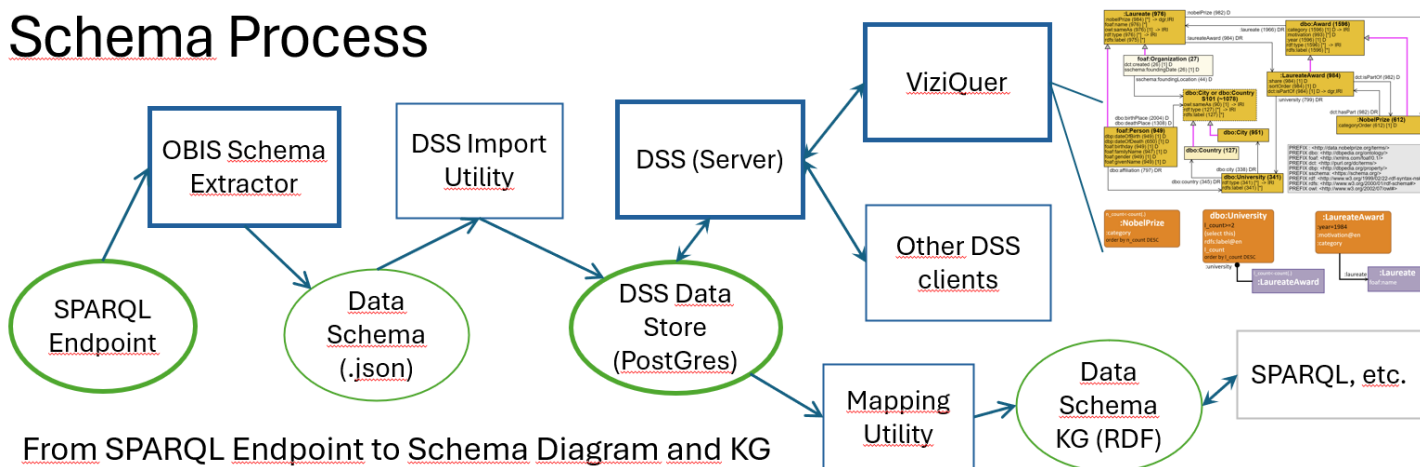


OBIS Extractor: Create a Schema for Your Knowledge Graph

Kārlis Čerāns, Aiga Romāne-Ritmane, Mikus Grasmanis, Lelde Lāce
Institute of Mathematics and Computer Science, University of Latvia
karlis.cerans@lumii.lv

Schema Process



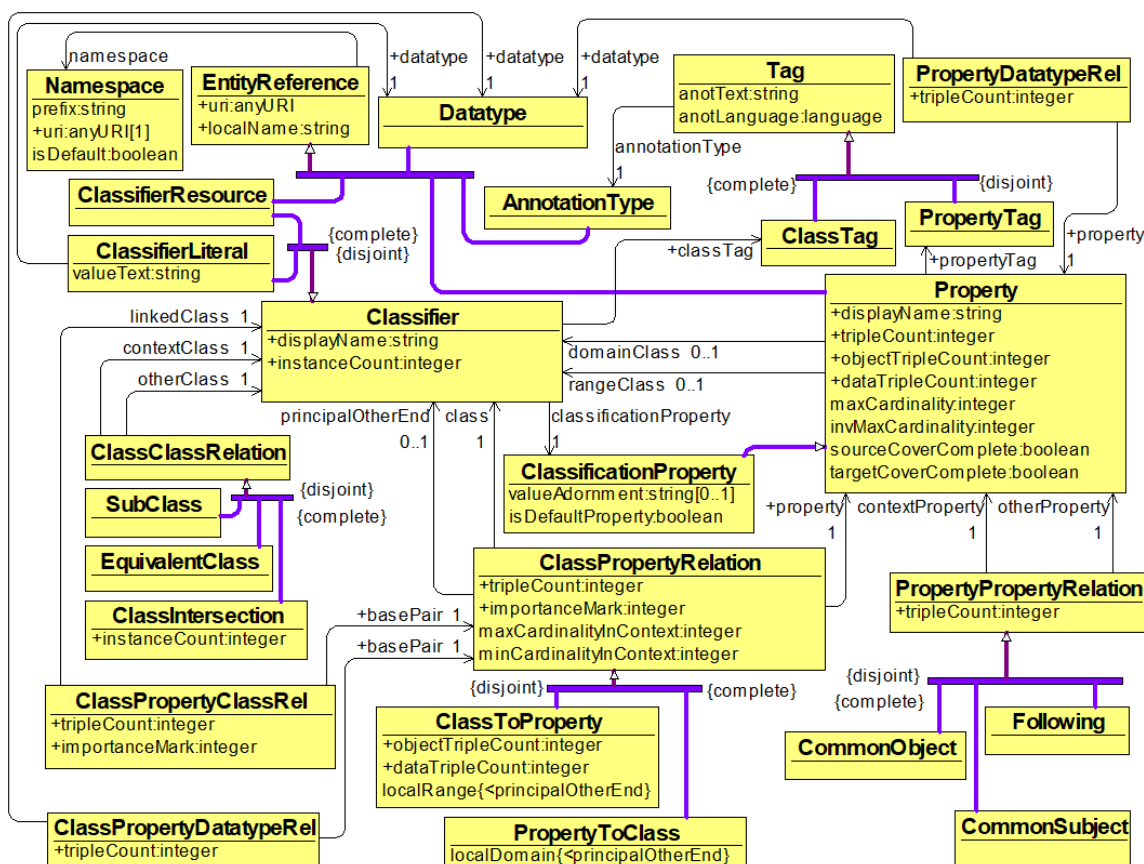
OBIS Extractor: locally deployable Java service for schema extraction from SPARQL endpoints <https://github.com/LUMII-Syslab/OBIS-SchemaExtractor>



Direct link to wiki:



Data Schema Structure



KG?

Supported in part by Latvian Science Council Grant lzp-2024/1-0665 "What is in Your Knowledge Graph?". https://viziquer.lumii.lv/projects/lzp_2024/



Latvian Council of Science

OBIS Extractor: Create a Schema for Your Knowledge Graph

Extraction highlights:

- **Sequence of SPARQL queries** for extracting various data schema aspects (e.g., class list, property list, subclass relation, class-to-property, property-to-class, class-property-class and property-to-property relations; cardinalities, datatypes, property domains and ranges)
- **Large generic queries** backed up by **sequences of individualized queries**, if the initial queries fail (e.g., timeout); examine a relation down to a single pattern existence (assume non-existence, if this fails)
- **Quality marks** and extraction **message logs** allow estimating extraction quality
- Tolerance for **temporary endpoint downtimes** (liveness checks in case of query failures)
- **Parameters** for extraction scope and precision. Consult the paper for **parameter value recommendations** (e.g., limit the data type extraction scope)
- **Missing quantity estimates and imputation** during schema import stage into DSS
- **Extraction time** from a couple of **minutes** to a couple of **days** (e.g., for a fully detailed extraction of a schema with about 1000 classes and 1000 properties).

Applications: visual schema diagrams (visual and textual query auto-completion possible, as well)

POST /schema-extractor-rest/v2/endpoint/buildFullSchema

Extract and analyze data from SPARQL endpoint and build full schema model (version 2)

Cancel

Parameters

Name	Description
endpointUrl * required string (query)	SPARQL Endpoint URL, for example, http://localhost:8890/sparql https://data.nobelprize.org/store/sparql
graphName string (query)	Named Graph (optional). If no graph name provided, the search will involve all graphs from the endpoint graphName - Named Graph (optional). If no
calculateSubClassRelations * required boolean (query)	Calculate subclass relation (strongly recommended) true
calculateMultipleInheritanceSuperclasses * required boolean (query)	Include multiple inheritance check, strongly recommended. If set to false, no more than a single direct super-class for a class shall be found. Set false only for very large endpoints, if the processing of subclass relations is stuck in the extractor for a longer time. true
minimalAnalyzedClassSize * required integer (\$int32) (query)	Minimal analyzed class size. The classes below the specified size shall be included in the schema, still they shall not be considered as possible super-classes of other classes and their links to properties shall not be analyzed individually. If 1, all classes should be analyzed. Values above 1, e.g. 10 or 100, may be essential for endpoints with more than several thousand classes. 1
requireClasses * required boolean (query)	At least 1 class is required for the schema extraction, if no classes found - the extractor work is aborted true
calculatePropertyPropertyRelations * required boolean (query)	Calculate property-property adjacency: following properties, same subject, same object. Important for visual and textual query auto-completion use case; can be used in schema visualization if property sources and targets are considered as potential nodes (work in progress). Can be time-consuming for larger schemas, consider appropriate propertyPropertyLinkCheckBackupMode true
propertyPropertyLinkCheckBackupMode * required string (query)	Set backup validation queries to calculate property-property relations in the case of a failing property co-occurrence query for a given base property. details, limitsPlus and limitsPlusSimple guarantee trying a single-line pattern of two property co-occurrence, if larger patterns fail. details gives exact statistics for two property co-occurrence, whenever possible. limitsPlus attempts at least a limit-based estimate, limitsPlusSimple can stay with yes/no answer for co-occurrence for performance reasons (with some count information imputed at later schema import steps). none and limits are faster, but can result in partially incomplete structure information. details

(more parameters in UI and config file)

