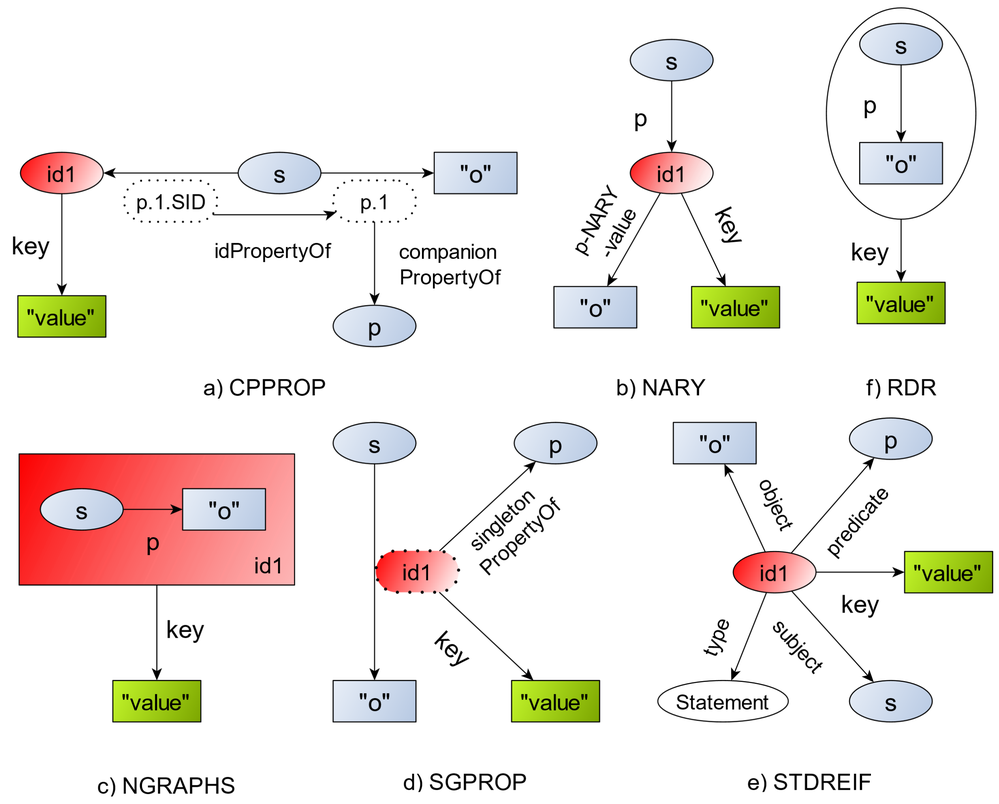
MaSQue: An Approach for Flexible Metadata Storage and Querying in RDF

1. Motivation

The maintenance and use of (statement-level) metadata, such as provenance or time-related information (when was a triple created or retrieved), is of increasing importance in the Semantic Web. Especially for Big Data applications, that work on heterogeneous data fused from multiple sources and which require high data quality, data processing metadata is essential. The storage of such metadata alongside the data in the same RDF store allows to record fine-grained traceability and provenance information, license and access rights, data trustworthiness and confidence scores for every single fact in a knowledge graph. However, studies investigating the performance of Metadata Representation Models (MRMs) show that choosing the appropriate metadata representation depends on the used data and metadata, queries and RDF store. Thus it is challenging to determine the best MRM for a scenario beforehand. To enable the development of an RDF application extensively using metadata, but without restricting it a-priori to one concrete implementation of an MRM, we propose MaSQue (Metadata Storage and Querying).

2. Metadata Representation Models (MRMs)

Six different ways of describing (or reifying) an RDF triple s, p, o with a metadata key and value pair are supported by MaSQue; Companion property (cpprop), nary relation (nary), named graphs (ngraphs), singleton properties (sgprop), standard reification (stdreif), and the Blazegraph-specific Reification Done Right (rdr). Besides rdr, which is based on the vendor-independent RDF* and SPARQL*, all approaches use an explicit statement identifier (red), which is used to attach metadata (green) to the data (grey). Cpprop and stdreif are based on additional triple handlers (white). Properties which also occur as subject in another triple are drawn with dashed lines.



3. MaSQue Approach

MaSQue is a Java-based framework and command line utility. Its paradigm is to hide the complexity and individual characteristics of various Metadata Representation Models **(MRMs)** behind a uniform "mask". It allows to switch between different MRMs without rewriting the application logic. It consists of 2 major components **meta-RDF** and **meta-SPARQL**, which establish an abstraction layer for RDF data and its metadata for storage & serialization and querying respectively.

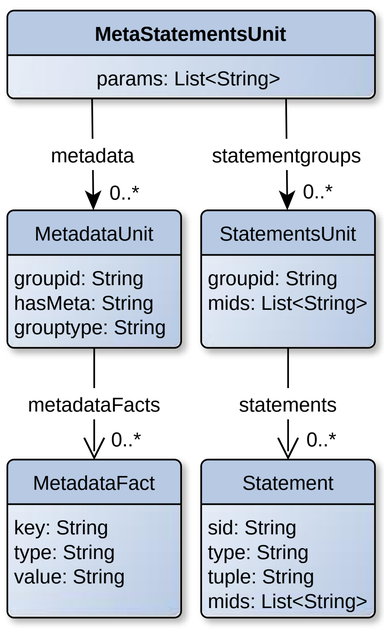
4.1 Meta-RDF: JSON-based metadata representation format

```
{ "statementgroups":[  ***contains all data triples/quads (of one resource) separated into groups
  { ***one statement group contains all triples sharing the same metadata (entity granularity level)
    "groupid":"<http://ex.org/id>", ***id used as (graph) identifier for this group
    "statements":[
      {
        "tuple":"<http://ex.org/person><http://ex.org/name>\"Person\".", ***raw ntriple/nquads
        "sid":"" *** optional, can be used to specify an explicit statement identifier for triple
      }
    ],
    "mids":[ ***list of metadataUnits (link to its groupid field) which hold for that statement group
      "<http://ex.org/meta-1>","<http://ex.org/meta-2>" ***meta-2 not listed for brevity
    ]
  }
 ],
 "metadata":[ ***contains all metadataUnits referenced in the statementgroups
   { ***a metadataUnit groups metadata facts which belong together or which have
       the same meta-metadata
     "groupid": "<http://ex.org/meta-1>", ***the id, if empty the id refers to the number
     "metadataFacts": [
       {
         "type": "kv-meta", ***use simple key value metadata (support for triples in the future)
         "key": "metadatakey", "value":"example value"
       }
     ],
     "grouptype": "flat" , ***shows that the metadata within the group is
                              logically independent: 'strong' for logical unit
     "hasMetadata": ""   ***optionally specify another metadataUnit id describing this metadata unit
   }
 ],
}
```

Excerpt from a meta-RDF JSON file

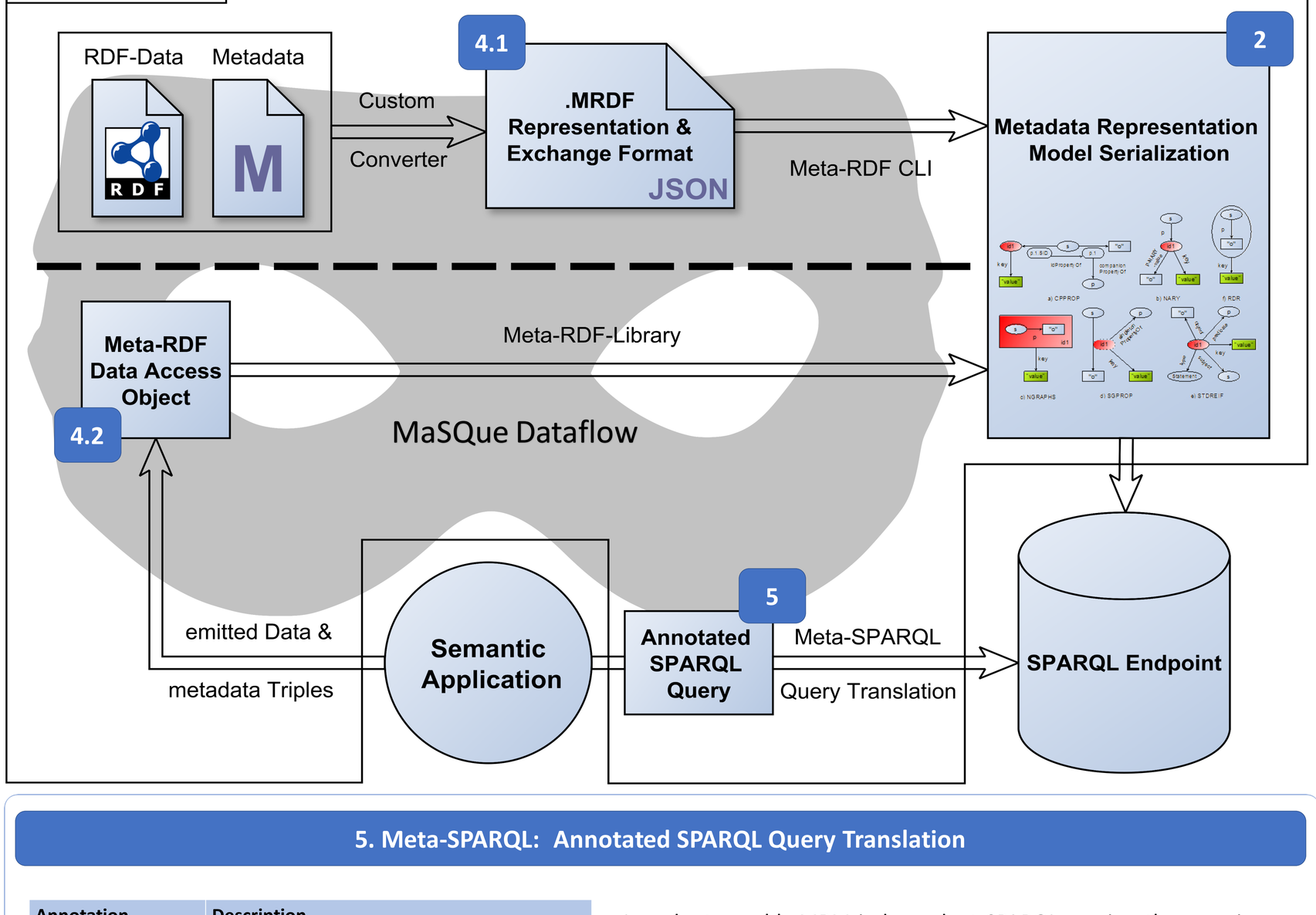
Meta-RDF is designed to convert datasets into various MRMs. The component features a novel JSON representation, which establishes an interoperable (store independ), large scale, file based abstraction layer to explicitly attach metadata to RDF triple(s)/quad(s). Once the source dataset is converted into the JSON representation, this intermediate format can be used to create NQuads files for the various MRMs. The JSON representation is optimized for a parallel conversion of huge datasets, which do not fit into main memory. It supports to easily express shared as well as nested metadata (meta-metadata), logical groups of metadata and different levels of granularity (graph-, entity-. triple-level metadata). In brief the format is able to explicitly represent (meta)information which can be leveraged by the different MRMs. Based on this explicit information, meta-RDF supports different serialization and optimization schemes (factorization for all MRMs, combination of ngraphs with other MRMs for efficient meta-metadata representation, logical metadata groups etc.) for the individual MRM

4.2 Meta-RDF Data Model (DAO)



While the JSON format is intended for a batch conversion of a complete dataset, applications can also use the integrated Java data model abstraction (DAO) to convert RDF metadata on-the-fly.

MaSQue Approach



5. Meta-SPARQL: Annotated SPARQL Query Translation

Annotation	Description
#!data(?s,?p,?o)!#	replacing a regular data triple pattern (for regular data queries)
#!reif(?id,?s,?p,?o)!#	analogous to #!data but retrieving statement id as well
#meta(?id,?k,?v)!#	retrieve metadata key and value, using a statement id
#meta2(?id,?k,?v)!#	retrieve metadata key and value, which is reified itself (due to meta-metadata), using a statement id

```
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT (count(distinct ?company) as ?c)
WHERE {
      #!reif (?id2,?company,dbo:locationCountry,<|country|>)!#
      #!reif(?id,?company,rdf:type,dbo:Company)!#
      #!meta2(?id,<http://ns.inria.fr/dbpediafr/voc#uniqueContributorNb>,?cont)!#
      #!meta2(?id2,<http://ns.inria.fr/dbpediafr/voc#revPerYear2016>,?revs)!#
      FILTER(?revs >5 && ?cont>10)
}
```

Query example

In order to enable MRM-independent SPARQL queries, the generic and extensible tool meta-SPARQL has been developed. It allows automatic rewriting of SPARQL queries for different MRMs. The idea is, to replace every triple pattern within a SPARQL query by a set of special annotations, which will be translated by meta-SPARQL into the appropriate format. Every query needs to be written as a template in an intermediate SPARQL dialect based on these annotations. It consists of 4 annotations explained in the Table on the left. The template can be converted into query instances of the various MRMs. Therefore query templates can be written independent of granularity support and other MRM-specific characteristics.

Acknowledgements

This work was supported by grants from the Federal Ministry for Economic Affairs and Energy of Germany (BMWi) for the Smart Data Web project (GA-01MD15010B), as well as from the European Union for the Horizon 2020 project ALIGNED (GA-644055). Special thanks go to Marvin Hofer for his layouting support. Printed at Universitätsrechenzentrum Leipzig.

Links:

https://github.com/AKSW/meta-sparql
https://github.com/AKSW/meta-rdf

CONTACT

Johannes Frey frey@informatik.uni-leipzig.de, Sebastian Hellmann hellmann@informatik.uni-leipzig.de
Knowledge Integration and Linked Data Technologies (AKSW/KILT) research group
Leipzig University & Institute for Applied Informatics (InfAI), Leipzig, Germany.

HOSTED BY

SUPPORTED BY