OBIS Extractor: Create a Schema for Your Knowledge Graph

Kārlis Čerāns[1,*], Aiga Romāne-Ritmane[1,*], Mikus Grasmanis[1,*] and Lelde Lāce[1]

[1]Institute of Mathematics and Computer Science, University of Latvia, Raiņa bulvāris 29, Rīga, LV-1459, Latvia

Abstract

We present the OBIS Schema Extractor tool for extracting rich data schemas from a SPARQL endpoint, involving class-to-property, as well as class-to-class and property-to-property relations, ready to be stored into a schema database and used for visual schema diagram creation and visual and textual query auto-completion.

Keywords

Knowledge graphs, Knowledge graph schema, Visual schema diagrams, Visual queries

1. Introduction

A schema of a Knowledge Graph (KG) describes vocabularies of properties and classes used in the KG construction, together with patterns of their co-occurrence (e.g., what properties can be observed for instances of what classes). Knowing the data schema is essential for creating queries over the data. The schema knowledge can help tools to provide a better user experience with the data set, e.g., by means of query auto-completion or schema visualization.

Still, the KG creation and publishing philosophy does not require explicit creation of its data schema, so obtaining the KG data schema is usually to be done independently of the KG data publication.

We demonstrate the feasibility of the task of extracting a data schema from a KG by providing a generic tool that is able to extract the schema from the SPARQL endpoint and store it into a database, ready for independent analysis, as well as for data schema and their fragment visualization (cf. [1]) and context-sensitive data query auto-completion (cf. e.g., [2]).

There are various existing approaches for KG schema description, starting from RDF VoID vocabulary[1], including SHACL [3] and ShEx [4] data shape languages, as well as earlier ontology languages, such as OWL [5] (despite its "open-world" semantics not being well suited to describe data schemas).

Most of these approaches, except VoID, are primarily meant for prescribing the data schema structure, as it is *to be*, not what is actually found in a particular KG (therefore, there are no native means for describing, e.g., the frequencies of a class or property, or their co-occurence, in SHACL, ShEx or OWL).

The existing work on schema extraction largely focuses on VoID (cf. VoID Generator [6], or Agentic Data Catalogue[7] support tools) and SHACL schema formats (cf. QSE [8] or Shacl!Play infrastructure [9]). This work, while impressive, supports only partial information about the data schemas, usually leaving out the class-to-class connections (class intersections, subclasses), important for schema visualizations, and property-to-property connections (such as one property following another in a path, or two properties sharing the triple subjects or triple objects), important for query auto-completion (e.g., in the quite typical cases when the class information for a variable may not be known).

The OBIS Schema Extractor tool, described in this paper, provides reliable means for extracting rich schema descriptions, involving the class-to-property, property-to-class, class-property-class, as well as class-to-class and property-to-property links, property domain/range information and cardinalities.

Semantics 2026: Posters and Demos, September 15–17, 2026, Ghent, Belgium

*Corresponding author.

karlis.cerans@lumii.lv (K. Čerāns); aiga.romane@inbox.lv (A. Romāne-Ritmane); mikus.grasmanis@lumii.lv (M. Grasmanis); lelde.lace@lumii.lv (L. Lāce)

0000-0002-0154-5294 (K. Čerāns); 0009-0003-3609-1485 (A. Romāne-Ritmane); 0000-0002-0668-0970 (M. Grasmanis); 0000-0001-7650-2355 (L. Lāce)

[1]https://www.w3.org/TR/void/

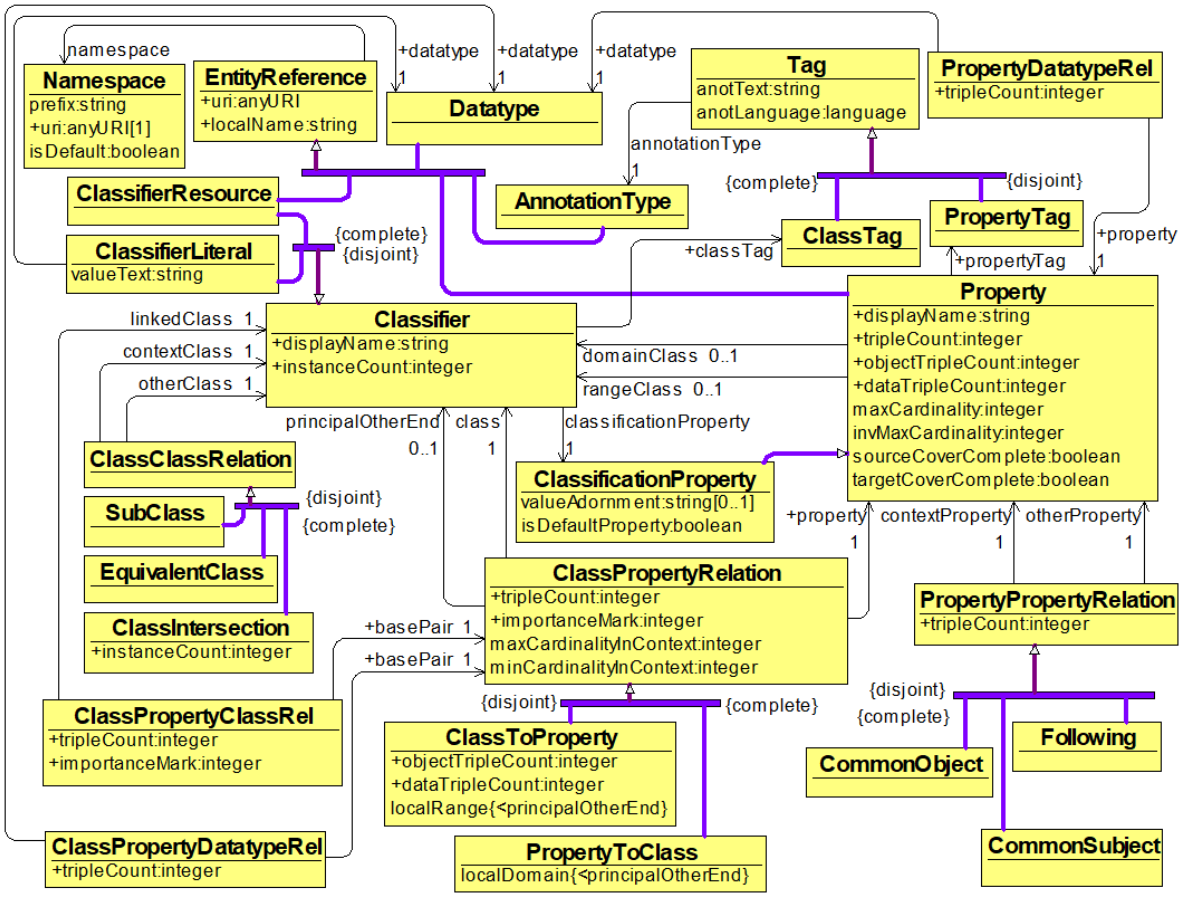

Figure 1: Conceptual data schema structure

2. Data Schema Model

The conceptual data meta-schema (schema of schemas, created by the extractor and used in the applications) is summarized in Fig. 1 (using OWLGrEd notation [2], with the UML style modeling and schema description plugins[3]). The core part of the schema is explained mathematically in [1]. An early structural schema presentation is in [10]. Some schema highlights are, as follows.

The data schema is based on the central (meta-)classes *Classifier* and *Property*, where a property is an entity reference (*EntityReference*, with *URI*, *namespace* and *localName*), while a classifier can be a *ClassifierResource* that is an entity reference, or *ClassifierLiteral* that has the classification text (*classifText*) and datatype instead. Importantly, each classifier has a classification property (*classificationProperty*), being *rdf:type* or possibly some other property. The frequencies for classes (classifiers) and properties are recorded in their *instanceCount* and *tripleCount* attributes. *objectTripleCount* and *dataTripleCount* for a property counts the triples with this property with an IRI/blank node or a literal in its object position, respectively. *domainClass* and *rangeClass* for a property have their strict, ontological meaning.

ClassPropertyRelation repesents situations where a class is a source (*ClassToProperty*) or target (*PropertyToClass*) of a property. *ClassPropertyClassRel* corresponds to triples with the property linking instances of two classes. The *importanceMark* attribute for class-property and class-property-class relations designate the ”principal ascription points” for a property, when visual schema diagrams are created, so that a property would not get ascribed both to a superclass and a subclass (cf. [1]).

PropertyPropertyRelation describes property co-occurrence pattern situations (one property ”fol-

[2]https://owlgred.lumii.lv

[3]An ascription of a property to a class indicates that the class is a domain for the property, except when the property name is prefixed by +.

lowing" the other, or two properties sharing the same subjects or the same objects), together with *otherProperty* triple count in the context of *contextProperty* occurrence.

3. Schema Extraction

The data schema is extracted by the OBIS Schema Extractor[4] software that is a JAVA-based web application which can be run on any internet-connected desktop computer. When run, the OBIS Schema Extractor executes a number of customized SPARQL queries against the SPARQL endpoint, aggregates the answers and produces a schema in JSON format to be imported into the schema database by a dedicated import utility[5].

The extractor receives the SPARQL endpoint (and a Named Graph, if applicable) as its parameters. The aim of the schema extraction is to fill the structure, as defined in Fig. 1 by the data from the considered endpoint. The Extractor sequentially obtains the following information:

- Classifier (class) information, with class instance counts
- Class intersection and subclass information
- Basic property information, with triple counts
- Detailed property information, including:
  - Source and target classes, source and target class pairs (with triple counts)
  - Source and target class relevance (importance marks) for the property, and the property in source/target class context
  - Property object data type information (also in source class context)
  - Domain and range (including local domains/ranges) and closed sources/targets (the property source/target class union cover all non-literal subjects/objects used in the property triples)
  - Property-property relations (co-occurence)
  - Cardinalities (both global and constrained by a source/target class)

3.1. Extraction Parameters

The extraction scope can be configured to exclude or restrict the extraction of some aspects of the data schema. For instance, the property class pair (class-property-class relations) or property co-occurrence (property-property relations) can be excluded from the extraction. The minimum class size can be specified for detailed information collection about the class, or the schema slice corresponding just to fixed lists of classes and properties can be considered; the restrictions can apply to the calculations of importance marks, domains and ranges, cardinalities, and data types, as well.

As a rule of thumb, full schemas comprising about 1000 classes and properties can be successfully extracted, if a couple of days can be allowed for the process, whereas for larger schemas some restrictions may need to be introduced. The property-property relations are ones of most time consuming, but these can also be ones of most interesting.

A restrictive parameter that is generally recommended for larger data sets is the data type sampling set size (set it to about 100K or 1M); the data types can be less crucial in schema applications, and unconstrained data type calculations can take unreasonably long time.

The full parameter model of the extractor, as well as the immediate structure of the .json files produced by the extractor, are described in its code repository wiki[6], including the possibility to specify multiple classification properties to be used for the schema extraction. There are options of manual parameter selection in a Swagger[7]-based interface and using a configuration file for richer configuration options.

The extractor console logs its activity stages, as well as its progress information that can be used to estimate, whether the extractor progress over the endpoint is going to fit into the expected time frame, or certain adjustments would be necessary.

[4]https://github.com/LUMII-Syslab/OBIS-SchemaExtractor
[5]https://github.com/LUMII-Syslab/data-shape-server/tree/main/import-generic
[6]https://github.com/LUMII-Syslab/OBIS-SchemaExtractor/wiki/OBIS-Schema-Extractor-wiki
[7]https://swagger.io

3.2. Extraction Strategy and Failure Handling

Initially, the extractor attempts to retrieve the information about specific aspects of the schema by means of large generic queries; if the initial query fails, backup strategies for information extraction are invoked, if possible. Such a strategy may include smaller-granularity queries (e.g., for a property check all classes individually, if the class is a source class for the property), or it may include limit-based estimates and queries without any statistics information.

If not specified otherwise, the check of the structural aspects of the schema (e.g., class intersection, class-to-property, class-property-class and property-property links) is attempted down to a single-line (LIMIT 1) query of existence of the respective pattern in the data set. If even the single-line pattern query fails, the non-existence of the pattern is recorded, together with a WARNING mark in the extractor notes, enabling possibly to re-evaluate the failures at a later point manually.

The failure of queries for domain/range and closed source/target assertions, and cardinalities is conservatively interpreted as the checked assertion/restriction not being true (the schema is, in a sense, less informative), with a WARNING mark raised, so that the situations can be localized and the respective assertions introduced manually.

The extractor has a protection mechanism against temporary endpoint failures, as well: after a failing query a simple endpoint liveness check is performed, and if that fails, as well, the extraction process is suspended until the endpoint becomes live again (tests done after a minute, then every 15 minutes).

The templates of queries used by the extractor are available explicitly in a dedicated file[8].

4. Applications

The described schema extraction process is an essential part of visual schema diagram creation in [1] and visual schema-backed query auto-completion (cf. e.g., [2]) within the frame of ViziQuer tool[9]. The schema definitions, created by the extractor and stored into the schema database provide the information needed both for the schema diagram visualization and the query auto-completion.

Technically, the schema diagram database is handled by a dedicated Data Shape Server[10] (DSS) that can serve the data schema information for ViziQuer, as well as other potential users, as explained below.

The ViziQuer Tools package[11] allows easy local setup of both the ViziQuer and DSS components. This approach allows to store and handle the OBIS Extractor schemas locally.

The ViziQuer playground[12] provides ready-to use visual environments for over 200 unique public SPARQL endpoints; an RDF-based view on the contents of part of its DSS database is available also as a SPARQL endpoint[13] (work in progress).

The DSS server, based on the schemas obtained by the Extractor, can also be used to offer schema-based auto-completion during textual SPARQL query creation. Fig. 2 shows an example of SPARQL text auto-completion, where the suggested properties for the subject ?S are offered in the context of ?S being just an object of another property triple (no class information provided). The illustrated SPARQL query auto-completion is based on methods created in [11] and is available both within the textual query view inside the ViziQuer environment and within an independent YASGUI-style DASA client[14] prototype.

5. Conclusions

OBIS Schema Extractor demonstrates the feasibility of extracting rich data schemas from a wide range of SPARQL endpoints. The schemas are suitable for visual diagrammatic presentation, use for query

[8] https://github.com/LUMII-Syslab/OBIS-SchemaExtractor/blob/master/build/queries.properties
[9] https://viziquer.lumii.lv/
[10] https://github.com/LUMII-Syslab/data-shape-server
[11] https://github.com/LUMII-Syslab/viziquer-tools
[12] https://viziquer.app
[13] https://qlever.semtech.lv/schemas
[14] https://yasgui.semtech.lv/

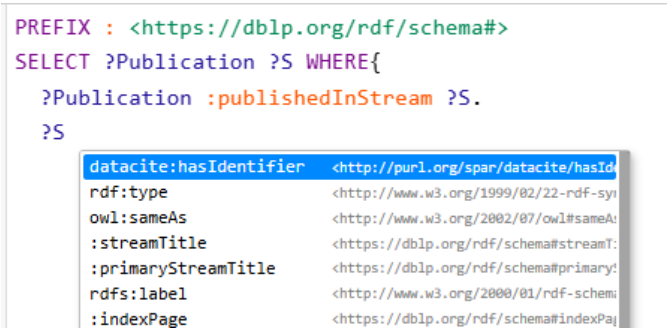

Figure 2: Property-based SPARQL Query Auto-completion

auto-completion (including the property co-occurrence patterns), as well as independent analysis.

Further work includes widening the test/use case basis for the extractor, as well as the expansion of the eco-system of the data schemas that are obtained by the means of using the extractor.

Acknowledgments

This work has been partially supported by a Latvian Science Council Grant lzp-2024/1-0665 “What is in Your Knowledge Graph?”.

Declaration on Generative AI

An AI tool (ChatGPT, v5.5) was used for reference BibTex entry formatting.

References

[1] L. Lāce, A. Romāne-Ritmane, M. Grasmanis, A. Sproģis, J. Ovčiņņikova, U. Bojārs, K. Čerāns, Visual presentation and summarization of Linked Data schemas, in: Proc. of KGSWC 2024, volume 15459 of *Lecture Notes in Computer Science*, 2024, pp. 290–305. doi:10.1007/978-3-031-81221-7_20.

[2] K. Čerāns, U. Bojārs, J. Ovčiņņikova, L. Lāce, M. Grasmanis, A. Sprogis, Towards visual federated sparql queries, in: SEMANTiCS-PDWT 2024, CEUR Workshop Proceedings, volume 3759, 2024. URL: https://ceur-ws.org/Vol-3759/paper24.pdf.

[3] H. Knublauch, D. Kontokostas, Shapes constraint language (SHACL), W3C Recommendation, 2017. URL: https://www.w3.org/TR/shacl/.

[4] E. Prud’hommeaux, J. E. Labra Gayo, H. Solbrig, Shape expressions (ShEx) 2.1 Primer, Shape Expressions Community Group Report, 2019. URL: https://shex.io/shex-primer/.

[5] OWL 2 web ontology language: Structural specification and functional-style syntax, W3C Recommendation, 2012. URL: https://www.w3.org/TR/owl2-syntax/.

[6] Bolleman, Jerven, et.al, A large collection of bioinformatics question-query pairs over federated knowledge graphs: Methodology and applications, 2024. URL: https://arxiv.org/abs/2410.06010. arXiv:2410.06010, arXiv preprint arXiv:2410.06010.

[7] The Catalogue for the Agentic SPARQL, Web catalogue, 2025. URL: https://catalogue.ai.wu.ac.at/, A registry of SPARQL endpoints, ontologies and semantic wikis optimized for agentic AI. Accessed 14 June 2026.

[8] K. Rabbani, M. Lissandrini, K. Hose, Extraction of validating shapes from very large knowledge graphs, Proc. of the Very Large Databases 16 (2023) 1023–1032.

[9] Sparna, Shacl play!, 2025. URL: https://shacl-play.sparna.fr/play/, accessed: 2026-06-14.

[10] S. Rikačovs, K. Čerāns, Visual data and schema queries over knowledge graphs, in: EKAW-PDWT 2024, volume 3967 of *CEUR Workshop Proceedings*, CEUR-WS.org, Amsterdam, Netherlands, 2024. URL: https://ceur-ws.org/Vol-3967/PD_paper_188.pdf, poster and Demo Track.

[11] O. Putāns, Improving SPARQL query construction using empirical ontology metadata. Bachelor thesis, University of Latvia, 2026.