TU WIEN Informatics

Extraktion von SHACL Shapes für Evolving Knowledge Graphs

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Eva Pürmayr, BSc
Matrikelnummer 11807199

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr.-Ing. Dipl.-Inf. Katja Hose

Wien, 1. Jänner 2024

Eva Pürmayr	Katja Hose

TU WIEN Informatics

SHACL Shape Extraction for Evolving Knowledge Graphs using QSE

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Business Informatics

by

Eva Pürmayr, BSc
Registration Number 11807199

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr.-Ing. Dipl.-Inf. Katja Hose

Vienna, January 1, 2024

Eva Pürmayr	Katja Hose

Erklärung zur Verfassung der Arbeit

Eva Pürmayr, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Jänner 2024

Eva Pürmayr

Danksagung

Ihr Text hier.

Acknowledgements

Enter your text here.

Kurzfassung

Ihr Text hier.

Abstract

Überarbeiten am Schluss, kopiert von Proposal

Graph databases serve as a foundation for knowledge graphs, which have a broad range of applications. Ensuring data quality is vital and SHACL can be used as a validation language for this purpose. Previously, an approach called QSE (quality shapes extraction) which automatically extracts SHACL shapes from large datasets, has been released. An extension to this program is called Shactor, a web-based tool that visualizes the extraction process and provides statistics. Since knowledge graphs are not static, there exist different versions of a graph, maybe with minimal changes. There is a lack of tools to compare shapes between these graph versions. To address this gap, the proposed solution involves creating a web-based tool and algorithm adaptions in this niche.

Contents

CHAPTER 1

Introduction

1.1 Problem Statement

will be written in the next version

CHAPTER 2

Related Work

This chapter delivers a comprehensive overview of knowledge graphs, RDF, SHACL, and SPARQL. Building on this foundation, a dedicated section explores evolving knowledge graphs, which is particularly important to this subject. Additionally, the chapter presents related work retrieved from a systematic literature review, along with a description of the QSE algorithm which forms the basis of this thesis.

2.1 Preliminaries

2.1.1 Knowledge Graphs

The topic of this thesis is located in the area of semantic systems and knowledge graphs. Semantic systems are systems that make use of explicitly represented knowledge, through conceptual structures such as ontologies, taxonomies or knowledge graphs [51]. Many formal definitions exist for knowledge graphs, such as the one articulated by professors at TU Wien: "A knowledge graph contains semantically related information as nodes and edges" or "Knowledge Graphs are graph-structured representations intended to capture the semantics about how entities relate to each other" [72]. Knowledge graphs find application across diverse domains, including commercial and academic sectors. Moreover, they serve as the foundation for Data Science, Graph Databases, Machine Learning, Reasoners and Data Integration/Wrangling. Noteworthy among these knowledge graphs is DBpedia, which contains cleaned data from Wikipedia in all languages. In 2023, DBpedia's Databus, the platform hosting its data, provided 4,100 GByte of data [4]. Additionally, other prominent knowledge graphs exist, such as Wikidata, compromising approximately 110 million editable records [17]. Wikidata serves as the central repository for Wikimedia projects like Wikipedia, Wikivoyage, Wiktionary, Wikisource and others. It is a free and open knowledge base. Besides general information, as it is provided by DBpedia, knowledge graphs have found compelling applications in fields such as chemistry and

biology. For instance, Linked Life Data offers access to 25 public biomedical databases, facilitating questions like "give me all human genes located in Y-chromosome with the known molecular interactions" [9]. Around 10 billion RDF statements are provided by Linked Life Data. Finally, PubChem, an open chemistry database, is another notable example of a large knowledge graph [73].

2.1.2 RDF

Knowledge graphs can be expressed as Property Graphs or by the Resource Description Framework [51]. The primary distinction lies in the fact that Property Graphs allow edges to have attributes. However, this thesis focuses on the Resource Description Framework, which was developed in the 1990s and is a W3C recommendation [11]. An RDF graph contains numerous statements, known as triples, each consisting of a subject, predicate and object. An example is illustrated in Figure 2.1. An important term in

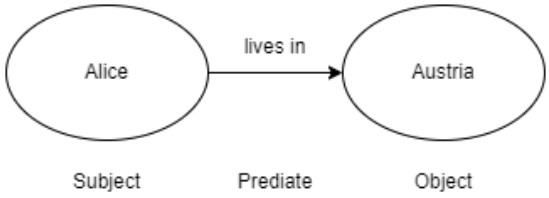

Figure 2.1: Example of RDF

knowledge graphs is URI, which stands for Uniform Resource Identifier and can be a Unifrom Resource Name (URN), a Uniform Resource Locator (URL) or a combination of both. An Internationalized Resource Identifier (IRI) is a URI. An example of defining a resource using a URI could be "http://semantics.id/ns/example#Alice".
The subject of an RDF statement is a resource that may be identified with a URI, the predicate is a URI-identified specification of the relationship between subject and object. The object is a resource or a literal. A literal can be a text or another datatype (such as integer, decimal, date or boolean), but it does not have a URI. An adapted example could have the predicate "age" and the object "25", where the object is a literal, unlike the previous example where the object is a resource. Besides resources and literals, blank nodes exist, that do not have a URI and are used for unnamed objects like lists. A small example of a full graph can be seen in Listing 2.1.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://example.org/alice#me> a foaf:Person .
<http://example.org/bob#me> a foaf:Person .
<http://example.org/alice#me> foaf:name "Alice" .
<http://example.org/alice#me> foaf:knows
    <http://example.org/bob#me> .
```

```
<http://example.org/bob#me> foaf:knows
    <http://example.org/alice#me> .
<http://example.org/bob#me> foaf:name "Bob" .
```

Listing 2.1: RDF graph in Turtle syntax

Prefixes can be used in RDF to shorten URIs. For instance, "foaf:name" abbreviates "<http://xmlns.com/foaf/0.1/name>". A dot always indicates the end of an RDF statement.
With RDF triples complex graphs can arise since there is no pre-configured, fixed schema. RDF graphs can be represented in various formats, such as RDF/XML, N3/Turtle or in N-Triples format, which is shortened as "nt". An example of an RDF graph in N-Triples format based on Listing 2.1 in a shortened version is shown in Listing 2.2.

```
<http://example.org/alice#me>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .
<http://example.org/alice#me>
<http://xmlns.com/foaf/0.1/name> "Alice" .
<http://example.org/alice#me>
<http://xmlns.com/foaf/0.1/knows> <http://example.org/bob#me> .
```

Listing 2.2: RDF graph in N-Triples Format

Knowledge graphs can be represented by different graph engines, for instance, GraphDB. GraphDB is provided by Ontotext, available as a free or commercial version. It offers a visual interface for the interaction with graphs but also supports integration with various programming languages [7]. Alternatives like Apache Jena or Eclipse RDF4j are solely compatible with Java [2, 34].

2.1.3 SPARQL

Knowledge graphs can be queried by the query language SPARQL [14]. SPARQL shares similarities with SQL, as it incorporates familiar keywords like "select", "where", "group by" and "order by". An example query, as shown in Listing 2.3, retrieves the names of people along with the number of their friends.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name (COUNT(?friend) AS ?count)
WHERE {
    ?person foaf:name ?name .
    ?person foaf:knows ?friend .
} GROUP BY ?person ?name
```

Listing 2.3: SPARQL example

The WHERE clause specifies a pattern to be matched in the graph. The variable "?person" represents any object in the graph and is further constrained by the statements, that the person must have a name and a friend. SPARQL offers various other functionalities, including insert, update and delete statements, filtering, ordering and subqueries.

2.1.4 SHACL

Since data quality is an important topic in all areas SHACL, which stands for Shapes Constraint Language, has evolved as a language to validate RDF graphs against a certain set of conditions [12]. SHACL can be expressed as an RDF graph itself, it can also serve as the description of the data graph. This representation is termed a "shapes graph", whereas the actual dataset is called the "data graph". Shapes graphs contain different shapes, which describe certain classes or their properties.
A key concept within SHACL is the node shape which is typically used to describe a class in the RDF graph and may contain multiple property shapes. A property shape is usually used to describe a property of a certain class. An illustrative example is presented in Listing 2.4, where the validation of the "ex:Person" class is specified. Every person should have at least one name.

```
ex:PersonShape a sh:NodeShape ;
sh:targetClass ex:Person ;
sh:property [
    a sh:PropertyShape;
    sh:path ex:name;
    sh:minCount 1
] .
```

Listing 2.4: SHACL example

SHACL offers various options such as "pattern", "nodeKind", "class", "datatype" and others, enabling a detailed description of a graph's schema.
There exist also other logic-based languages, which can describe the schema of graph, namely the Web Ontology Language (OWL) or Shape Expressions (ShEx) [10, 13].

2.2 Background

2.2.1 Evolving Knowledge graphs

As one might expect, knowledge graphs are not static; they evolve over time just like the reality they present. DBpedia, for instance, is updated regularly as new knowledge is generated continuously.
Time can be represented in evolving knowledge graphs in different ways [72]. It can be saved as data, for example, the construction year of a building can be recorded in the knowledge graph. Contrastingly, time can also be saved as metadata. This includes the creation time of an entry, the time an entry was deleted, and the existence of different

versions of an entry over time. Consequently, when time is saved as metadata, a knowledge graph can be dynamic − providing access to all observable atomic changes over time − or versioned − offering static snapshots of the knowledge graph at specific points in time. In this thesis, only versioned knowledge graphs will be relevant.
When discussing evolution, several new terms must be considered. First, structural evolution can be measured, where different descriptive statistics, such as centrality or connectedness are measured over time within a graph. Dynamics in an evolving knowledge graph can be assessed by examining growth and change frequencies, which can be interesting to compare across different areas of the graph. Timeliness refers to the freshness of the data, indicating if data is out-of-date or out-of-sync. Monotonicity describes whether data is only added to the graph or if there are also updates and deletions. Semantic drift happens when there is a change in the meaning of a concept over time.
It is also important to consider who makes changes in a knowledge graph. These changes can be made by anonymous users, registered users, authoritative users, or bots. Understanding who is making the changes is crucial, as it can impact data quality in various ways.
Often, data in knowledge graphs undergoes little to no changes, and when changes do occur, they are primarily at the level of object literals [72]. Changes can be atomic, focusing on operations at the resource level, or they can be local or global. Changes can also be monitored at the schema level. In a study of the DyLOD dataset (Dynamic Linked Data Observatory) over one year, between 20% and 90% of schema structures changed between two versions. This indicates that while some nodes retained the same schema structure, new schema structures were added, and some were no longer used. This observation is particularly important for this thesis.
However, the dimension of time opens new challenges in knowledge graphs. For instance, there is a need for new data models and storage methods tailored to evolving knowledge graphs. Additionally, there will be novel approaches to query processing, reasoning and learning that incorporate the temporal aspect in evolving knowledge graphs.

2.2.2 Results from Literature Review

There has been a lot of research going on in the areas of Evolving Knowledge graphs, leading to various approaches addressing diverse topics. For instance, EvolveKG is a framework designed to reveal cross-time knowledge interactions [61]. Another approach involves creating a summary graph for different versions of objects, which refers to the dynamic option of evolving knowledge graphs, as discussed earlier [81]. Additionally, the software KGdiff has been developed to track changes in both the schema and the individual data points [58].
Numerous approaches have also been developed in the area of data quality. GraphGuard introduces a framework aimed at improving data quality in knowledge graph pipelines for both humans and machines [36]. Additionally, there has been a systematic review of quality management in knowledge graphs, as well as a general survey on knowledge graphs, including the temporal aspects of them [87, 55]. Another paper aims to enhance

existing quality assessment frameworks by incorporating additional quality dimensions and metrics [52]. Also in the area of quality, a method has been developed to judge, whether an incoming graph change is correct or not with classifiers based on topographical features of a graph [65]. PAC (property assertion constraints) have the goal of checking data before it gets added to the knowledge graph. With this approach, errors can be prevented and the quality of knowledge graphs can be enhanced. PAC works by restricting the range of properties using SPARQL [35]. Additionally, a dissertation has focused on incorporating data transformations in knowledge graphs to clean the data and complete the graphs by calculating derived data [32].
Several projects have been developed to automatically derive SHACL shapes or ontologies from existing graphs, eliminating the need for manual schema creation. One such project is SCOOP, which extracts SHACL shapes from existing knowledge graphs, primarily using ontologies and data schemas rather than individual entities [37]. IOP is a predicate logic formalism that identifies specific shapes over connected entities in the knowledge graph, with its corresponding learning method known as SHACLearner [8]. SheXer, using a Python library, produces SHACL shapes similar to QSE [41]. A logic called ABSTAT which automatically extracts SHACL shapes from knowledge graphs has already been used in QSE [20]. Another approach, the Ontology Design Pattern, uses ontologies to automatically generate SHACL shapes. Astrea employs Astrea-KG, which provides mappings between ontology constraint patterns and SHACL constraint patterns [67, 29]. While not generating SHACL shapes itself, Trav-SHACL plans the execution and traversal of a shapes graph to detect invalid entries early [43].
Visualizing SHACL shapes has also been explored. A master's thesis was written that generates a 3D model of interconnected SHACL shapes [20].

2.2.3 Quality Shapes Extraction

QSE (Quality Shapes Extraction) is an algorithm designed to extract SHACL shapes from knowledge graphs to ensure data quality [74]. It is particularly designed for very large knowledge graphs and it is written in Java. During calculation, QSE calculates two parameters: support and confidence. The support parameter determines the number of entities associated with a certain class for a node shape while for a property shape, it defines the cardinality of entities conforming to that class. The confidence parameter measures the ratio between the number of entities that conform to a property shape and the total number of entities that are instances of the target class.
QSE is available as a command-line tool, primarily designed to extract SHACL shapes from graphs provided in "nt"-Format. Additionally, QSE can be used on a graph store like GraphDB through a query-based option.
The QSE algorithm involves two iterations through all entities. During the initial phase (entity extraction), all instances based on their type declarations are counted. Subsequently, in the second run (entity constraints extraction), the algorithm gathers metadata for property shapes. Following this, support and confidence metrics are computed and finally, in the last step, the algorithm generates SHACL shapes.

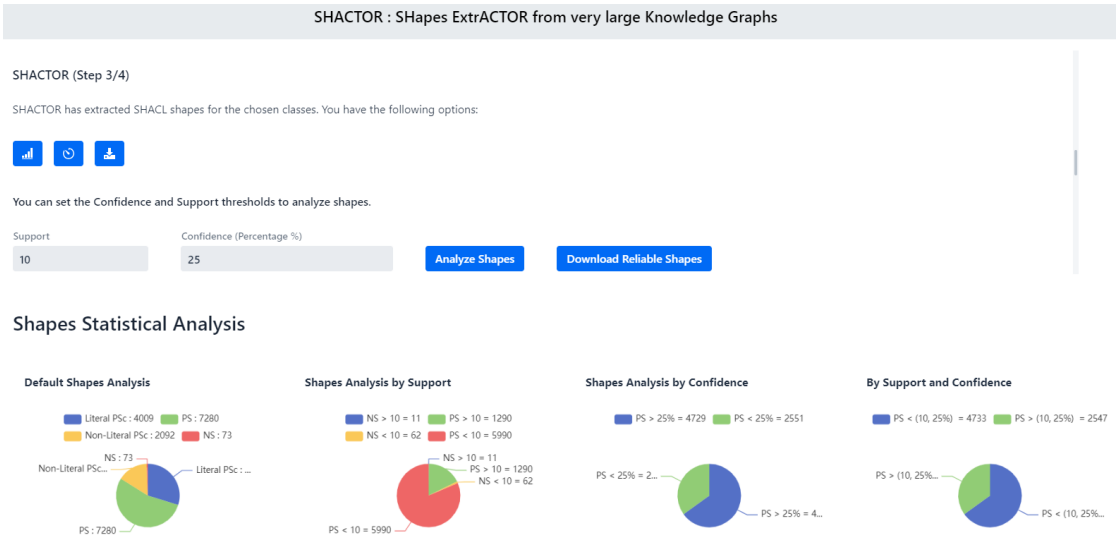

Figure 2.2: Statistics provided by Shactor after analyzing version 1 of the Bear-B dataset

Shactor

The visual extension to QSE is called Shactor [75]. It is a web-based tool that visualizes the extraction process and provides valuable statistics. Shactor was developed using Vaadin and supports only the file-based version of QSE. An example screenshot after the extraction process is shown in Figure 2.2

image could be bigger

Other shapes extraction algorithms

Many algorithms follow the idea of extracting SHACL shapes or other schema information from existing data sources to improve data quality. Additionally to the projects mentioned in Subsection 2.2.2, examples are SHACLGEN or ShapeDesigner, which follow a similar approach as QSE-exact but lack the ability to process large knowledge graphs [74, 24, 56]. However, there exist even more algorithms, that automatically extract a schema from data, but have several other shortcomings [64, 41, 80, 15, 68, 30, 66].

CHAPTER 3

Methods

This chapter describes the scientific methods that are used in this work. First, all methods are described individually, then the research questions are presented. At the end, the process of answering the research questions using the mentioned methods is explained.

3.1 Research Methods

Delete Research Methods again and keep only Research Questions and Methodological steps?

3.1.1 Design Science Research

Design Science, rooted in engineering, focuses on problem-solving through creation of useful artifacts [50]. In Design Science Research design is both a process and a product and the evaluation of both is crucial. This leads to a continuous cycle where artifacts undergo regular evaluation and subsequent refinement, often referred to as the Design Cycle [49].
Two important terms in design science are relevance and rigor. The artifact's relevance to its environment (people, organizations, technology) is crucial. This relationship forms a cycle: the artifact's development is shaped by the environmental needs and, in turn, influences that environment through its application, termed the Relevance Cycle. Simultaneously, ensuring rigorous development involves addressing existing knowledge. Here, the knowledge base influences the design cycle, while the finished product contributes back to the knowledge base in what's known as the Rigor Cycle.
The heart of Design Science is the iterative Design Cycle, with Relevance and Rigor Cycle as its inputs. These cycles depend on each other, although during the actual research execution, the Design Cycle remains relatively independent.
Hevner proposed seven guidelines for the design science framework:

1. Design as artifact: an artifact (construct, model, method, instantiation) must be produced

2. Problem relevance: ensuring the solution is relevant to business problems
3. Design evaluation: evaluation methods must be executed to ensure the utility, quality and efficacy
4. Research contributions: contributions in the area of design artifact, design foundations and/or design methodologies must be provided
5. Research rigor: employing rigorous methods during construction and evaluation
6. Design as a Search Process: searching for an artifact requires available means to reach desired ends while satisfying laws
7. Communication of Research: the result must be presented to a technology-oriented and management-oriented audience

An especially important topic within the design cycle revolves around evaluation [70]. This involves establishing specific criteria, such as functionality, performance or usability as benchmarks for assessment. The study has evaluated several artifact types (e.g. algorithm, instantiation, construct) and evaluation method types (e.g. expert evaluation, technical experiment, prototype). The choice of evaluation method depends on the type of artifact; technical experiments were predominantly used for algorithms, instantiations, methods, and models.

3.1.2 Systematic Literature Review

Both rigor and relevance cycle require an understanding of the current state of the art. Particularly, the relevance cycle relies on identifying existing research gaps, a task that will be addressed through a partial systematic literature review (SLR) [59]. Typically, a systematic literature review includes the following steps:

1. Planning: This phase begins by identifying the need for a SLR, (optionally) commissioning the SLR and defining research questions. After that, the review protocol must be developed and evaluated, which includes search terms, resources where to search, study selection criteria and a quality checklist. A pilot phase helps refine these elements.
2. Conducting: Firstly, the research must be identified and primary studies should be selected. After that the study quality must be assessed, followed by data extraction, monitoring and synthesis.
   It is important for the search terms to also consider synonyms, abbreviations and combinations. Documentation throughout ensures transparency and replicability. After all primary papers have been obtained, the papers must be filtered. Instead of reading all the papers, some papers can be dropped based on the titles and then on the abstracts (study selection criteria). Then the remaining papers can be read and further reduced. In the end, data needs to be extracted with a pre-defined

form, if possible, by multiple reviewers. During data synthesis, data is summarized. This can be done descriptively, qualitatively or quantitatively.

3. Reporting: The last step involves specifying the dissemination mechanism (how and where the results should be published, e.g. in a magazine, journal articles or on a website), formatting and evaluating the report. During the evaluation, peer reviewing can be used e.g. for journal articles. Quality checklists and other prepared documents from the planning phase can be utilized here.

3.1.3 Prototyping

Within the design cycle of Design Science Research (DSR), an artifact is developed. Prototyping has parallels to software engineering, which often uses agile development cycles with frequent feedback [54]. The primary aim is to construct the Minimum Viable Product before completing the development process, aligning with the Plan-Do-Check-Act methodology.
Camburn et al. describe prototypes as a pre-production representation of some aspect of the final design [25]. Prototypes can be used for refinement, communication, exploration of new concepts and learning. Noteworthy guidelines include testing, timing, ideation, fixation, feedback, usability and fidelity. Various prototyping techniques exist, such as iterative, parallel, requirement relaxation or subsystem isolation, each offering distinct advantages for particular objectives. Despite the diverse techniques, detailed guidance on prototyping remains limited, except for the iterative aspect.

3.1.4 Semi-structured expert interviews

Semi-structured interviews (SSIs) are a qualitative evaluation method within the design cycle, allowing feedback integration into the construct phase [19]. Unlike traditional surveys, SSIs engage fewer participants, employing open-ended questions that focus on 'how' and 'why' and the structure is not fixed. Disadvantages of SSIs are that they are time-consuming, labor intensive and require interviewer sophistication. SSIs prove beneficial when seeking open responses and individual perspectives, suitable for groups like program recipients or interested stakeholders.
The methodological steps include selecting respondents and arranging interviews. The target group should be identified, researchers choose ordinarily a manageable random sample. A brief introduction letter detailing the interview's time frame is crucial before sending invitations. This should be piloted before sending. Next, the questions should be drafted and an interview guide should be created. There should not be too many issues on the agenda. Closed-ended questions can be an ideal start, open-ended questions can follow these questions. Attention to translations and avoiding questions that evoke pressure to give socially acceptable answers is essential. Flexibility during the interview allows agenda adaptation, keeping easy questions at the start, challenging ones at the end, and demographic inquiries at the interview's conclusion.
During the interview, the interviewer should establish a positive first impression and

ask for permission to record the interview. This allows the interviewer to participate more actively. Preparation ensures clear prioritization of questions. Interviewers should maintain a calm, listen actively and seek for clarification when necessary.
The analysis involves visualizing closed-ended responses in tables/graphs and employing qualitative content analysis to categorize open-ended ones.

3.1.5 Qualitative Content analysis

This is an approach of systematic, rule guided qualitative text analysis which can be used in the design cycle of DSR [63]. It is used for fitting text material into a model of communication. The aspects of text interpretation will be put into categories, which follow the research question. There are two ways for category development, namely inductive and deductive category development. Inductive category development forms the categories out of the text, whereas deductive formulates the categories more on theory. The detailed steps of inductive category development involve, based on the research question, the determination of the category definition. Then, categories are formulated based on the material. after 10-50% of the material, the categories are revised and the process starts again. After working through all the material, everything should be checked on reliability and the results can be interpreted, which can trigger another iteration.

3.1.6 Algorithmic Design

During the design cycle, another method applicable is Algorithmic Design, also known as Algorithm Engineering, a scientific approach to developing efficient algorithms [78]. The method involves iterative steps: design, analysis, implementation, and experiments. A key focus lies in crafting algorithms that are simple, implementable, and reusable, with performance being a crucial factor.
It is crucial in the analysis phase, that the design algorithm is easy to analyze, to close the gap between theory and practice. Implementing the algorithm demands attention to nuances across programming languages and hardware, as minor details can yield significant differences. The last step is experiments, which can influence the analysis again. Many experiments can be done with relatively little effort. However, they also require nontrivial planning, evaluation, archiving and interpretation of the results. The starting point should be a falsifiable hypothesis, which can then be used for the next iteration of the cycle.
Realistic models serve as inputs during the design phase, representing abstractions of application problems. During the experimentation phase, real inputs are required. Most of these steps are closely related to applications.

3.1.7 Technical experiments

Design of Experiments is a statistical tool that can be used for planning and conducting experiments. [22]. A practical methodology can be split into the planning, designing, conducting and analyzing phase [38]. The planning phase requires a clear, objective,

specific, and measurable statement of the problem. A meaningful response characteristic should be carefully chosen, there can also be multiple. Furthermore, process variables and design parameters must be selected, mostly they are set from knowledge of historical data. The next steps are classification and determining the level of process variables, interaction between all variables must also be known.
The design phase is highly individualized, depending on many factors. Key principles to consider here include randomization, replication, and blocking [21].
In the conducting phase, the experiment is carried out and the results are evaluated. It is important to document everything during the experiment, even unusual occurrences. The analysis phase involves interpreting results and drawing conclusions based on the collected data.

3.2 Research Questions

Given two versions of a graph (V1 and V2) and the corresponding SHACL shapes generated by QSE (S1 and S2):

RQ1: What is an appropriate way to compare S1 and S2 in a user-friendly way and explain the differences (addition, removal, change)?

RQ2: Given S1 and the changeset between V1 and V2, how can we use the changeset and S1 to derive S2?

RQ3: What is an appropriate extension of the QSE algorithm so that - after executing standard QSE on V1, we can parse V2 to obtain the changed shapes without having to run the complete QSE algorithm again on V2?

RQ4: Given SPARQL endpoints hosting the graphs, how can we use SPARQL queries to derive which shapes of S1 remain unchanged and which were removed for V2?

Appropriate for RQ1 means user-friendly, useful and correct. It can be measured during the evaluation of semi-structured interviews with experts. Appropriate for RQ2, RQ3 and RQ4 means correct and faster in comparison to the method described in the evaluation section.

3.3 Methodological steps

To answer these research questions, the framework of Design Science Research is be used. As previously discussed, Design Science Research uses three cycles. In this thesis, a Systematic Literature Review serves as the methodology for the relevance and rigor cycle. Therefore, business needs can be derived from the literature and knowledge from the rigor cycle can influence the thesis. Ultimately, this process contributes to the knowledge base and aligns with evolving business needs by the cycle's conclusion.

3.3.1 Systematic Literature Review

Given the time-consuming nature of a Systematic Literature Review, a partial review was chosen. During the planning phase, a review protocol was established and only one online library was defined, where strict selection criteria were applied, to limit the number of papers. In the conducting phase, qualitative data extraction was done, aiming to provide a broad overview of the knowledge base and business needs rather than addressing a specific research question. Reporting was done in a simplified way.
The review protocol contained the following items:

- Objectives: Establish a broad understanding of the current knowledge base in the area of evolving knowledge graphs, automatic extraction of data quality constraints from knowledge graphs and comparing of SHACL shapes and differences between knowledge graphs
- Search Keywords: “evolving knowledge graphs”, “data quality in knowledge graphs”, “shacl extraction from knowledge graphs”, “comparing shacl shapes”, “differences between knowledge graphs versions”
- Study selection criteria: published after 2000, written in English or German, accessible with TU-Vienna Account for free
- Library: Google Scholar
- Study Exclusion Criteria: Has nothing to do with research questions or objectives
- Procedure: In the conduct phase, the first ten results for each search keyword were evaluated. The outcomes of this phase are detailed in Table A.1 in Appendix A. After reviewing the titles and applying the exclusion criteria, the abstracts of the papers were read and each paper was rated for relevance on a scale from 1 (very relevant) to 3 (not relevant). The results of this phase are presented in Table A.2 in Appendix A. Papers with a high relevance (rated 1) were further assessed by examining parts of the paper or the entire paper.
- Data Extraction: Based on their abstracts and in some cases the full paper, key sections of the content have been summarized.
- Data Synthesis: The content of the papers was descriptively synthesized.
- Data Reporting: The results of the Systematic Literature Review are presented in Section 2.2.

3.3.2 Prototyping

During the design cycle, RQ1 was answered using the prototyping method within the construct phase [54]. JustInMind, a design and prototyping tool for web and mobile apps, was chosen for this purpose [6]. The free version was sufficient for this task, as

the offered features were sufficient. The initial stages involved low-fidelity prototypes to determine the most suitable design that satisfies the end user's requirements. Although only one prototype was developed, this one was already pretty similar to the final web app. Examples of this prototype are showcased in Figure 3.1 and Figure 3.2. Iteratively, the minimum viable product was developed, incorporating feedback from the evaluation phase.

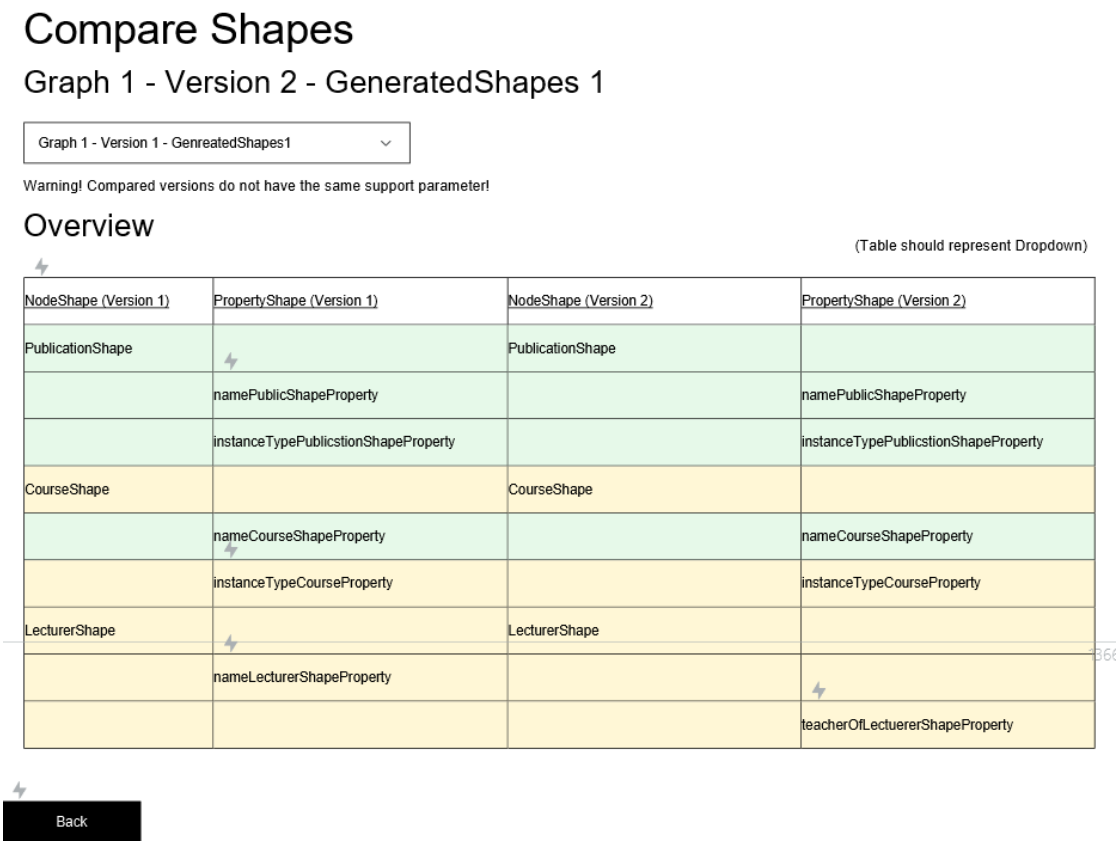

Figure 3.1: Prototype of a comparison

3.3.3 Algorithmic Design

Not finished, adjust, after algorithms are implemented

Contrastingly, RQ2, RQ3 and RQ4 used algorithmic design in the construct phase of the design cycle [78]. The approach entails maintaining algorithm simplicity, reusability, and utilizing libraries. Iterative development through small experiments refined the algorithms. As realistic input, graph snapshots from the BEAR datasets and other data sources were used. During development, synthesized data and smaller versions of these graphs were utilized.

3.3.4 Evaluation

Not finished, adjust after the evaluation is done, or move to Evaluation-section?

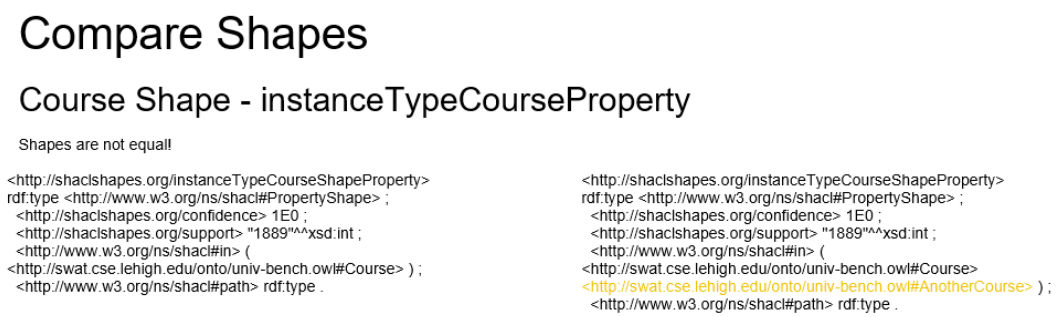

Figure 3.2: Prototype of the comparing a shape in detail

For the evaluation part of the design cycle for RQ1, semi-structured expert interviews with students, who have already participated in Introduction To Semantic Systems were conducted [19]. The important part here was to measure the usability of the Web app in comparison to finding differences across the shapes without the tool. The interviews contained a demonstration of the tool's functionality. Open-ended questions were used to get comprehensive insights about usability. Since this method is time-consuming, 3-5 experts were interviewed. In the planning phase, the interview guide was created with prioritized questions, keeping the length of the interview as short as possible. The format of the interviews, online or offline, was based on the preferences of the experts. It was planned to record the interviews.
The analysis of interview content employed the inductive approach of qualitative content analysis [63]. Depending on the content of the interviews, categories were created.
For the evaluation phase in the design cycle for RQ2, RQ3 and RQ4, technical experiments were conducted, aligning with common practices in DSR projects for instantiations [22] [38]. The experiments ran on a virtual machine since the data size of the test data was huge. After it was ensured, that the algorithm worked correctly, speed became the measurable metric.
The baseline here is the time it takes to generate S1 by using V1, plus the generation time of S2 by using V2. Additionally, the execution time of a script that compares S1 and S2 by their shape names was added to the baseline.

CHAPTER 4

Architecture and Design

Einleitung

4.1 Technical Details

Wissenschaftliche Herangehensweise, wenn Algorithmen fertig sind

4.2 Workflow

Algorithmen hinzufügen

4.3 Functionalities

Gesamtarchitektur, nicht sicher, wie ich das darstellen soll

The main functionalities of the Web app are summarized as a Use-Case diagram in Figure 4.1.

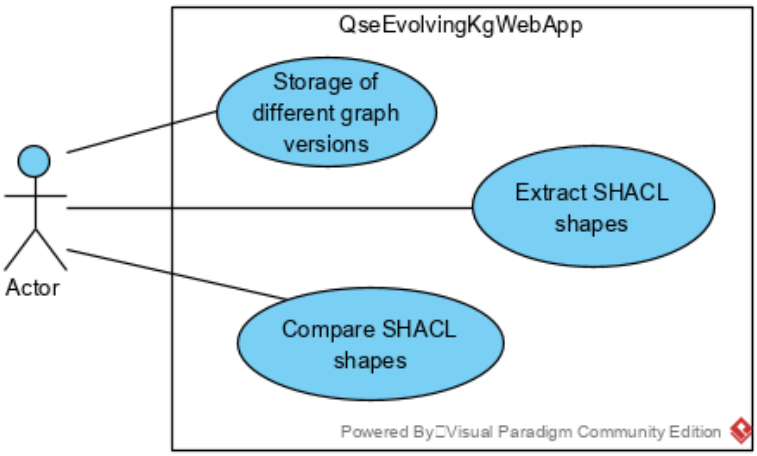

Figure 4.1: Use-Case Diagram for the Web app

Technical Details: nicht sicher, was hier hingehört. Der Teil von Implementation?

Workflow: Ist das mit Functionalities schon abgedeckt?

Not sure how to write Web-App: should be web app in my opinion? Or Web app?

Storage of different graph versions

To make the comparison of graph versions easier, the Web app offers the possibility to save graphs and their respective versions. Each graph can have multiple versions associated with it. For instance, the graph "Bear-B" has versions ranging from 1 to 89. In compliance with QSE, the web app requires that only graphs or versions in N-Triples format can be uploaded. Users have the option to either upload a file or use a pre-configured graph file.
The capability to store graphs and versions improves the user experience of the Web app significantly. Without this feature, users would have to upload files repeatedly for comparisons which would be very inconvenient and consume additional time and effort. Shactor does not include this saving feature [75]. This is specifically convenient as there may arise scenarios where running QSE multiple times on the same graph file is necessary, for instance, with varying support or confidence parameters.

Extract SHACL shapes

When QSE should be run, the user chooses a graph and the desired version. Similar to Shactor, all classes and their instance counts are listed. The user can choose the classes which should be analyzed. Additionally, the user can either extract default shapes (all shapes) or set the support and confidence parameters for pruning. Notably, a node shape does not have a confidence parameter. The program also stores the extracted SHACL shapes, allowing them to be selected for comparison later.

Compare SHACL shapes

Finally, the extracted shapes are available for comparison. Users can choose one or multiple QSE runs along with their extracted shapes for comparison. In general, there is the possibility to compare all kinds of extracted shapes. However, there will be warnings, if extracted shapes with different support or confidence parameters are compared or if extracted shapes with different classes are selected. The order in which the objects are compared can also be chosen. For instance, if there are two extracted shape objects from two QSE runs (S1 and S2), then S1 can be compared with S2, or vice versa, S2 with S1.
For an overview of the comparison, node and property shapes are organized in a tree-view format. Objects from different QSE runs are displayed side by side, with shapes mapped by their name across different QSE runs. Non-matching shapes are highlighted in red. Users can search for specific shapes per name or filter by identical node shapes, identical property shapes or different shapes in general.
By clicking on a row in the comparison overview, users are directed to a detailed view, where the SHACL shapes of the QSE runs are displayed as text. In this detailed view, discrepancies are also visually highlighted in red or green, indicating additions or removals in the text. If multiple shapes were selected for comparison, there is the possibility to select the desired shapes at the top, as only two shapes can be directly compared at a time. Underneath the text comparison, all shapes are listed along with their corresponding support and confidence values, as these were omitted from the original SHACL shape.

When a user selects a row in the overview where a shape was deleted, a prompt will appear, explaining why this shape was excluded. This information is retrieved from the default shapes which are also preserved during shape extraction.

4.4 Contributions to the QSE algorithm

Not finished yet, only notes

Implement and make Pull Request to shactor branch

- Shapes get duplicated after every run in QueryBased: -> deleteOutputDir in runParser (line 74)
- Bug: Query for literals property does not get replaced in QbParser -> incomplete shapes
- Shactor: SAIL repository is locked when executed with all classes and then only with one class e.g. ScriptWriter -> Shactor needs to be restarted

4.5 Limitations in the QSE algorithm

Not finished yet, only notes

- Problems, when same but different IRIs are used: http://dbpedia.org/property/almaMater or http://dbpedia.org/ontology/almaMater -> This was not implemented on purpose, it should be two shapes
- QueryBased: sometimes unclear empty shapes, because of incorrect data
- Shactor: Example Film: If all classes are selected and Support is set to 3 (confidence 1), dateOfBirth (ScriptWriter, support = 1) is also printed. If only ScriptWriter class is selected, dateOfBirth property is not printed.
- When a comment is added at the beginning of the file, everything crashes. E.g. Bear-B first line: #V 2015-08-01T21:33:00Z
- InstanceShapes get filtered in QueryBased, but not in FileBased
- Shactor: Encoding issues for special characters: e.g. <http://dbpedia.org/resource/2015_US_Open_(tennis)> <http://dbpedia.org/ontology/budget> "4.22534E7"^^<http://dbpedia.org/datatype/usDollar>
- Pruning works with >, not with >=, which can be very confusing
- Objects in Shactor only support only one targetClass, not multiple

CHAPTER 5

Implementation

Introduction, wenn all algorithms are implemented

5.1 Technical Details

Rewrite, improve when application is finished

In reference to Shactor, the application was developed with Vaadin (Version 24.2.4), Java (Version 17) and Spring [16, 75]. Vaadin is a full-stack framework that allows front-end development with Java. For this project, the free version of Vaadin was selected as the features provided were sufficient.

5.2 Version of QSE

Shactor utilizes a variant of the QSE algorithm that differs from the main branch. This version generates objects for each node shape, property shape, and similar elements, in addition to producing SHACL shapes as a Turtle file. Consequently, the Web app employs this specific version of QSE, referred to as the "shactor" branch [5]. This version of QSE is also used for the algorithms since the performance impact is minimal. Of course, the modifications described in Section 4.4 were implemented beforehand.

5.3 Web App

To answer RQ1, a Web application has been created that allows users to compare generated SHACL shapes from QSE effectively. This comparison is available across multiple versions of a graph. The goal was to maximize the usability of the web application, enabling the user to see differences as easily as possible.

5.3.1 Demonstration with Screenshots

An extended version of the example from Section 2.1.2 was chosen to showcase the website's capabilities. The complete RDF graphs are available in Appendix B.
The Web app is designed for laptop screens, a full screenshot is illustrated in Figure 5.1 The initial screen of the Web app shows a list of all graphs saved in the application. Each

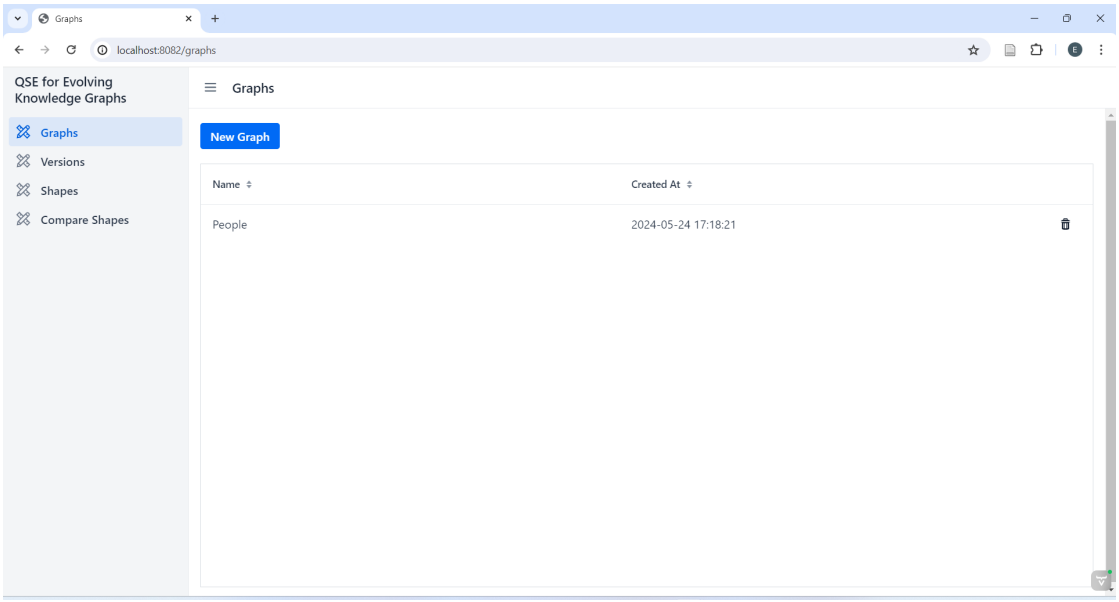

Figure 5.1: Full screenshot of the Web app

graph can have multiple versions associated with it. In this demonstration, the "People" knowledge graph has three versions. The code for these versions of the knowledge graph is fully listed in Listings B.1, B.2 and B.3.
In the first version of the knowledge graph, there are two people, Alice and Bob, who both have a name attribute and an attribute that indicates that they know each other. A third person (Jenny) was additionally added. In the second version of this knowledge graph three cats, each with an associated color, are added. In the third version, the color attribute is removed from two of the three cats, and instead of knowing each other, Alice, Bob and Jenny each now know one cat. The user interface for these functionalities is illustrated in Figure 5.2 and Figure 5.3 and the interface for the graph and version upload is shown in Figure 5.4.
After the knowledge graph versions are saved, the user can continue to extract SHACL shapes. In this example, QSE is run on all three versions for all classes. The support parameter is set to two and the confidence parameter is set to zero for each run. This means that all triples, where two or fewer triples conform to that class are not considered for the resulting shapes. The confidence parameter is set to zero percent, ensuring no triples are dropped, as the confidence is at least 33% when only one of three triples conforms to a property shape. The detailed definitions of the support and confidence

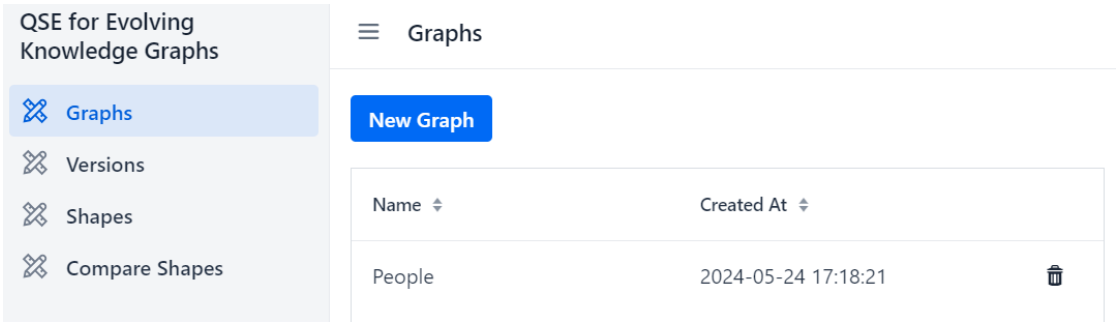

Figure 5.2: Overview of all graphs

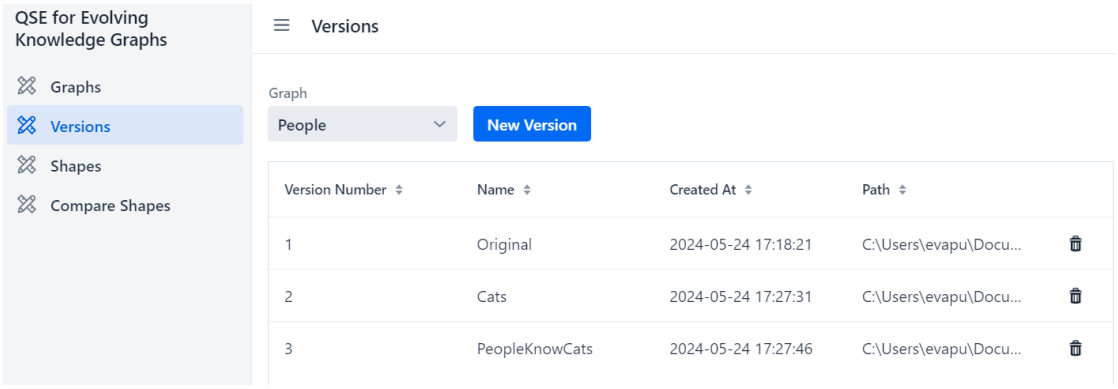

Figure 5.3: Overview of all versions of a graph

parameters are provided in Section 2.2.3. The overview of all extracted shapes for the third version is illustrated in Figure 5.5. In this example, the default shapes are also extracted, as indicated by the support and confidence parameter values of zero. However, this extraction is not relevant to the subsequent example; it simply demonstrates that the program supports multiple shape extractions for a single version. The process of extraction for the second version of the knowledge can be seen in Figure 5.6. QSE automatically lists all classes along with their instance counts for selection. As previously described, this version includes three people and three cats.
Finally, the extracted shapes are available for comparison in Figure 5.7. At the top of the page, the three previously generated QSE runs along with their extracted shapes are chosen. The overview indicates changes for the "knows" property in the class Person, as this property points to Cats in the third version of the graph. Additionally, there are changes for the Cat class.
Clicking on the row "knowsPersonShapeProperty" in the comparison table (Figure 5.8) redirects to a more detailed comparison view, where the discrepancies are visually highlighted in red and green, indicating additions or removals in the text. "Cat" is written in green, as the targeted class is now a "Cat" and not "Person" anymore, which is written in red. Below the comparison, all SHACL shapes are listed along with their corresponding support and confidence values. In this example, the text for the first version was removed,

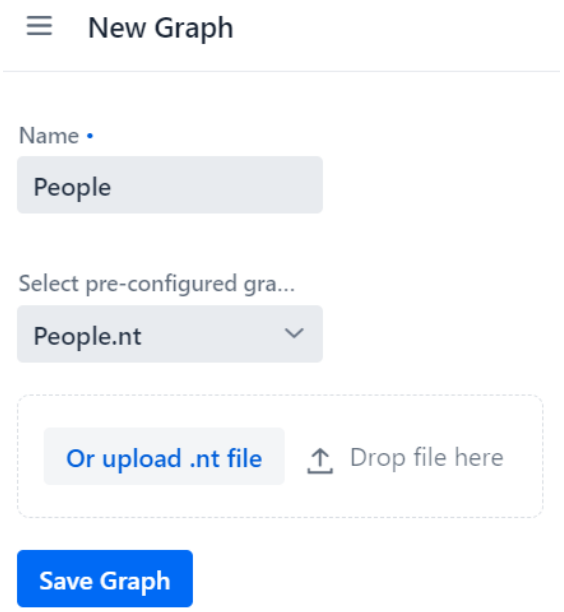

Figure 5.4: A new version of a graph can be uploaded or chosen from a pre-configured list of graphs

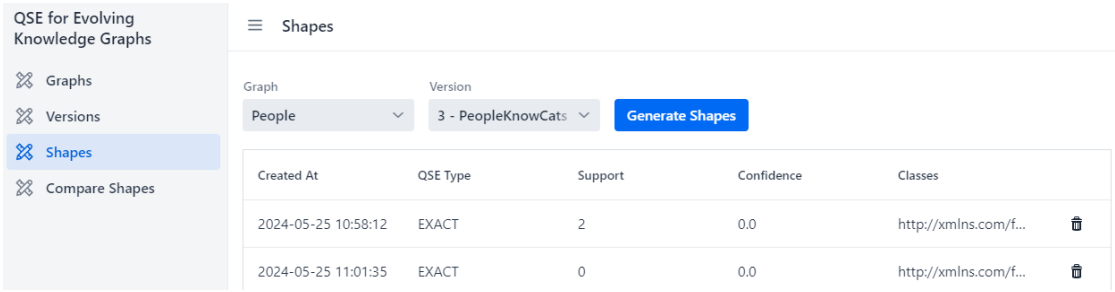

Figure 5.5: Overview of all extracted shapes for a version of a graph

since nothing has changed between the first and the second version.
To showcase the removal of a shape due to low support or confidence values, the detailed comparison of the "colorCatShapeProperty" is chosen. In this example, the SHACL shape no longer exists in the third version of the graph because the "color" attributes of the black and grey cats were deleted. As a result, the support for this property is only one, which is below the defined threshold of two when the shapes were created. This explanation is displayed in Figure 5.9.

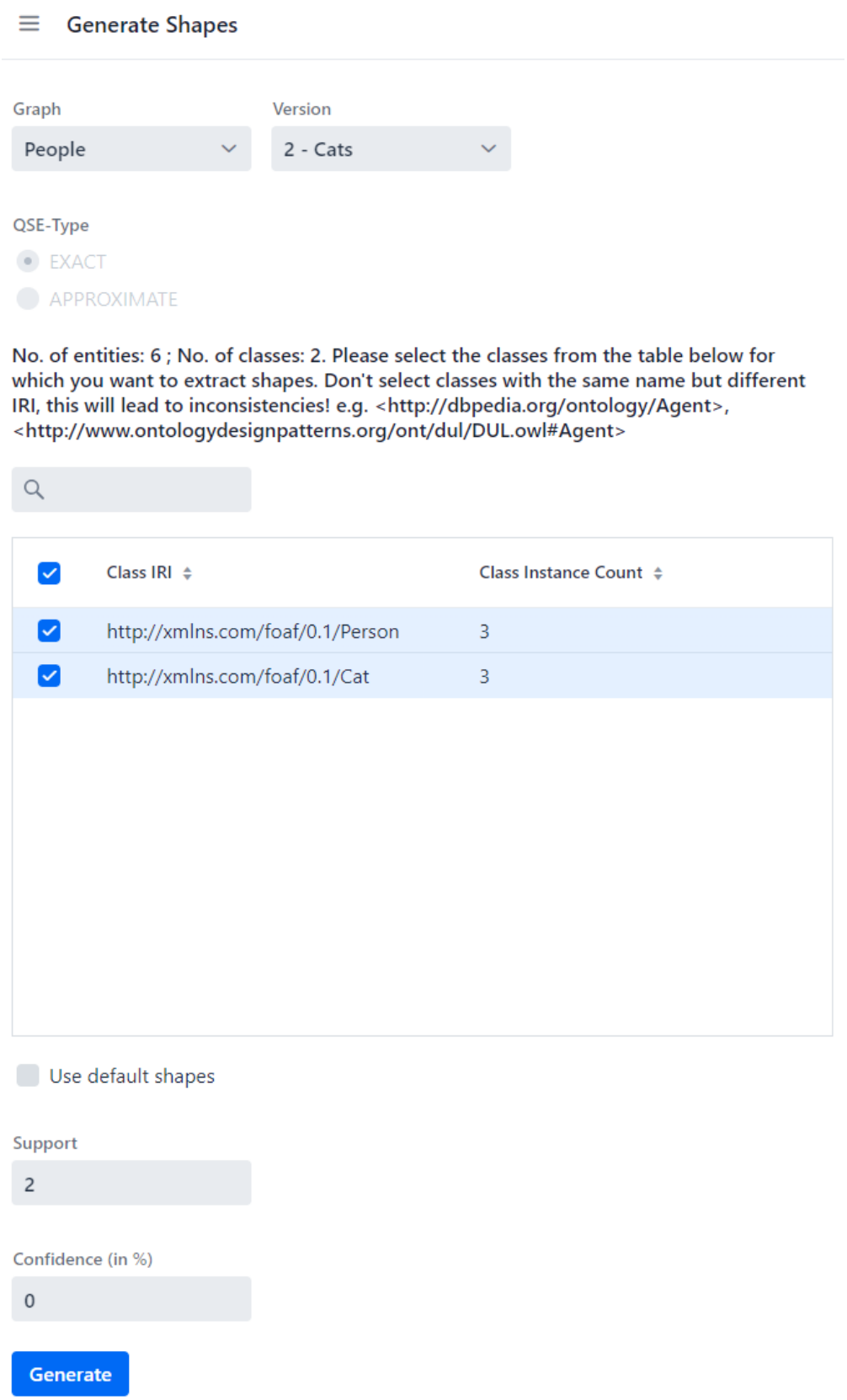

Figure 5.6: SHACL shapes can be extracted for a version of a graph

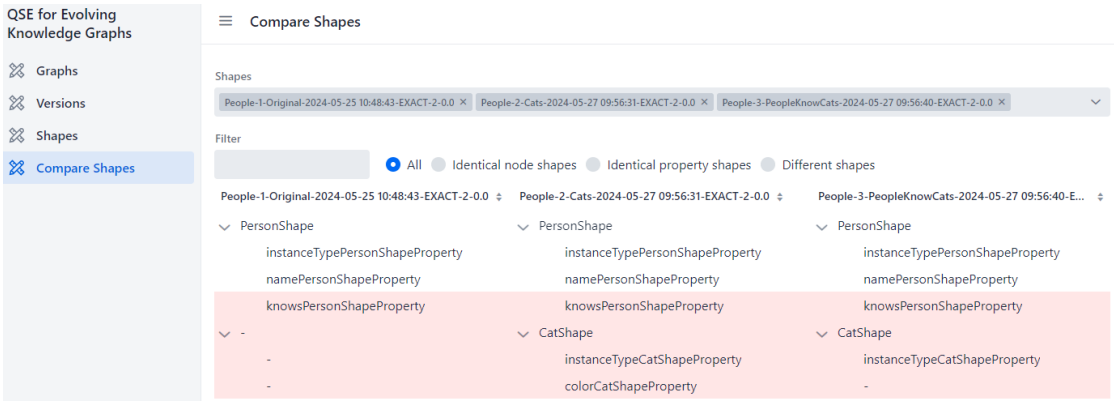

Figure 5.7: Comparison between extracted shapes

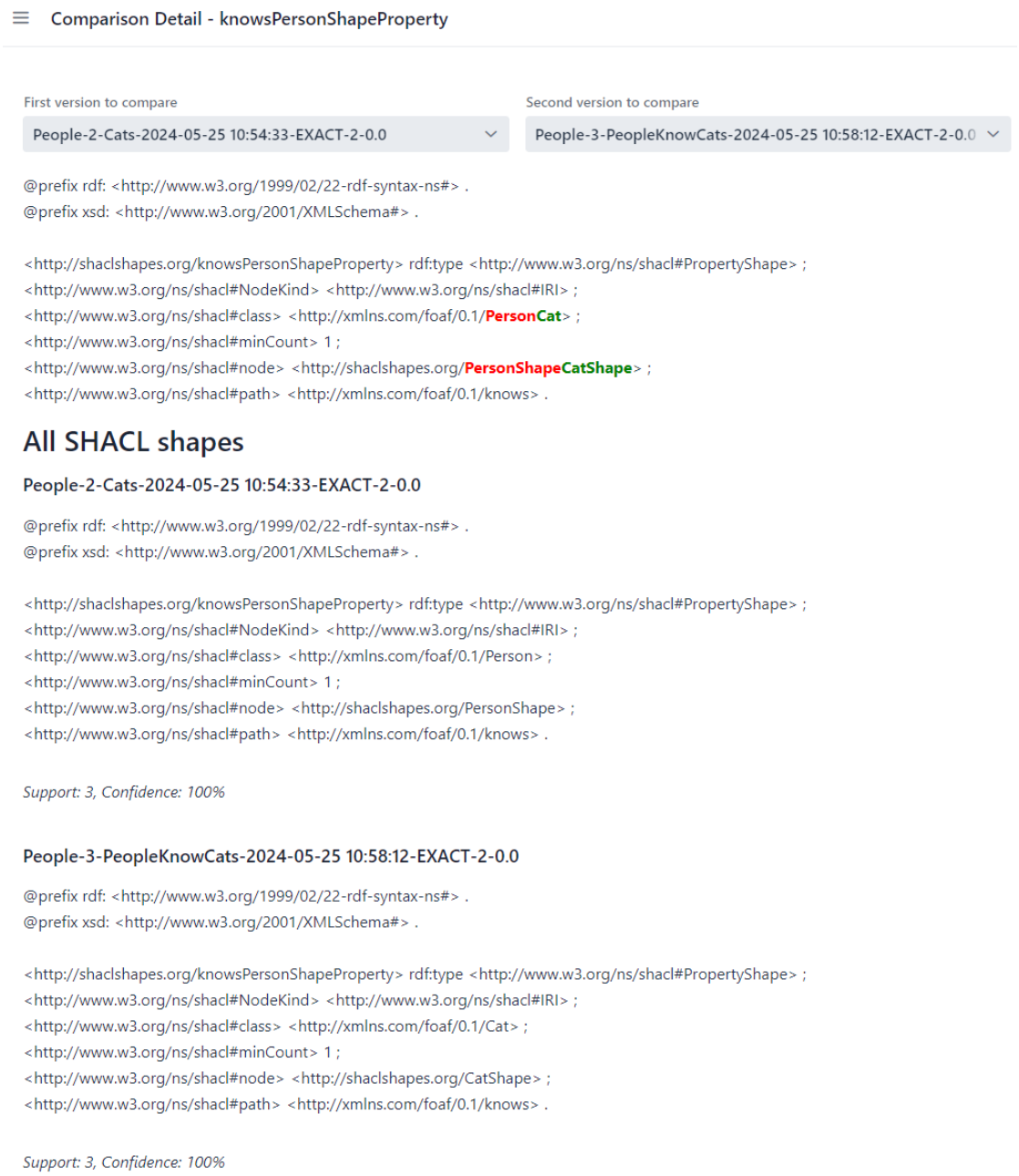

Figure 5.8: Detailed comparison view

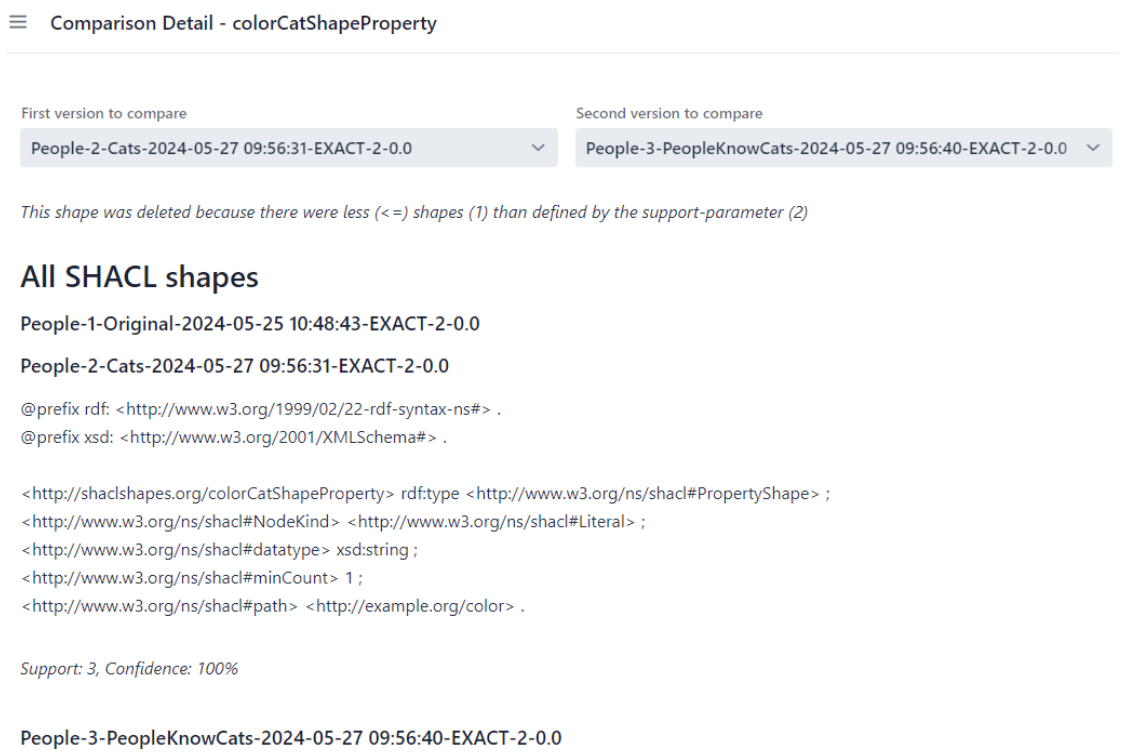

Figure 5.9: Detailed comparison view in case a shape was deleted

Database model

Figure 5.10 shows the database tables which were auto-created by Spring. A graph can have multiple versions and those can have different extracted shapes. For each QSE run (i.e., extracted shapes), classes were defined during the extraction process (specifying which classes of the graph QSE should be run on). Every extracted-shapes object contains multiple node shapes, with default shapes also being stored here. If the user chooses to extract only the default shapes, then no data is stored under default shapes. Each node shape has multiple property shapes. These classes were originally copied from QSE, however, they were duplicated to accommodate customization and other modifications. Every property shape can have ShaclOrList items, as it was previously modeled in QSE.

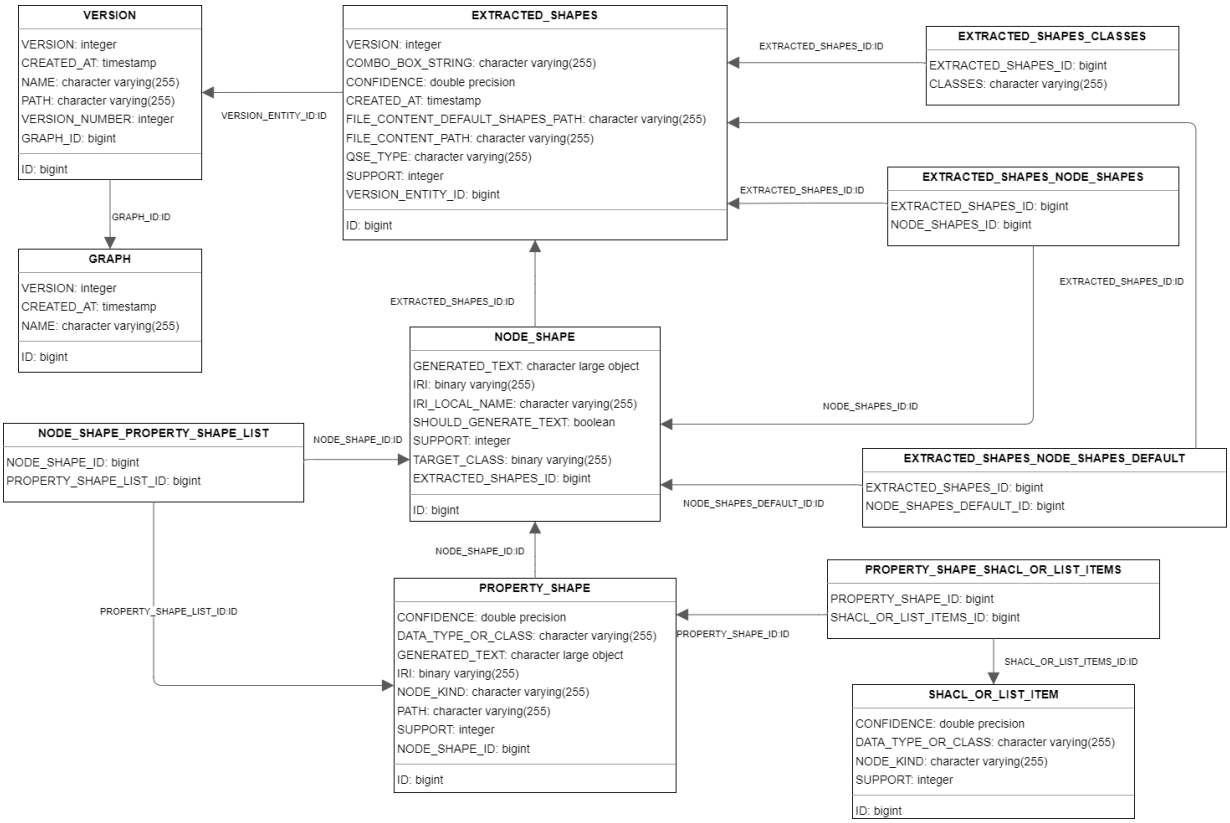

Figure 5.10: ERD of the database model used for the Web app

ERD: Make text size bigger and image smaller in the final version

Performance enhancements

After completing the initial draft of the web app, it was tested with very small datasets to uncover potential edge cases. However, as testing progressed with larger datasets like BEAR-B, it became clear that certain features needed to be adjusted to enhance the speed of the web application.
Originally, it was planned to store the extracted SHACL files directly in the database.

However, this approach quickly led to performance issues while querying data. As a result, a file path is now specified for the location of the SHACL file. The file is stored in a local directory of the project, therefore it cannot be deleted. Similarly for the graph and version files, storing them in the database caused performance issues, so now they are stored as files as well.
A further performance improvement involved saving the generated text for the detail view for each node and property shape, which notably accelerated the comparison process. Additionally, implementing lazy fetching for node shapes, rather than fetching them eagerly, contributed to further performance improvement. Also, the objects for the comparison overview were cached, to prevent reloading all objects when the user returns from the comparison details page.
Another aspect of performance optimization involved determining the most efficient method for extracting text segments for each shape from QSE. There were multiple approaches considered on how to use the information provided by QSE which includes a list of objects and the final SHACL file. However, inconsistencies between these objects were often encountered. In the final version, a combination of the two artifacts was used, the list of objects was used internally and the SHACL file was used for text generation. One attempt involved reading the SHACL file into an RDF4J or Jena model and then filtering it using SPARQL or provided methods to retrieve only the relevant text for each shape. Unexpectedly, this approach included dealing with blank nodes. The generated SHACL shapes included objects like SHACL-or and SHACL-in which are internally stored with blank nodes. To retrieve this full information, an algorithm had to be designed to recursively load the triples in the blank nodes from the model. However, this approach in general proved to be inefficient, therefore the final solution was to use regex to extract the shape segments directly from the SHACL file. Although this approach has several disadvantages, such as potential inconsistencies in recreating SHACL-In items and items in a SHACL-Or List, it is notably faster. However, QSE uses the TurtlePrettyFormatter to format SHACL files, therefore the general, consistent structure of the text is ensured which makes the comparison of shapes easy [83]. In the end, the support and confidence triples were removed by converting the text segment into a Jena file and then filtering the statements. To address the issue of SHACL-Or and SHACL-in items, algorithms were developed to order these text lines alphabetically.
The mapping between the shapes in different QSE runs is based on the name generated by QSE. To compare the text segments of the SHACL shapes, the JavaScript library jsdiff was used to highlight the differences automatically in the details view of the web app [33].

CHAPTER 6

Evaluation

Einleitung, Expert Interview, Experiments

System information

The web application was tested on a Linux Virtual machine with Debian (Version 11) and 8 CPU cores. As RAM allocation varies due to multiple users accessing the server, a specific RAM size cannot be specified.

Test-data

The test data utilized for testing the web application is sourced from BEAR [3] which was designed for testing archiving and querying of evolving semantic web data. These datasets, collectively known as the **BE**nchmark of RDF **AR**chives (BEAR) consist of three real-world datasets. The BEAR data offers valuable statistics and provides various methods for accessing the data, such as having one N-Triples file per version.
For testing the Web application, data from BEAR-B was used. While datasets from BEAR-A (snapshots from Dynamic Linked Data Observator) were too large, with one file reaching 5GB, the data from BEAR-C (Open Data portals) was too homogeneous. For BEAR-B the daily changesets were used since there are already 89 different versions available. The dataset has been compiled from DBPedia Live Changesets over three months (August to October 2015), including the 100 most volatile resources along with their updates.

CHAPTER 7

Discussion

CHAPTER 8

Conclusion & Future Work

APPENDIX A

Systematic Literature Review

Referenzen von ~~aussortierten~~ Quellen trotzdem behalten?

	Title, Reference	Exclusion Reason
	"evolving knowledge graphs"	
1	Evolving Knowledge Graphs [61]	
2	Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs [84]	Title: Reasoning is not in line thesis topic
3	Summarizing Entity Temporal Evolution in Knowledge Graphs [81]	
4	How does knowledge evolve in open knowledge graphs? [72]	
5	Analysing the Evolution of Knowledge Graphs for the Purpose of Change Verification [65]	
6	EvolveKG: a general framework to learn evolving knowledge graphs [60]	
7	KGdiff: Tracking the Evolution of Knowledge Graphs [58]	
8	Predicting the co-evolution of event and Knowledge Graphs [39]	
9	Knowledge Graphs Evolution and Preservation – A Technical Report from ISWS 2019 [18]	
10	Representing Scientific Literature Evolution via Temporal Knowledge Graphs [76]	Title: Scientific Literature Evolution not in line with thesis topic
	"data quality in knowledge graphs"	
11	GraphGuard: Enhancing Data Quality in Knowledge Graph Pipelines [36]	
12	Improving and Assessing Data Quality of Knowledge Graphs [32]	
13	Knowledge Graph Quality Management: A Comprehensive Survey [87]	

Continuation of Table A.1		
	Title, Reference	Exclusion Reason
14	Knowledge Graph Completeness: A Systematic Literature Review [53]	
15	Steps to Knowledge Graphs Quality Assessment [52]	
16	What Are Links in Linked Open Data? A Characterization and Evaluation of Links between Knowledge Graphs on the Web [47]	Title: Links in Linked Open Data not in line with thesis topic
17	Knowledge Graphs 2021: A Data Odyssey [86]	
18	Knowledge graphs [40]	
19	A Practical Framework for Evaluating the Quality of Knowledge Graph [26]	
20	Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL [79]	
"shacl extraction from knowledge graphs"		
21	A Library for Visualizing SHACL over Knowledge Graphs [20]	
22	SCOOP all the Constraints' Flavours for your Knowledge Graph [37]	
23	Learning SHACL shapes from knowledge graphs [8]	
24	Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL [79]	Duplicate
25	Extraction of Validating Shapes from Very Large Knowledge Graphs [74]	
26	Using Knowledge Graph Technologies to Contextualize and Validate Declarations in the Social Security Domain [27]	
27	Automatic extraction of shapes using sheXer [41]	
28	Property assertion constraints for an informed, error-preventing expansion of knowledge graphs [35]	
29	Automatic Construction of SHACL Schemas for RDF Knowledge Graphs Generated by Direct Mappings [28]	Language: Korean, not available with TU-Wien Account
30	Trav-SHACL: Efficiently Validating Networks of SHACL Constraints [43]	
"comparing shacl shapes"		
31	Comparing ShEx and SHACL [44]	Not available with TU-Wien account
32	Using Ontology Design Patterns To Define SHACL Shapes [67]	
33	Semantic rule checking of cross-domain building data in information containers for linked document delivery using the shapes constraint language [46]	

Continuation of Table A.1		
	Title, Reference	Exclusion Reason
34	Astrea: automatic generation of SHACL shapes from ontologies [29]	
35	Trav-SHACL: Efficiently validating networks of SHACL constraints [43]	Duplicate
36	Formalizing Property Constraints in Wikidata [42]	
37	Semantics and Validation of Recursive SHACL [31]	
38	An Argument for Generating SHACL Shapes from ODPs [69]	
39	Comparison of the eye's wave-front aberration measured psychophysically and with the Shack–Hartmann wave-front sensor [77]	Title: not relevant for knowledge graphs
40	Absolute sphericity measurement:? a comparative study of the use of interferometry and a Shack–Hartmann sensor [1]	Title: not relevant for knowledge graphs
"differences between knowledge graphs versions"		
41	Summarizing entity temporal evolution in knowledge graphs [82]	Duplicate
42	KGDiff: Tracking the evolution of knowledge graphs [58]	Duplicate
43	Knowledge Graphs on the Web-An Overview. [48]	
44	Knowledge graphs: A practical review of the research landscape [57]	
45	Bias in Knowledge Graphs–an Empirical Study with Movie Recommendation and Different Language Editions of DBpedia [85]	Title: Bias not relevant for thesis topic
46	Explaining and suggesting relatedness in knowledge graphs [71]	
47	A survey on knowledge graphs: Representation, acquisition, and applications [55]	
48	Measuring accuracy of triples in knowledge graphs [62]	Title: Accuracy not relevant for thesis topic
49	AYNEC: all you need for evaluating completion techniques in knowledge graphs [23]	
50	Modelling dynamics in semantic web knowledge graphs with formal concept analysis [45]	Title: Modelling dynamics not relevant for thesis topic

Table A.1: All initial search results for all search terms

add space after table

	Title, Reference	Comments	Relevance
1	Evolving Knowledge Graphs [61]	Theoretical description of EvolveKG [60] - a framework that reveals cross-time knowledge interaction with desirable performance. This is partly relevant for this thesis since an extra framework is created with a Derivative Graph, allowing knowledge prediction. In this thesis, only snapshots of evolving graphs are considered.	2
3	Summarizing Entity Temporal Evolution in Knowledge Graphs [81]	Envisions an approach where a summary graph is created to catch the evolution of all entities across all versions.	2
4	How does knowledge evolve in open knowledge graphs? [72]	Explains the basics of evolving knowledge graphs.	1
5	Analysing the Evolution of Knowledge Graphs for the Purpose of Change Verification [65]	This paper deals with topological features of a graph to generate classifiers that can judge whether an incoming graph change is correct or incorrect.	2
6	EvolveKG: a general framework to learn evolving knowledge graphs [60]	Already mentioned in [61].	3
7	KGdiff: Tracking the Evolution of Knowledge Graphs [58]	Proposes a software that tracks changes on the schema and the individuals.	2
8	Predicting the co-evolution of event and Knowledge Graphs [39]	The goal of this paper is to predict unobserved facts by using previous temporal information from knowledge graphs and static information. This is not relevant since prediction is not part of this thesis.	3
9	Knowledge Graphs Evolution and Preservation – A Technical Report from ISWS 2019 [18]	Collection of ten papers: Papers 5,6,10 are not relevant. Paper 1 differentiates between different types of evolution and checks if machine learning can capture this evolution. Paper 2 focuses on the characteristics of an evolving knowledge graph, in this case, DBpedia. The third paper discusses changes between two versions of a knowledge graph. Paper 4 is about changes is ontologies and how they can be characterized. Paper 7 deals with the integration of data in knowledge graphs. Versioned knowledge graphs are targeted in Chapter 8. Finally, paper 9 deals with the support of interactive updates in knowledge graphs.	2

Continuation of Table A.2			
	Title, Reference	Comments	Relevance
11	GraphGuard: Enhancing Data Quality in Knowledge Graph Pipelines [36]	Introduces a framework for better data quality in knowledge graph pipelines for humans and machines. This paper is partly relevant to this thesis because the goal of QSE and SHACL is not to check data quality during data ingestion.	2
12	Improving and Assessing Data Quality of Knowledge Graphs [32]	Dissertation which focuses on including data transformations in knowledge graphs that can help to clean the data and complete knowledge graphs by calculating derived data. Furthermore, this paper is about validating knowledge graphs. Validation is done by a reasoning solution called Validatrr. Cleaning, completing and reasoning are not directly addressed by this therefore this paper is not directly relevant.	2
13	Knowledge Graph Quality Management: A Comprehensive Survey [87]	This paper provides a systematic review of quality management in knowledge graphs, also including quality management processes, such as quality assessment, error detection, error correction and completion.	2
14	Knowledge Graph Completeness: A Systematic Literature Review [53]	Different quality dimensions exist - such as accuracy or completeness. This paper summarizes terminologies related to completeness. Therefore it is not relevant for this thesis, since completeness is not a topic.	3
15	Steps to Knowledge Graphs Quality Assessment [52]	The goal of this paper is to extend existing quality assessment frameworks by adding quality dimensions and quality metrics. It is only partly relevant for this thesis since quality dimensions are evaluated with SHACL.	2
17	Knowledge Graphs 2021: A Data Odyssey [86]	This paper discusses advances and lessons learned in the history of knowledge graphs. The abstract is too general that it could be relevant for this thesis.	3
18	Knowledge graphs [40]	Entire book on knowledge graphs that includes a general introduction, how to build and use knowledge graphs and specific use cases.	2
19	A Practical Framework for Evaluating the Quality of Knowledge Graph [26]	In this paper, existing frameworks for quality of knowledge graphs are assessed and a practical framework is proposed which determines if a knowledge graph is fit for purpose.	3

Continuation of Table A.2

	Title, Reference	Comments	Relevance
20	Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL. [79]	The approach of this paper is used in the QSE approach. It proposes a semantic profiling tool that helps to enhance the understanding of the data by using SHACL.	1
21	A Library for Visualizing SHACL over Knowledge Graphs [20]	This master thesis created a library called SHACLViewer which illustrates SHACL shapes in a 3D context.	1
22	SCOOP all the Constraints' Flavours for your Knowledge Graph [37]	The goal of the SCOOP framework is to extract SHACL shapes from already existing knowledge graphs (similar to QSE). However, it mainly uses ontologies and data schemas instead of the entities itself.	2
23	Learning SHACL shapes from knowledge graphs [8]	This paper introduces Inverse Open Path (IOP), a predicate logic formalism that presents specific shapes over connected entities from the knowledge graph. The corresponding learning method is called SHACLearner.	2
25	Extraction of Validating Shapes from Very Large Knowledge Graphs [74]	This thesis builds on this paper, therefore it is excluded.	3
26	Using Knowledge Graph Technologies to Contextualize and Validate Declarations in the Social Security Domain [27]	A master thesis, which generates SHACL shapes for a specific use case, namely forms which describe the work of employees in a quarter. Since SHACL shapes are generated manually, the thesis is not relevant to this thesis.	3
27	Automatic extraction of shapes using sheXer [41]	sheXer produces SHACL shapes similar to QSE using a python library. This paper is also mentioned in the related work of QSE.	2
28	Property assertion constraints for an informed, error-preventing expansion of knowledge graphs [35]	PAC (property assertion constraints) have the goal of checking data before it gets added to the knowledge graph. With this approach, errors can be prevented and the quality of knowledge graphs can be enhanced. PAC works by restricting the range of properties using SPARQL.	2

Continuation of Table A.2			
	Title, Reference	Comments	Relevance
30	Trav-SHACL: Efficiently Validating Networks of SHACL Constraints [43]	Trav-SHACL plans the execution and the traversal of a shapes graph so that invalid entries are detected early. For this task, the shapes graph is reordered.	2
32	Using Ontology Design Patterns To Define SHACL Shapes [67]	Similar to [37], this paper wants to reuse the Ontology Design Pattern (ODP) and contexts to automatically generate SHACL shapes.	2
33	Semantic rule checking of cross-domain building data in information containers for linked document delivery using the shapes constraint language [46]	This case study uses SHACL in the domain of Information Container for Linked Document Delivery (ICDD). Since this is a specific case study, it is not relevant for this thesis.	3
34	Astrea: automatic generation of SHACL shapes from ontologies [29]	Similar to [37] this approach also uses ontologies to automatically generate SHACL shapes. Astrea uses Astrea-KG which provides mappings between ontology constraint patterns and SHACL constraint patterns.	2
36	Formalizing Property Constraints in Wikidata [42]	This paper compares constraints in Wikidata, which uses its own RDF data model for constraints, with constraints that can be created with SPARQL and SHACL. This paper is not relevant, since Wikidata will not be used in this thesis.	3
37	Semantics and Validation of Recursive SHACL [31]	As the title suggests, this paper explores the recursion for SHACL constraints. It proposes concise formal semantics of the core elements of SHACL and validates recursion for these elements. This paper is not relevant since the recursion of SHACL elements is not a topic for this thesis.	3
38	An Argument for Generating SHACL Shapes from ODPs [69]	This book chapter extends the idea of [67].	3
43	Knowledge Graphs on the Web-An Overview. [48]	This book chapter provides an overview and comparison of publicly available knowledge graphs (DBpedia, Wikidata) and gives insights into their sizes and contents. This chapter is not relevant to this thesis since it is too general.	3

Continuation of Table A.2			
	Title, Reference	Comments	Relevance
44	Knowledge graphs: A practical review of the research landscape [57]	The cross-disciplinary nature of knowledge graphs is very important. This paper gives an overview of the major strands in the research landscapes and their different communities. This paper is too general for this thesis.	3
46	Explaining and suggesting relatedness in knowledge graphs [71]	It provides a tool called RECAP, which explains the relatedness of a pair of entities in a knowledge graph. Relatedness is not a topic of this thesis, therefore it is not relevant.	3
47	A survey on knowledge graphs: Representation, acquisition, and applications [55]	This survey covers a broad range of topics regarding knowledge graphs. However, also temporal knowledge graphs are discussed, which makes the paper partly relevant.	2
49	AYNEC: all you need for evaluating completion techniques in knowledge graphs [23]	The tool AYNEC provides a suite for the evaluation of knowledge graph completion techniques. Since knowledge graph completion is not a topic of this thesis, this paper is not relevant.	3

Table A.2: Comments and possible exclusion reasons on all papers after the first round based on the Abstract. Relevance ranges from 1 (absolutely relevant) to 3 (not relevant)

add space after table

APPENDIX B

Demonstration Knowledge Graphs

```
<http://example.org/alice#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/bob#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/jenny#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/alice#me>
    <http://xmlns.com/foaf/0.1/name> "Alice" .
<http://example.org/bob#me>
    <http://xmlns.com/foaf/0.1/name> "Bob" .
<http://example.org/jenny#me>
    <http://xmlns.com/foaf/0.1/name> "Jenny" .
<http://example.org/alice#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/bob#me> .
<http://example.org/bob#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/alice#me> .
<http://example.org/jenny#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/alice#me> .
```

Listing B.1: People Knowledge Graph (Version 1)

```
<http://example.org/alice#me>
```

```
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/bob#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/jenny#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/alice#me>
    <http://xmlns.com/foaf/0.1/name> "Alice" .
<http://example.org/bob#me>
    <http://xmlns.com/foaf/0.1/name> "Bob" .
<http://example.org/jenny#me>
    <http://xmlns.com/foaf/0.1/name> "Jenny" .
<http://example.org/alice#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/bob#me> .
<http://example.org/bob#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/alice#me> .
<http://example.org/jenny#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/alice#me> .
<http://example.org/orangeCat#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Cat> .
<http://example.org/blackCat#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Cat> .
<http://example.org/greyCat#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Cat> .
<http://example.org/orangeCat#me>
    <http://example.org/color> "orange" .
<http://example.org/blackCat#me>
    <http://example.org/color> "black" .
<http://example.org/greyCat#me>
    <http://example.org/color> "grey" .
```

Listing B.2: People Knowledge Graph (Version 2)

```
<http://example.org/alice#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
```

```
<http://example.org/bob#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/jenny#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/alice#me>
    <http://xmlns.com/foaf/0.1/name> "Alice" .
<http://example.org/bob#me>
    <http://xmlns.com/foaf/0.1/name> "Bob" .
<http://example.org/jenny#me>
    <http://xmlns.com/foaf/0.1/name> "Jenny" .
<http://example.org/alice#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/orangeCat#me> .
<http://example.org/bob#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/blackCat#me> .
<http://example.org/jenny#me>
    <http://xmlns.com/foaf/0.1/knows>
    <http://example.org/greyCat#me> .
<http://example.org/orangeCat#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Cat> .
<http://example.org/blackCat#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Cat> .
<http://example.org/greyCat#me>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Cat> .
<http://example.org/orangeCat#me>
    <http://example.org/color> "orange" .
```

Listing B.3: People Knowledge Graph (Version 3)

List of Figures

List of Tables

List of Algorithms

Referenzen überarbeiten mit Feedback von erster Version. Dblp verwenden.

Bibliography

[1] Absolute sphericity measurement:?a comparative study of the use of interferometry and a Shack–Hartmann sensor.

[2] Apache Jena - Home. \url{https://jena.apache.org/}.

[3] BEAR | BEnchmark of RDF ARchives.

[4] Dataset releases Archives.

[5] dkw-aau/qse at shactor.

[6] Free prototyping tool for web & mobile apps - Justinmind.

[7] GraphDB Downloads and Resources.

[8] Learning SHACL shapes from knowledge graphs - IOS Press.

[9] Linked Life Data - A Semantic Data Integration Platform for the Biomedical Domain.

[10] OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition).

[11] RDF Delta Use Cases.

[12] Shapes Constraint Language (SHACL).

[13] ShEx - Shape Expressions.

[14] SPARQL 1.1 Overview.

[15] TopQuadrant | Enterprise Models for Data Governance.

[16] Vaadin Docs.

[17] Wikidata.

[18] Abbas, N., Alghamdi, K., Alinam, M., Alloatti, F., Amaral, G., d'Amato, C., Asprino, L., Beno, M., Bensmann, F., Biswas, R., Cai, L., Capshaw, R., Carriero, V. A., Celino, I., Dadoun, A., De Giorgis, S., Delva, H., Domingue, J., Dumontier, M., Emonet, V., van Erp, M., Arias, P. E., Fallatah, O., Ferrada, S., Ocaña, M. G., Georgiou, M., Gesese, G. A., Gillis-Webber, F., Giovannetti, F., Buey, M. G., Harrando, I., Heibi, I., Horta, V., Huber, L., Igne, F., Jaradeh, M. Y., Keshan, N., Koleva, A., Koteich, B., Kurniawan, K., Liu, M., Ma, C., Maas, L., Mansfield, M., Mariani, F., Marzi, E., Mesbah, S., Mistry, M., Tirado, A. C. M., Nguyen, A., Nguyen, V. B., Oelen, A., Pasqual, V., Paulheim, H., Polleres, A., Porena, M., Portisch, J., Presutti, V., Pustu-Iren, K., Mendez, A. R., Roshankish, S., Rudolph, S., Sack, H., Sakor, A., Salas, J., Schleider, T., Shi, M., Spinaci, G., Sun, C., Tietz, T., Dhouib, M. T., Umbrico, A., Berg, W. v. d., and Xu, W. Knowledge Graphs Evolution and Preservation – A Technical Report from ISWS 2019, Dec. 2020. arXiv:2012.11936 [cs].

[19] Adams, W. Conducting Semi-Structured Interviews. Aug. 2015.

[20] Alom, H. A Library for Visualizing SHACL over Knowledge Graphs. Master's thesis, Hannover : Gottfried Wilhelm Leibniz Universität Hannover, Mar. 2022.

[21] Antony, J. 2 - Fundamentals of Design of Experiments. In *Design of Experiments for Engineers and Scientists (Second Edition)*, J. Antony, Ed., second edition ed. Elsevier, Oxford, 2014, pp. 7–17.

[22] Antony, J. 4 - A Systematic Methodology for Design of Experiments. In *Design of Experiments for Engineers and Scientists (Second Edition)*, J. Antony, Ed., second edition ed. Elsevier, Oxford, 2014, pp. 33–50.

[23] Ayala, D., Borrego, A., Hernández, I., Rivero, C. R., and Ruiz, D. AYNEC: All You Need for Evaluating Completion Techniques in Knowledge Graphs. In *The Semantic Web* (Cham, 2019), P. Hitzler, M. Fernández, K. Janowicz, A. Zaveri, A. J. Gray, V. Lopez, A. Haller, and K. Hammar, Eds., Springer International Publishing, pp. 397–411.

[24] Boneva, I., Dusart, J., Fernández Alvarez, D., and Gayo, J. E. L. Shape Designer for ShEx and SHACL Constraints, Oct. 2019. Published: ISWC 2019 - 18th International Semantic Web Conference.

[25] Camburn, B., Viswanathan, V., Linsey, J., Anderson, D., Jensen, D., Crawford, R., Otto, K., and Wood, K. Design prototyping methods: state of the art in strategies, techniques, and guidelines. *Design Science 3* (2017), e13.

[26] Chen, H., Cao, G., Chen, J., and Ding, J. A Practical Framework for Evaluating the Quality of Knowledge Graph. In *Knowledge Graph and Semantic Computing: Knowledge Computing and Language Understanding* (Singapore, 2019), X. Zhu, B. Qin, X. Zhu, M. Liu, and L. Qian, Eds., Springer, pp. 111–122.

[27] Chiem Dao, D., and Université de Liège > Master ingé. civ. info., F. Using Knowledge Graph Technologies to Contextualize and Validate Declarations in the Social Security Domain. Accepted: 2023-07-12T02:11:50Z Publisher: Université de Liège, Liège, Belgique Section: Université de Liège.

[28] Choi, J.-W. Automatic Construction of SHACL Schemas for RDF Knowledge Graphs Generated by Direct Mappings. *25*, 10 (2020), 23–34.

[29] Cimmino, A., Fernández-Izquierdo, A., and García-Castro, R. Astrea: Automatic Generation of SHACL Shapes from Ontologies. In *The Semantic Web* (Cham, 2020), A. Harth, S. Kirrane, A.-C. Ngonga Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase, and M. Cochez, Eds., Springer International Publishing, pp. 497–513.

[30] Cimmino, A., Fernández-Izquierdo, A., and García-Castro, R. Astrea: Automatic Generation of SHACL Shapes from Ontologies. In *The Semantic Web*, A. Harth, S. Kirrane, A.-C. Ngonga Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase, and M. Cochez, Eds., vol. 12123. Springer International Publishing, Cham, 2020, pp. 497–513. Series Title: Lecture Notes in Computer Science.

[31] Corman, J., Reutter, J. L., and Savković, O. Semantics and Validation of Recursive SHACL. In *The Semantic Web – ISWC 2018* (Cham, 2018), D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, and E. Simperl, Eds., Springer International Publishing, pp. 318–336.

[32] De Meester, B. *Improving and assessing data quality of knowledge graphs.* PhD Thesis, Ghent University, 2020.

[33] Decker, K. kpdecker/jsdiff, Apr. 2024. original-date: 2011-03-29T16:25:28Z.

[34] developers, E. R. Welcome · Eclipse RDF4J™ | The Eclipse Foundation.

[35] Dibowski, H. *Property Assertion Constraints for an Informed, Error-Preventing Expansion of Knowledge Graphs.* Nov. 2021. Pages: 248.

[36] Dorsch, R., Freund, M., Fries, J., and Harth, A. GraphGuard: Enhancing Data Quality in Knowledge Graph Pipelines.

[37] Duan, X. dtai-kg/SCOOP: v1.0.0, Dec. 2023.

[38] Durakovic, B. Design of experiments application, concepts, examples: State of the art. *Periodicals of Engineering and Natural Sciences 5* (Dec. 2017), 421–439.

[39] Esteban, C., Tresp, V., Yang, Y., Baier, S., and Krompaß, D. Predicting the co-evolution of event and Knowledge Graphs. In *2016 19th International Conference on Information Fusion (FUSION)* (July 2016), pp. 98–105.

[40] Fensel, D., Simsek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., Toma, I., Umbrich, J., and Wahler, A. *Knowledge graphs.* Springer, 2020.

[41] Fernandez-Álvarez, D., Labra-Gayo, J. E., and Gayo-Avello, D. Automatic extraction of shapes using sheXer. *Knowledge-Based Systems 238* (Feb. 2022), 107975.

[42] Ferranti, N., Polleres, A., de Souza, J. F., and Ahmetaj, S. Formalizing Property Constraints in Wikidata.

[43] Figuera, M., Rohde, P. D., and Vidal, M.-E. Trav-SHACL: Efficiently Validating Networks of SHACL Constraints. In *Proceedings of the Web Conference 2021* (New York, NY, USA, June 2021), WWW '21, Association for Computing Machinery, pp. 3337–3348.

[44] Gayo, J. E. L., Prud'hommeaux, E., Boneva, I., and Kontokostas, D. Comparing ShEx and SHACL. In *Validating RDF Data*, J. E. L. Gayo, E. Prud'hommeaux, I. Boneva, and D. Kontokostas, Eds. Springer International Publishing, Cham, 2018, pp. 233–266.

[45] González, L., and Hogan, A. Modelling Dynamics in Semantic Web Knowledge Graphs with Formal Concept Analysis. In *Proceedings of the 2018 World Wide Web Conference* (Republic and Canton of Geneva, CHE, Apr. 2018), WWW '18, International World Wide Web Conferences Steering Committee, pp. 1175–1184.

[46] Hagedorn, P., Pauwels, P., and König, M. Semantic rule checking of cross-domain building data in information containers for linked document delivery using the shapes constraint language. *Automation in Construction 156* (Dec. 2023), 105106.

[47] Haller, A., Fernández, J. D., Kamdar, M. R., and Polleres, A. What Are Links in Linked Open Data? A Characterization and Evaluation of Links between Knowledge Graphs on the Web. *Journal of Data and Information Quality 12*, 2 (May 2020), 9:1–9:34.

[48] Heist, N., Hertling, S., Ringler, D., and Paulheim, H. Knowledge Graphs on the Web-An Overview. *Knowledge Graphs for eXplainable Artificial Intelligence* (2020), 3–22.

[49] Hevner, A. A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems 19* (Jan. 2007).

[50] Hevner, A., R, A., March, S., T, S., Park, Park, J., Ram, and Sudha. Design Science in Information Systems Research. *Management Information Systems Quarterly 28* (Mar. 2004), 75–.

[51] Hose, K., and Sallinger, E. Introduction to Semantic Systems, Nov. 2024.

[52] Huaman, E. Steps to Knowledge Graphs Quality Assessment, Aug. 2022.

[53] Issa, S., Adekunle, O., Hamdi, F., Cherfi, S. S.-S., Dumontier, M., and Zaveri, A. Knowledge Graph Completeness: A Systematic Literature Review. *IEEE Access 9* (2021), 31322–31339. Conference Name: IEEE Access.

[54] Jesús, M. Scientific Prototyping: A Novel Approach to Conduct Research and Engineer Products. pp. 32–35.

[55] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Transactions on Neural Networks and Learning Systems 33*, 2 (Feb. 2022), 494–514. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[56] Keely, A. shaclgen: Shacl graph generator.

[57] Kejriwal, M. Knowledge Graphs: A Practical Review of the Research Landscape. *Information 13*, 4 (Apr. 2022), 161. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[58] Keshavarzi, A., and Kochut, K. J. KGdiff: Tracking the Evolution of Knowledge Graphs. In *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)* (Aug. 2020), pp. 279–286.

[59] Kitchenham, B., and Charters, S. Guidelines for performing Systematic Literature Reviews in Software Engineering.

[60] Liu, J., Yu, Z., Guo, B., Deng, C., Fu, L., Wang, X., and Zhou, C. EvolveKG: a general framework to learn evolving knowledge graphs. *Frontiers of Computer Science 18*, 3 (Jan. 2024), 183309.

[61] Liu, J., Zhang, Q., Fu, L., Wang, X., and Lu, S. Evolving Knowledge Graphs. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications* (Paris, France, Apr. 2019), IEEE, pp. 2260–2268.

[62] Liu, S., d'Aquin, M., and Motta, E. Measuring Accuracy of Triples in Knowledge Graphs. In *Language, Data, and Knowledge* (Cham, 2017), J. Gracia, F. Bond, J. P. McCrae, P. Buitelaar, C. Chiarcos, and S. Hellmann, Eds., Springer International Publishing, pp. 343–357.

[63] Mayring, P. Qualitative Content Analysis. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research [On-line Journal], http://qualitative-research.net/fqs/fqs-e/2-00inhalt-e.htm 1* (June 2000).

[64] Mihindukulasooriya, N., Rashid, M. R. A., Rizzo, G., García-Castro, R., Corcho, O., and Torchiano, M. RDF shape induction using knowledge base profiling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing* (Pau France, Apr. 2018), ACM, pp. 1952–1959.

[65] Nishioka, C., and Scherp, A. Analysing the Evolution of Knowledge Graphs for the Purpose of Change Verification. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)* (Jan. 2018), pp. 25–32.

[66] Omran, P. G., Taylor, K., Mendez, S. R., and Haller, A. Towards SHACL Learning from Knowledge Graphs.

[67] Pandit, H. J., O'Sullivan, D., and Lewis, D. Using Ontology Design Patterns To Define SHACL Shapes.

[68] Pandit, H. J., O'Sullivan, D., and Lewis, D. Using Ontology Design Patterns To Define SHACL Shapes.

[69] Pandit, H. J., O'Sullivan, D., and Lewis, D. An Argument for Generating SHACL Shapes from ODPs. In *Advances in Pattern-Based Ontology Engineering*. IOS Press, 2021, pp. 134–141.

[70] Peffers, K., Rothenberger, M., Tuunanen, T., and Vaezi, R. Design Science Research Evaluation. vol. 7286, pp. 398–410.

[71] Pirrò, G. Explaining and Suggesting Relatedness in Knowledge Graphs. In *The Semantic Web - ISWC 2015* (Cham, 2015), M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, K. Thirunarayan, and S. Staab, Eds., Springer International Publishing, pp. 622–639.

[72] Polleres, A., Pernisch, R., Bonifati, A., Dell'Aglio, D., Dobriy, D., Dumbrava, S., Etcheverry, L., Ferranti, N., Hose, K., Jiménez-Ruiz, E., Lissandrini, M., Scherp, A., Tommasini, R., and Wachs, J. How does knowledge evolve in open knowledge graphs? *Transactions on Graph Data and Knowledge 1*, 1 (Dec. 2023), 11:1–11:59. Publisher: Dagstuhl.

[73] PubChem. About.

[74] Rabbani, K., Lissandrini, M., and Hose, K. Extraction of Validating Shapes from Very Large Knowledge Graphs. *Proceedings of the VLDB Endowment 16*, 5 (Jan. 2023), 1023–1032.

[75] Rabbani, K., Lissandrini, M., and Hose, K. SHACTOR: Improving the Quality of Large-Scale Knowledge Graphs with Validating Shapes. In *Companion of the 2023 International Conference on Management of Data* (New York, NY, USA, 2023), SIGMOD '23, Association for Computing Machinery, pp. 151–154. event-place: Seattle, WA, USA.

[76] Rossanez, A., Reis, J., and Torres, R. D. S. Representing Scientific Literature Evolution via Temporal Knowledge Graphs. *CEUR Workshop Proceedings* (2020). Accepted: 2021-09-06T11:20:56Z Publisher: CEUR Workshop Proceedings.

[77] Salmon, T. O., Thibos, L. N., and Bradley, A. Comparison of the eye's wave-front aberration measured psychophysically and with the Shack–Hartmann wave-front sensor. *JOSA A 15*, 9 (Sept. 1998), 2457–2465. Publisher: Optica Publishing Group.

[78] Sanders, P. Algorithm Engineering – An Attempt at a Definition. In *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, S. Albers, H. Alt, and S. Näher, Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 321–340.

[79] Spahiu, B., Maurino, A., and Palmonari, M. Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL.

[80] Spahiu, B., Maurino, A., and Palmonari, M. Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL. In *WOP@ISWC* (2018).

[81] Tasnim, M., Collarana, D., Graux, D., Orlandi, F., and Vidal, M.-E. Summarizing Entity Temporal Evolution in Knowledge Graphs. In *Companion Proceedings of The 2019 World Wide Web Conference* (San Francisco USA, May 2019), ACM, pp. 961–965.

[82] Tasnim, M., Collarana, D., Graux, D., Orlandi, F., and Vidal, M.-E. Summarizing Entity Temporal Evolution in Knowledge Graphs. In *Companion Proceedings of The 2019 World Wide Web Conference* (New York, NY, USA, May 2019), WWW '19, Association for Computing Machinery, pp. 961–965.

[83] Textor, A. atextor/turtle-formatter, Oct. 2023. original-date: 2021-02-03T05:07:11Z.

[84] Trivedi, R., Dai, H., Wang, Y., and Song, L. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In *Proceedings of the 34th International Conference on Machine Learning* (July 2017), PMLR, pp. 3462–3471. ISSN: 2640-3498.

[85] Voit, M. M., and Paulheim, H. Bias in Knowledge Graphs – an Empirical Study with Movie Recommendation and Different Language Editions of DBpedia, May 2021. arXiv:2105.00674 [cs].

[86] Weikum, G. Knowledge graphs 2021: a data odyssey. *Proceedings of the VLDB Endowment 14*, 12 (July 2021), 3233–3238.

[87] Xue, B., and Zou, L. Knowledge Graph Quality Management: A Comprehensive Survey. *IEEE Transactions on Knowledge and Data Engineering 35*, 5 (May 2023), 4969–4988. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Referenzen überarbeiten mit Feedback von erster Version. Dblp verwenden.