Facilitating RDF Querying in Research Data Management through ShEx-Based SPARQL Query Construction

Christoph Göpfert[1,*], Sheeba Samuel[1] and Martin Gaedke[1]

[1]Chemnitz University of Technology, Germany

Abstract

Metadata about research data is increasingly represented using the Resource Description Framework (RDF) across diverse domains, as RDF offers a structured yet flexible way to represent and query data. However, querying RDF data can be challenging, as users need to be proficient in the SPARQL query language to create syntactically correct queries. Furthermore, users must be familiar with the ontologies used for describing the desired data to choose appropriate classes and properties for their queries. As a result, the manual query construction process is both time-consuming and error-prone. To address this, we propose an approach that leverages schemas in the Shape Expressions Language (ShEx) as blueprints for constructing SPARQL queries. This approach is demonstrated using two case studies in the research data domain — one focusing on image data and one focusing on survey data — where each data type is described in a respective ShEx shape. We then use our ShEx2SPARQL tool to automatically translate a ShEx schema into a SPARQL query, thereby relieving users of the burden of manual query construction and enhancing the accessibility of RDF data.

Keywords

Research Data Management, Shape Expressions, SPARQL, Linked Data, RDF

1. Introduction

Research data management encompasses the handling of a large diversity of data types, such as tabular data, survey results, videos, or images [1]. The FAIR principles [2] call for the use of rich and relevant metadata, In particular, the FAIR principles *F2* and *R1* require that 1) *"Data are described with rich metadata"* and 2) *"(Meta)data are richly described with a plurality of accurate and relevant attributes"*. Since research data types differ in their attributes, either a highly flexible schema or several data type-specific schemas are required to describe these attributes in detail. In the context of Linked Data, data type-specific schemas can be realized using ontologies [3], which provide formal, structured descriptions of classes and their relationships. The triple-based, schemaless data model of the Resource Description Framework (RDF) [4] serves as a basis for this, providing the means to describe complex data structures.

Accessing RDF data is typically realized using the SPARQL query language [5]. However, the use of type-specific ontologies also leads to an increase in the complexity of data accessibility. As more granular and specialized schemas are used, searching for specific research data via an SPARQL endpoint becomes increasingly challenging, especially for users who are not well-versed in semantic web technologies.

Consider the scenario of Alice, a researcher who is looking for detailed information about survey data stored in a research data repository. The repository provides an SPARQL endpoint to retrieve the data Alice is searching for. However, Alice faces two major challenges hindering her from finding the data. First, she must investigate which ontologies the repository uses, as she needs to identify all relevant classes and properties for her query. Second, Alice is new to RDF and SPARQL, which makes the task of manually constructing an effective query both time-consuming and labor-intensive. In this scenario, predefined templates to query survey data would mitigate the challenges faced by Alice. Different data types can be expressed in formal data shapes that describe the expected structure of RDF data. These

2nd International Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2025), co-located with ESWC 2025, June 01–02, 2025, Portorož, Slovenia

*Corresponding author.

christoph.goepfert@informatik.tu-chemnitz.de (C. Göpfert); sheeba.samuel@informatik.tu-chemnitz.de (S. Samuel); martin.gaedke@informatik.tu-chemnitz.de (M. Gaedke)

0000-0001-6659-8947 (C. Göpfert); 0000-0002-7981-8504 (S. Samuel); 0000-0002-6729-2912 (M. Gaedke)

description can be realized by defining constraints on classes and their relationships, e.g. by using the Shape Expressions Language (ShEx) [6].

In this paper, we leverage shape definitions to automatically construct SPARQL queries conforming to the specified shape of a research data type. For this purpose, we use our ShEx2SPARQL tool to demonstrate query construction for both image and survey data which are described using the domain-specific ontologies: *Lightweight Image Ontology* [7, 8] and *The Survey Ontology* [9, 10].

The remainder of this paper is structured as follows: Section 2 offers a review of related work. Section 3 contains technical information on the Shape Expressions Language and ShEx2SPARQL. In Section 4, we demonstrate in two case studies how shape definitions for survey and image data can be used to automatically construct SPARQL queries. Section 5 provides a discussion of the results for the two case studies. Finally, Section 6 offers a conclusion.

2. Related Work

There exists a multitude of approaches in the domain of SPARQL query construction. Mussa et al. [11] and Kuric et al. [12] divide them into the groups form-based, graph-based and natural language-based query building approaches. A further group of approaches can be found in some RDF-based question answering (QA) approaches, which tend to employ template-based methods for constructing SPARQL queries.

Among form-based query building approaches, a prominent example is the Wikidata Query Service (WDQS) [13]. WDQS provides a query editor that allows users to filter by entities suggested through the editor. If a suggested entity is selected, a corresponding SPARQL query to retrieve data for this entity is generated. A downside of this approach is that it is tailored exclusively to Wikidata entities. Similarly, VizQuery[1] offers a user interface consisting of a form that enables users to specify rules for their query. However, the flexibility of these rules is severely constrained, as users are required to select a concrete predicate and object for each rule from a list of suggestions. Linked Data Query Wizard (LDQW) [14] is an approach that focuses on the visualization of query outcomes. LDQW makes use of a tabular-based visualization approach for presenting query results. Before the data can be displayed, it is required that the user issues an initial full-text search over the entire dataset or to select a pre-defined RDF Data Cube[2] dataset. The results for this query are then displayed in a table, providing users with functions familiar from spreadsheet applications.

Graph-based query builders offer a more intuitive approach to query construction by enabling the interaction with RDF data through visual representation as a graph. QueryVOWL [15] leverages the visual representation of VOWL [16] to map graphical elements, such as classes, literals, or properties, to corresponding SPARQL fragments. Users can interactively select an element within the graph to retrieve relevant data for the selected node. A limitation of the approach is that it is not supporting logical combinations of filters, thereby limiting its ability to express complex queries. Vargas et al. [17] present another graph-based approach named RDF Explorer. For their approach, they propose a visual query language. Users can create a query visually by choosing suggestions from a search pane or by directly adding nodes and linking them with edges. Visual elements are expressed in their visual query language, which is finally translated to SPARQL. Although the visual query language simplifies visual query construction, it is restricted and not capable of expressing query operators such as UNION or OPTIONAL.

Sparklis [18] offers a visual query builder based on natural language and faceted views to simplify query construction. Sparklis guides users by suggesting available entities, classes, properties, filters and modifiers to refine their queries interactively. However, its reliance on predefined query patterns also restricts its flexibility. The authors state that Sparklis covers a *"large subset of SPARQL 1.1"*, it does however not support CONSTRUCT queries.

[1] https://hay.toolforge.org/vizquery/
[2] https://www.w3.org/TR/vocab-data-cube/

Cocco et al. [19] use a template-based approach for their QA system which links natural language questions to SPARQL queries. A *tagger* component is used to identify entities and relationships contained in a new question. The result returned from the *tagger* are then used to construct a SPARQL query by selecting the most suitable template based on similarity. The templates contain placeholders that are filled with the identified entities and relationships. Chen et al. [20] also use a template-based approach for the Light-QAWizard approach. Their approach makes use of a recurrent neural network and multi-label classification, mapping questions to templates. The classifier is used to select the most suitable template and attempts to create only the SPARQL clauses that are required to answer the question to increase query performance by avoiding any redundant query patterns. Many recent approaches leverage Large Language Models (LLM) in QA systems. Taffa and Usbeck [21] use an LLM to generate SPARQL queries in the context of scholarly knowledge graphs. Their approach relies on a training set of test questions, from which the top-n most similar questions are retrieved for a new question. Similarity is calculated based on the question's embedding score using a BERT-based sentence encoder. The SPARQL queries associated with the top-n most similar questions are used in the prompt to the LLM and serve as guidance for creating a SPARQL query. A disadvantage of the presented template-based approaches is that they require an initial phase in which the templates used as guidance or reference data must either be compiled or manually created.

3. Background

The Shape Expressions Language (ShEx) [6] is a language that enables the formal description of structures in RDF data. A ShEx schema describes one or multiple shapes, where each shape essentially consists of a set of rules that specify expected patterns, properties or constraints within an RDF graph. Returning to the initial motivation scenario described in Section 1 in which Alice is looking for detailed metadata about surveys, a survey shape can be defined to specify that entities of the type *Survey* are related to entities of the type *Question*, which in turn are related to entities of the type *Answer*. The shape may further require that literal values must be provided for both *Question* and *Answer* entities, which hold the respective question and answer texts. ShEx supports various constructs, such as constraints on the subjects and objects of a triple (*Node Constraints*) and constraints on triple structures (*Triple Constraints*), as well as grouping and combinations of constraints.

Traditionally, the intended use for ShEx schemas has been data validation [22], ensuring that data conforms to specified shape structures and maintaining consistency within a graph. However, as shape expressions provide a precise description of expected graph structures, this information also offers a blueprint for SPARQL query patterns. Our tool ShEx2SPARQL[3] is able to leverage these blueprints to automatically construct SPARQL queries. ShEx2SPARQL processes a given ShEx schema and translates it into a corresponding SPARQL query that reflects the constraints of a designated start shape in the schema. The tool parses each element of the schema, including shape expressions, node constraints and triple constraints, and maps them onto SPARQL query patterns conforming to the defined constraints. Consequently, the data retrieved through the constructed query inherently conforms to the underlying ontologies, making use of the same classes and properties that were used in the shape definition. This automated construction procedure addresses the challenges faced by Alice in the initial scenario, who otherwise had to construct the query manually and also had to be familiar with SPARQL and the structure of the underlying ontologies in the knowledge graph.

Despite these benefits, the current implementation of the ShEx2SPARQL tool is subject to limitations. Due to the limited support for recursion provided in the query language, recursive shape references are not and can not be supported. Specifically, it is not possible to express constrained traversal with SPARQL; for instance, it is not possible to enforce for a resource nested at an arbitrary depth to match a specific shape. Currently, the tool offers limited support for cardinality constraints (i.e., optionality and exclusivity) and does not support importing schemas.

[3] https://purl.org/shex2sparql/code

With the technical foundations established, we now turn to the practical application of ShEx2SPARQL for specific types of research data. In the following section, we explain how ShEx schemas together with ShEx2SPARQL can be utilized to create SPARQL queries for survey and image data.

4. Case Studies

In this section, we use two case studies to show how ShEx2SPARQL can be used to construct SPARQL queries from predefined ShEx schemas. The case studies focus on image and survey data. To this end, we selected a domain-specific ontology relevant to each of the respective data types: the *Lightweight Image Ontology* [7] for image data, and *The Survey Ontology* [9] for survey data. Based on these ontologies, we derive ShEx schemas comprising shapes that express the respective research data type and its properties. These ShEx schemas are then used to construct SPARQL queries with our ShEx2SPARQL tool.

4.1. Case Study 1: Image Data

Images are a common type of research data used in various research domains. Effective research data management aims for obtaining rich metadata to not only describe the image itself, but also contextual, descriptive properties. A structured RDF-based representation of image metadata is offered by the *Lightweight Image Ontology* (LIO) [7].

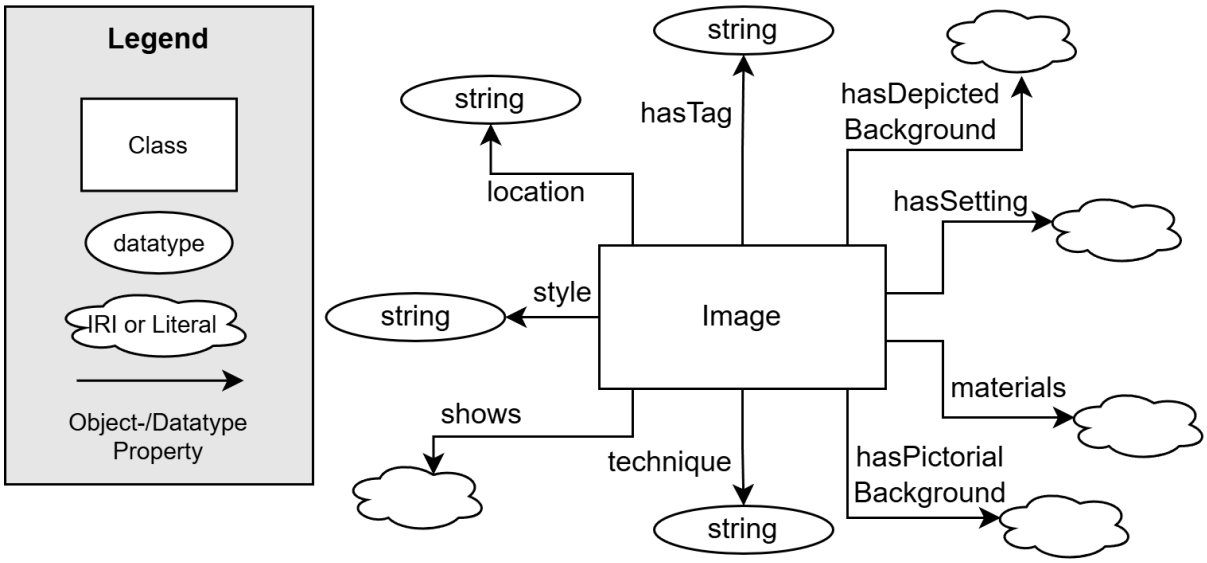

Figure 1: Classes and Properties of the *Lightweight Image Ontology* (Excerpt).

We define an *Image* shape which is limited to classes and properties of the LIO ontology that are needed for describing the *Image* shape in ShEx. Specifically, this comprises the class *Image* which is described by several properties describing its contextual and visual elements. Figure 1 depicts the image class and associated properties. The *hasDepictedBackground* property specifies background elements in the depicted scene, while *hasPictorialBackground* refers to the surface background of the image itself. The *hasSetting* property provides contextual information about where or under what circumstances an image was created. The *location* property offers more specific spatial metadata about where an image was taken. The *shows* property refers to entities that are visually represented in an image, while *hasTag* relates an image to a keyword. The properties *materials*, *style* and *technique* are used to relate to the physical material used to create an image, as well as the used artistic style and technique.

Next, we formalize the classes and properties from the illustrated excerpt of the LIO ontology in a shape named *Image* using the ShEx language. The schema defining the *Image* shape is shown in Fig. 2. First, the schema defines the shape *Image* to be the start shape of the schema. On lines 3–14, the shape

definition specifies the expected properties for the *Image* shape. The shape consists of the four datatype properties *hasTag*, *location*, *style*, and *technique*. Node Constraints restrict the literal values for these properties to the datatype *xsd:string*. The shape describes further properties on lines 8–12 without any Node Constraints on objects of the triples. All properties described on lines 4–12 are optional which is indicated by the *'?'* quantifier. Finally, line 13 enforces that instances of this shape must be typed to the class *:Image*.

```
00 PREFIX : <https://w3id.org/lio/v1#>
01 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
02 start=@<Image>
03 <Image> {
04   :hasTag xsd:string? ;
05   :location xsd:string? ;
06   :style xsd:string? ;
07   :technique xsd:string? ;
08   :hasDepictedBackground .? ;
09   :hasPictorialBackground .? ;
10   :hasSetting .? ;
11   :materials .? ;
12   :shows .? ;
13   a [ :Image ]
14 }
```

Figure 2: ShEx Schema of *Image* Shape.

The above *Image* shape can be translated into a corresponding SPARQL query using the ShEx2SPARQL tool. Figure 3 depicts the generated SPARQL query in a condensed form. While for the illustrated example a CONSTRUCT query was chosen, the tool does also allow the generation of ASK and SELECT queries.

```
00 CONSTRUCT {
01   ?image a ?type.
02   ?image lio:hasTag ?tag.
03   ?image lio:hasDepictedBackground ?depictedBg.
04   (... shortened ...)
05 } WHERE {
06   ?image a ?type.
07   VALUES ?type { lio:Image }
08   OPTIONAL {
09     ?image lio:hasTag ?tag.
10     FILTER((DATATYPE(?tag)) = xsd:string)
11   }
12   OPTIONAL { ?image lio:hasDepictedBackground ?depictedBg. }
13   (... shortened ...)
14 }
```

Figure 3: Generated SPARQL Query corresponding to *Image* Shape (Shortened).

The CONSTRUCT graph template on lines 1–4 contains all unique graph patterns that occur within the WHERE clause. The graph pattern on line 6 with the VALUES restriction on the subsequent line express the enforced type constraint to the *:Image* class. The *Image* shape consists of Triple Constraints specified on lines 4–7 in Fig. 2, which describe the (optional) presence of datatype properties together with Node Constraints, which constrain the value ranges of literals to the *xsd:string* datatype. The corresponding segment in the SPARQL query can be found in Fig. 3 within the OPTIONAL fragment on lines 8–11, with a graph pattern described on line 9 and the datatype constraint expressed using a

FILTER on line 10.

The schema contains further Triple Constraints on lines 8–12. These constraints are mapped to SPARQL structures as shown on line 12 in Fig. 3. As previously, the constraints are mapped to a corresponding graph pattern. However, an additional FILTER is not required as the schema does not specify any Node Constraints for these patterns. For the remaining properties, the complete SPARQL query contains analogous segments to those on lines 8–11 for datatype properties, as well as line 12 for the remaining properties without Node Constraints.

4.2. Case Study 2: Survey Data

Surveys such as standardized questionnaires play an important role in several different areas of research. Data gathered in surveys can be presented in a structured form that establishes the relationships between the survey, the questions it consists of, answers to the questions, and participants that took the survey. A way to represent survey data in RDF is presented by Scrocca et al. with *The Survey Ontology* [9]. Similarly to the first case study, a shape named *CompletedQuestion* is defined: in the following, we will focus solely on classes and properties of *The Survey Ontology* that are needed for describing this shape. This comprises the following classes: *SurveyDataSet*, *SurveyCompletionTask*, *CompletedQuestion*, *Question*, *Answer* and *Participant*.

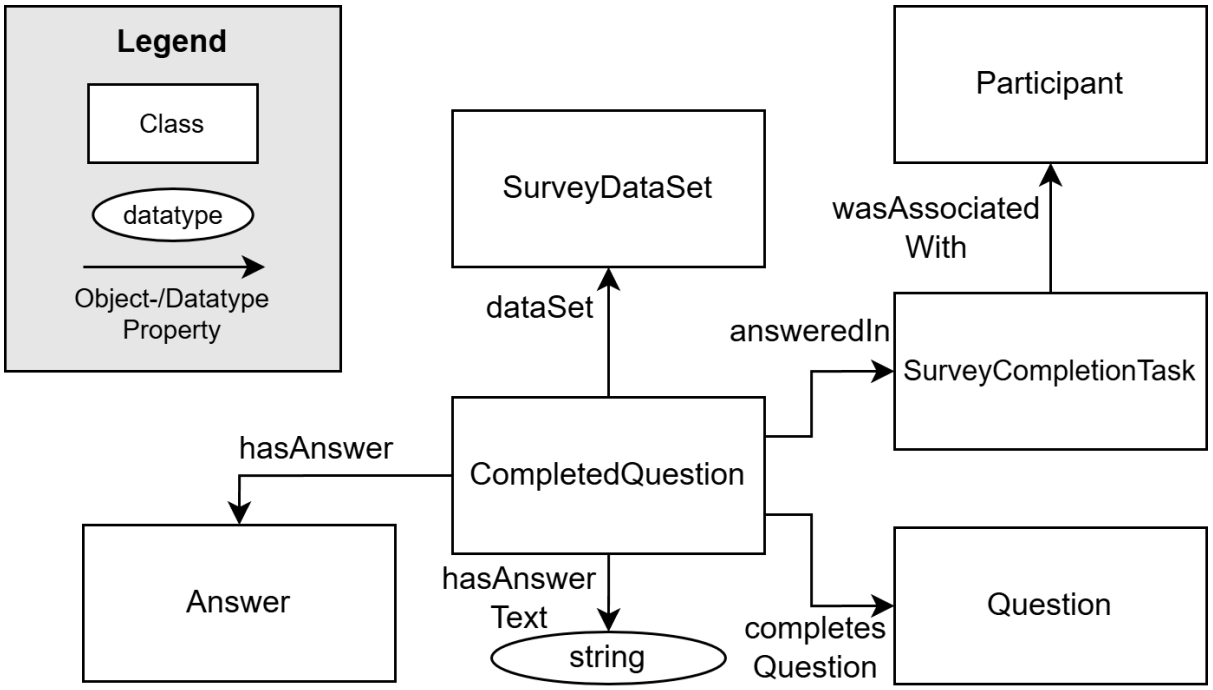

Figure 4: Classes and Properties of *The Survey Ontology* (Excerpt).

The relationships between these classes are shown in Fig. 4 and are described in the ontology as follows: One or multiple instances of a completed question (*CompletedQuestion*) are part of a survey dataset (*SurveyDataset*) and are related to an *Answer* (*hasAnswer*) and an answer text (*hasAnswerText*). Furthermore, a completed question has been answered (*answeredIn*) in the context of a *SurveyCompletionTask* which is associated (*wasAssociatedWith*) with a particular *Participant*. *The Survey Ontology* describes further subclass relationships that we have omitted from our example for reasons of clarity, as they were not relevant for the ShEx schema.

We follow the same procedure as in the first case study and formalize the aforementioned relationships in a ShEx schema by defining a *CompletedQuestion* shape, as shown in Fig. 5. This shape specifies the expected structure of survey-related RDF data based on properties and classes defined in *The Survey Ontology*. First, the schema defines the *CompletedQuestion* shape as its start shape. Lines 5–8 specify the

```
00 PREFIX : <https://w3id.org/survey-ontology#>
01 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
02 PREFIX cube: <http://purl.org/linked-data/cube#>
03 start=@<CompletedQuestion>
04 <CompletedQuestion> {
05   :answeredIn @<SurveyCompletionTask>? ;
06   :completesQuestion @<Question>? ;
07   cube:dataSet @<SurveyDataSet>? ;
08   :hasAnswer @<Answer>? ;
09   :hasAnswerText xsd:string? ;
10   a [ :CompletedQuestion ]
11 }
12 <Answer> { a [ :Answer ] }
13 <Observation> { a [ :Observation ] }
14 <Participant> { a [ :Participant ] }
15 <Question> { a [ :Question ] }
16 <SurveyCompletionTask> { a [ :SurveyCompletionTask ] }
17 <SurveyDataSet> { a [ :SurveyDataSet ] }
```

Figure 5: ShEx Schema of *CompletedQuestion* Shape.

```
00 CONSTRUCT {
01  ?completedQuestion a ?questionType.
02  ?completedQuestion cube:dataSet ?dataSet.
03  ?dataSet a ?dataSetType.
04  ?completedQuestion sur:hasAnswerText ?hasAnswerText.
05  (... shortened ...)
06 } WHERE {
07   ?completedQuestion a ?questionType.
08   VALUES ?questionType { sur:CompletedQuestion }
09   OPTIONAL {
10     ?completedQuestion cube:dataSet ?dataSet.
11     {
12       ?dataSet a ?dataSetType.
13       VALUES ?dataSetType { sur:SurveyDataSet }
14     }
15   }
16   OPTIONAL {
17     ?completedQuestion sur:hasAnswerText ?hasAnswerText.
18     FILTER((DATATYPE(?hasAnswerText)) = xsd:string)
19   }
20   (... shortened ...)
21 }
```

Figure 6: Generated SPARQL Query corresponding to *CompletedQuestion* Shape (Shortened).

CompletedQuestion shape as comprising of the four object properties *:answeredIn*, *:completesQuestion*, *cube:dataSet* and *:hasAnswer*, whose objects are in turn described with the shapes *SurveyCompletionTask*, *Question*, *SurveyDataSet* and *Answer*. The shape includes also the datatype property *:hasAnswerText* and enforces the literal's datatype for this property to be *xsd:string* on line 9. In addition, each shape holds an object property that specifies the type of the respective shape (lines 10, 12–17). With the exclusion of the type properties ('*a*' alias rdf:type), all properties of the schema are specified to be optional.

Next, the *CompletedQuestion* shape is translated into a SPARQL CONSTRUCT query using the ShEx2SPARQL tool. The resulting SPARQL query is shown in Fig. 6. Lines 9–15 specify a graph pattern for the *cube:dataSet* property using the variable *?dataSet* for the object of the triple. As objects for this

relation must conform to a shape — in this particular example, the *SurveyDataSet* shape — a nested graph pattern follows on lines 11–14. This nested graph pattern describes the constraints associated with the *SurveyDataSet* shape, i.e. its type property as well as the VALUES expression enforcing its type to be *sur:SurveyDataSet*. In a similar manner, the type of the *CompletedQuestion* is constrained on lines 7–8. The segment on lines 16–19 reflects the mapping of the datatype property *:hasAnswerText* and constrains its value range to *xsd:string* using a FILTER expression. For all other properties not included in the excerpt, the full query contains segments that are structured in the same way as those on lines 16–19.

5. Discussion

In the two case studies shown, we demonstrated how SPARQL queries can be constructed automatically from a ShEx schema for image and survey data respectively. In these examples, an ontology was assumed as the starting point, from which then a corresponding ShEx schema was derived. In practice, however, it is common for multiple ontologies to be used in combination with each other — as is also the case, for example, with the complete version of The Survey Ontology. Such scenarios can be realized with the shown procedure as well: properties and classes from different ontologies can be incorporated easily in the derived ShEx schema, the translation of the ShEx schema to a SPARQL query remains unaffected.

Some Semantic Web applications (such as Wikidata[4]) facilitate data retrieval for their users by providing them with SPARQL query templates for reference. Using our approach, such templates can be generated directly from ShEx schemas. We believe that this approach can be particularly useful for instances where shape expressions are already employed for data validation, as this eliminates the effort of developing the ShEx schemas or, if needed at all, necessitates only minor adjustments to the schemas.

Although our case studies focused exclusively on research data management, the procedure can be applied to other domains as well. It should be noted, however, that the aforementioned limitations of the ShEx2SPARQL tool still apply.

6. Conclusion

In this paper, we demonstrate how the accessibility of RDF data can be enhanced, particularly in the context of research data, by leveraging the formal structure defined in a Shape Expressions (ShEx) schema. We demonstrate the necessary steps through two case studies, one on image data utilizing the *Lightweight Image Ontology*, and on survey data using *The Survey Ontology*. A ShEx schema is derived from each of the two ontologies, serving as a blueprint for generating a corresponding SPARQL query. The translation of ShEx schemas into SPARQL is realized using our ShEx2SPARQL tool, which facilitates the retrieval of instance data conforming to the schema. This approach is particularly beneficial for users who otherwise struggle with the manual formulation of complex queries or who are unfamiliar with the ontologies used in a knowledge graph. Overall, the approach represents a step toward more accessible RDF-based research data management, ultimately fostering the effective use of Linked Open Data in research. Future work includes extending ShEx2SPARQL to translate all types of cardinality constraints.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — Project-ID 514664767 — TRR 386.

[4] https://query.wikidata.org/

Declaration on Generative AI

During the preparation of this work, the authors used GPT-4 and DeepL in order to: Grammar and spelling check, Improve writing style. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] B. Gualandi, L. Pareschi, S. Peroni, What do we mean by "data"? A proposed classification of data types in the arts and humanities, Journal of Documentation 79 (2022) 51–71. doi:10.1108/JD-07-2022-0146, publisher: Emerald Publishing Limited.

[2] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, et al., The FAIR Guiding Principles for scientific data management and stewardship, Scientific Data 3 (2016) 160018. doi:10.1038/sdata.2016.18.

[3] W3C OWL Working Group, OWL 2 Web Ontology Language Document Overview (Second Edition), 2012. URL: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[4] G. Klyne, J. J. Carroll, B. McBride, RDF 1.1 Concepts and Abstract Syntax, 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[5] S. Harris, A. Seaborne, E. Prud'hommeaux, SPARQL 1.1 Query Language, 2013. URL: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[6] E. Prud'hommeaux, I. Boneva, J. Emilio, G. Kellog, Shape Expressions Language 2.1, 2019. URL: https://shex.io/shex-semantics-20191008/.

[7] M. Warren, P. Hayes, Lightweight Image Ontology (LIO), 2010. URL: https://w3id.org/lio/v1, accessed: 8 March 2025.

[8] M. Warren, D. A. Shamma, P. Hayes, Knowledge Engineering with Image Data in Real-World Settings, in: Proceedings of the AAAI Spring Symposium on Combining Machine Learning and Knowledge Engineering, 2021, pp. 1–4.

[9] M. Scrocca, D. Scandolari, G. Re Calegari, I. Celino, The Survey Ontology, 2021. URL: https://w3id.org/survey-ontology, accessed: 8 March 2025.

[10] I. Baroni, I. Celino, D. Scandolari, G. Re Calegari, M. Scrocca, The Survey Ontology: Packaging Survey Research as Research Objects, in: 2nd Workshop on Data and Research Objects Management for Linked Open Science, 2021, pp. 1–10. doi:10.4126/FRL01-006429412.

[11] B. Goossens, O. Mussa, P. Orozco-terWengel, C. Perera, Making linked-data accessible: A review, 2022. URL: https://core.ac.uk/works/144545039/.

[12] E. Kuric, J. D. Fernández, O. Drozd, Knowledge Graph Exploration: A Usability Evaluation of Query Builders for Laypeople, in: M. Acosta, P. Cudré-Mauroux, M. Maleshkova, T. Pellegrini, H. Sack, Y. Sure-Vetter (Eds.), Semantic Systems. The Power of AI and Knowledge Graphs, Springer International Publishing, Cham, 2019, pp. 326–342. doi:10.1007/978-3-030-33220-4_24.

[13] S. Malyshev, M. Krötzsch, L. González, J. Gonsior, A. Bielefeldt, Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph, in: D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, E. Simperl (Eds.), The Semantic Web – ISWC 2018, Springer International Publishing, Cham, 2018, pp. 376–394.

[14] P. Hoefler, M. Granitzer, E. Veas, C. Seifert, Linked Data Query Wizard: A Novel Interface for Accessing SPARQL Endpoints, in: CEUR Workshop Proceedings, volume 1184, 2014, pp. 1–10.

[15] F. Haag, S. Lohmann, S. Siek, T. Ertl, QueryVOWL: Visual Composition of SPARQL Queries, in: F. Gandon, C. Guéret, S. Villata, J. Breslin, C. Faron-Zucker, A. Zimmermann (Eds.), The Semantic Web: ESWC 2015 Satellite Events, Springer International Publishing, Cham, 2015, pp. 62–66. doi:10.1007/978-3-319-25639-9_12.

[16] S. Lohmann, S. Negru, F. Haag, T. Ertl, Visualizing Ontologies with VOWL, Semantic Web 7 (2016) 399–419.

[17] H. Vargas, C. Buil-Aranda, A. Hogan, C. López, RDF Explorer: A Visual SPARQL Query Builder,

in: C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, F. Gandon (Eds.), The Semantic Web – ISWC 2019, Springer International Publishing, Cham, 2019, pp. 647–663. doi:10.1007/978-3-030-30793-6_37.

[18] S. Ferré, Sparklis: An Expressive Query Builder for SPARQL Endpoints with Guidance in Natural Language, Semantic Web 8 (2017) 405–418. doi:10.3233/SW-150208, publisher: IOS Press.

[19] R. Cocco, M. Atzori, C. Zaniolo, Machine Learning of SPARQL Templates for Question Answering Over LinkedSpending, in: 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2019, pp. 156–161. doi:10.1109/WETICE.2019.00041, iSSN: 2641-8169.

[20] P. Chen, Y. Lu, V. W. Zheng, X. Chen, X. Li, An Automatic Knowledge Graph Construction System for K-12 Education, in: Proceedings of the Fifth Annual ACM Conference on Learning at Scale, L@S '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1–4. doi:10.1145/3231644.3231698.

[21] T. A. Taffa, R. Usbeck, Leveraging LLMs in Scholarly Knowledge Graph Question Answering, 2023. doi:10.48550/arXiv.2311.09841, arXiv:2311.09841 [cs].

[22] T. Baker, E. Prud'hommeaux, Shape Expressions (ShEx) 2.1 Primer, 2019. URL: https://shex.io/shex-primer-20191009/.