knowledge graphs?, Transactions on Graph Data and Knowledge 1 (2023) 11–1.

[14] S. M. Gurk, C. Abela, J. Debattista, Towards ontology quality assessment, in: Joint proceedings of the Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW) and the Workshop on Linked Data Quality (LDQ) co-located with European Semantic Web Conference (ESWC), volume 1824 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017, pp. 94–106.
[15] R. Y. Wang, D. M. Strong, Beyond Accuracy: What Data Quality Means to Data Consumers, Journal of Management Information Systems 12 (1996) 5–33.
[16] L. L. Pipino, Y. W. Lee, R. Y. Wang, Data quality assessment, Communications of the ACM 45 (2002) 211–218. doi:10.1145/505248.506010.
[17] C. Cichy, S. Rass, An Overview of Data Quality Frameworks, IEEE Access 7 (2019) 24634–24648.
[18] S. Issa, O. Adekunle, F. Hamdi, S. S. Cherfi, M. Dumontier, A. Zaveri, Knowledge Graph Completeness: A Systematic Literature Review, IEEE Access 9 (2021) 31322–31339. doi:10.1109/ACCESS.2021.3056622.
[19] A. Nayak, B. Bozic, L. Longo, Linked Data Quality Assessment: A Survey, in: Web Services - ICWS - International Conference, Held as Part of the Services Conference Federation, SCF, volume 12994 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 63–76. doi:10.1007/978-3-030-96140-4_5.
[20] H. Chen, G. Cao, J. Chen, J. Ding, A practical framework for evaluating the quality of knowledge graph, Communications in Computer and Information Science 1134 CCIS (2019) 111–122. doi:10.1007/978-981-15-1956-7_10.
[21] J. Debattista, S. Auer, C. Lange, Luzzu—A Methodology and Framework for Linked Data Quality Assessment, Journal on Data and Information Quality (JDIQ) 8 (2016). doi:10.1145/2992786.
[22] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, Databugger: a test-driven framework for debugging the web of data, in: International World Wide Web Conference, ACM, 2014, pp. 115–118. doi:10.1145/2567948.2577017.
[23] M. A. Pellegrino, A. Rula, G. Tuozzo, Kgheartbeat: An open source tool for periodically evaluating the quality of knowledge graphs, in: The Semantic Web - International Semantic Web Conference, volume 15233 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 40–58. doi:10.1007/978-3-031-77847-6_3.
[24] C. Fürber, M. Hepp, SWIQA - A Semantic Web Information Quality Assessment Framework, in: European Conference on Information Systems, 2011, p. 76.
[25] W3C, Shapes Constraint Language (SHACL), https://www.w3.org/TR/shacl/, 2017. W3C Recommendation.
[26] E. Prud'hommeaux, J. E. L. Gayo, H. R. Solbrig, Shape expressions: an RDF validation and transformation language, in: Proceedings of the International Conference on Semantic Systems, SEMANTiCS, ACM, 2014, pp. 32–40. doi:10.1145/2660517.2660523.
[27] K. Rabbani, M. Lissandrini, K. Hose, SHACL and shex in the wild: A community survey on validating shapes generation and adoption, in: Companion of The Web Conference 2022, Virtual Event / Lyon, France, April 25 - 29, 2022, ACM, 2022, pp. 260–263. doi:10.1145/3487553.3524253.
[28] B. Spahiu, A. Maurino, M. Palmonari, Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL, in: Proceedings of the Workshop on Ontology Design and Patterns (WOP) co-located with International Semantic Web Conference (ISWC), volume 2195 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018, pp. 52–66.
[29] K. Rabbani, M. Lissandrini, K. Hose, SHACTOR: Improving the Quality of Large-Scale Knowledge Graphs with Validating Shapes, in: Companion of the International Conference on Management of Data, SIGMOD/PODS, ACM, 2023, pp. 151–154. doi:10.1145/3555041.3589723.
[30] I. Boneva, J. Dusart, D. Fernández-Álvarez, J. E. L. Gayo, Shape Designer for ShEx and SHACL constraints, in: Proceedings of the ISWC Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with International Semantic Web Conference (ISWC), volume 2456 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 269–272.
[31] A. Cimmino, A. Fernández-Izquierdo, R. García-Castro, Astrea: Automatic generation of SHACL shapes from ontologies, in: The Semantic Web - International Conference, ESWC, Proceedings, volume 12123 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 497–513. doi:10.1007/

978-3-030-49461-2_29.
[32] H. J. Pandit, D. O'Sullivan, D. Lewis, Using Ontology Design Patterns to Define SHACL Shapes, in: Proceedings of the Workshop on Ontology Design Patterns (WOP) co-located with the International Semantic Web Conference (ISWC), 2018.
[33] X. Duan, D. Chaves-Fraga, O. Derom, A. Dimou, SCOOP All the Constraints' Flavours for Your Knowledge Graph, in: Proceedings of the Extended Semantic Web Conference (ESWC), volume 14665 LNCS, Springer, 2024, pp. 217–234. doi:10.1007/978-3-031-60635-9_13.
[34] D. Fernandez-Álvarez, J. E. Labra-Gayo, D. Gayo-Avello, Automatic extraction of shapes using sheXer, Knowledge-Based Systems 238 (2022) 107975. doi:10.1016/J.KNOSYS.2021.107975.
[35] X. Yang, H. Zhang, J. Li, S. Li, A Method for Data Quality Validation Based on Shapes Constraint Language, Proceedings - International Conference on Big Data, Information and Computer Network, BDICN (2023) 83–87. doi:10.1109/BDICN58493.2023.00024.
[36] M. J. Luthfi, F. Darari, A. C. Ashardian, SoCK: SHACL on Completeness Knowledge, in: Proceedings of the Workshop on Ontology Design and Patterns (WOP) co-located with the International Semantic Web Conference (ISWC), CEUR-WS.org, 2022.
[37] C. Cortés, L. Ehrlinger, L. Etcheverry, F. Naumann, Is SHACL Suitable for Data Quality Assessment?, 2025. URL: https://arxiv.org/abs/2507.22305. arXiv:2507.22305.
[38] P. Y. Vandenbussche, J. Umbrich, L. Matteis, A. Hogan, C. Buil-Aranda, SPARQLES: Monitoring Public SPARQL Endpoints, Semantic Web 8 (2017) 1049–1065. doi:10.3233/SW-170254.
[39] N. Mihindukulasooriya, R. García-Castro, A. Gómez-Pérez, LD Sniffer: A Quality Assessment Tool for Measuring the Accessibility of Linked Data, in: Knowledge Engineering and Knowledge Management - EKAW Satellite Events, EKM and Drift-an-LOD, volume 10180 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 149–152. doi:10.1007/978-3-319-58694-6_20.
[40] A. Langer, V. Siegert, C. Göpfert, M. Gaedke, Semquire - assessing the data quality of linked open data sources based on DQV, in: Current Trends in Web Engineering - ICWE International Workshops, MATWEP, EnWot, KD-WEB, WEOD, TourismKG, volume 11153 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 163–175. doi:10.1007/978-3-030-03056-8_14.
[41] D. Kontokostas, A. Zaveri, S. Auer, J. Lehmann, TripleCheckMate: A Tool for Crowdsourcing the Quality Assessment of Linked Data, in: Knowledge Engineering and the Semantic Web - International Conference, volume 394 of *Communications in Computer and Information Science*, Springer, 2013, pp. 265–272. doi:10.1007/978-3-642-41360-5_22.
[42] Y. Yamamoto, A. Yamaguchi, A. Splendiani, YummyData: providing high-quality open life science data, Database J. Biol. Databases Curation 2018 (2018). doi:10.1093/DATABASE/BAY022.
[43] E. Ruckhaus, M. Vidal, S. Castillo, O. Burguillos, O. Baldizan, Analyzing Linked Data Quality with LiQuate, in: The Semantic Web: ESWC Satellite Events, volume 8798 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 488–493. doi:10.1007/978-3-319-11955-7_72.
[44] D. Pizhuk, L. Ehrlinger, G. Denk, V. Geist, A data quality dashboard for (security) knowledge graphs, in: Datenbanksysteme für Business, Technologie und Web (BTW), Fachtagung des GI-Fachbereichs, Datenbanken und Informationssysteme" (DBIS), Proceedings, volume P-361 of *LNI*, Gesellschaft für Informatik e.V., 2025, pp. 803–810. doi:10.18420/BTW2025-45.
[45] A. Flemming, Qualitätsmerkmale von Linked Data-veröffentlichenden Datenquellen, Master's thesis, Universität Leipzig, 2011. Diplomarbeit (Quality Criteria for Linked Data Sources).
[46] D. Berrueta, J. Phipps, Cool URIs for the Semantic Web, W3C Recommendation, W3C, 2008. Accessed: 2025-07-24.
[47] G. A. Bernadette Hyland, B. Villazón-Terrazas, Best Practices for Publishing Linked Data, W3C Working Group Note, W3C, 2014. Accessed: 2025-07-24.
[48] C. Cortés, SHACL-DQA-prototype, https://github.com/HPI-Information-Systems/SHACL-DQA, 2025. GitHub repository.
[49] C. Cortes, Datasets used for SHACL-based Data Quality Assessment prototype, https://doi.org/10.5281/zenodo.16644385, 2025.
[50] TopQuadrant, DASH - Data Shapes, https://datashapes.org/dash, 2024. Accessed: 2025-07-29.
[51] W3C SHACL Community Group, SHACL Advanced Features, https://www.w3.org/TR/shacl-af/,

2023. Accessed: 2025-07-29.
[52] T. Hartmann, B. Zapilko, J. Wackerow, K. Eckert, Validating RDF Data Quality Using Constraints to Direct the Development of Constraint Languages, Proceedings - IEEE International Conference on Semantic Computing (ICSC) (2016) 116–123. doi:10.1109/ICSC.2016.43.
[53] A. Kraft, R. Usbeck, The lifecycle of "facts": A survey of social bias in knowledge graphs, in: Proceedings of the Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, 2022, pp. 639–652. doi:10.18653/V1/2022.AACL-MAIN.49.
[54] S. Mohammed, L. Ehrlinger, H. Harmouch, F. Naumann, D. Srivastava, The Five Facets of Data Quality Assessment, SIGMOD Record 54 (2025) 18–27.