Data integration and disintegration: Managing Springer Nature SciGraph with SHACL and OWL

Tony Hammond, Michele Pasin, and Evangelos Theodoridis

Springer Nature, The Campus, 4 Crinan Street, London N1 9XW, UK
{tony.hammond,michele.pasin,evangelos.theoridis}@springernature.com

We give an overview of the technical challenges involved in building a large-scale linked data knowledge graph, with a focus on the processes involving the normalization and control of data both entering and leaving the graph. In particular, we discuss how we are leveraging features of the Shapes Constraint Language (SHACL) [1] to combine closed-world, constrained views over an enterprise data integration setting with the open-world (OWL), unconstrained setting of the global semantic web.

About a year ago we began developing Springer Nature SciGraph (hereafter SciGraph) [2], our high-quality, linked data knowledge graph that describes the Springer Nature publishing world. SciGraph builds on various earlier projects [3] and collates information from across the research landscape, such as funders, research projects, conferences, affiliations and publications. In February 2017 we published a first release of the SciGraph dataset, consisting of metadata for journal articles from 2012–2016 and related research grants. Later this year we plan to release more historical publication data, this time including also books and chapters info. We are further planning to integrate additional data, such as citations, patents, clinical trials, usage numbers and linksets.

The data in SciGraph currently amounts to around 1–1.5 billion triples. We describe here the strategies we are adopting in order to scale up our ETL and data management capabilities to deal with actual and projected volumes.

We are using GraphDB [4] for our RDF storage. The GraphDB rulesets use an R-Entailment formalism that operates over our SciGraph core ontology that is expressed in OWL. We are making use of this mechanism to enrich our dataset using a couple of RDFS rules (to materialize range and domain types) along with some additional custom rules for simple compositions. This is a work in progress and we are still learning how to manage the data expansion ratios and to keep within a 'safe' maximum.

By contrast, SHACL provides an RDF language for applying constraints and rules to specified subgraphs that are described using SHACL 'shapes'. This gives us immediate access to arbitrary data patterns within the knowledge graph without having to make use of any complex or cumbersome OWL constraint machinery. We are using SHACL for three main purposes:

Data Validation. We have multiple ETL pipelines that bring in various entity types from various data sources. Entity types may be distributed across data sources, with SHACL shapes for each entity type specific to each data source. We assign each ETL pipeline to a specific RDF named graph and use a corresponding shapes graph

particular to that data graph. Each shapes graph constrains the list of entities and the properties for each entity that are allowed on that ETL pipeline.

Data Publishing. We are also making use of SHACL shapes as a direct replacement for our earlier data contracts work [5]. We use a separate set of export shapes for each customer. As the shapes are expressed in RDF we can use SPARQL to query over these export shapes to limit the entity types and the properties for each entity to publish for each customer.

Data Transformation. Finally, we are beginning to explore using SHACL for data transformations from our internal SciGraph model to other well-known models (e.g. schema.org) using one of the SHACL advanced features: rules. There are two main rule types: Triple rules and SPARQL rules. The former gives us a declarative means of specifying transformations from a source model to a target model and is an area that we are actively pursuing to foster a wider consumption of our data. While the SPARQL rules are generally more expressive, we feel that the Triple rules are more scalable from a maintenance point of view.

In conclusion, with SHACL we have improved our data quality and the integrity of our data products. We have improved the way we manage, develop and maintain multiple heterogeneous data flows both for ingest with multiple data producers and a weekly rebuild schedule, and for flexible data publishing to multiple data consumers.

There are two main challenges we are facing. One challenge is uniformity. As we are using GraphDB rules and SHACL rules/shapes, how can we better align these constructs with our OWL ontology? Ideally we would like to derive rulesets and shapes directly from the ontology as well as making use of a modular approach.

The other challenge is scalability. As the number of statements in the graph grows, our existing SHACL validator is adequate only for small datasets on single-node triplestores allowing us to validate entities at a given ingest time only. The goal would be to validate entities with properties aggregated over multiple ingests at the storage level. One possible direction for investigation is the use of federated SPARQL querying techniques in order to optimize query execution and SHACL validation on the whole graph by exploring distributed data stores using partitioning strategies [6].

References

1. Knublauch, H., Kontokostas, D., (Eds.): Shapes Constraint Language (SHACL). W3C Recommendation. 20 July 2017. https://www.w3.org/TR/shacl/.
2. Springer Nature SciGraph. http://www.springernature.com/scigraph.
3. Hammond, T., Pasin, M.: Linked data experience at Macmillan. http://data.nature.com/downloads/docs/iswc-2014-hammond-pasin-final.pdf.
4. Ontotext: GraphDB. http://graphdb.ontotext.com/graphdb/.
5. Hammond, T., Pasin, M.: The nature.com ontologies portal. http://data.nature.com/downloads/docs/iswc-2015-hammond-pasin-final.pdf.
6. Abbassi, S., Faiz, R.: RDF-4X: a scalable solution for RDF quads store in the cloud. In: Proceedings of the 8th International Conference on Management of Digital EcoSystems (MEDES), pp. 231-236. ACM, New York, NY, USA (2016). https://doi.org/10.1145/3012071.3012104.