This document aims to serve as input to the creation of a W3C community group that focuses on analyzing methods for accessing RDF data using GraphQL queries.

Group name

Bridging GraphQL and RDF

Short name

graphql-rdf

Group description

The aim of this group is to explore how GraphQL and RDF can be combined, and to what respect they can benefit each other.

This group explores possible combinations of GraphQL and RDF. We identify and compare existing approaches that bridge these worlds, collect use cases and requirements for such approaches, and characterize corresponding application areas. This will produce deliverables that may serve as input for one or more possible future standardization efforts.

Examples of application areas combining GraphQL and RDF are:

- Read/Write Access (CRUD) to and from RDF data via GraphQL queries and/or interfaces
- Validation of RDF graphs using GraphQL schemas
- Mapping between GraphQL and RDF-based shape languages

This group aims to produce the following deliverables:

- An overview of all known approaches that combine GraphQL and RDF. This overview will include a brief description of each approach, the company/organization that has developed it, and links to the corresponding documentation.
- An analysis of all known approaches that combine GraphQL and RDF. This analysis will include a categorization across one or more facets that will be identified.
- A final report on suggestions/possibilities for standardization.

TODO: Is the group limited in time? Or will it exist indefinitely?

Possible New Topics

(We don't yet know whether we'll participate in the CG, but here are some ideas to increase its scope ;-) --Vladimir Alexiev, Ontotext

Multi-storage architectures

Most significant apps use more than an RDF store, eg Lucene/SOLR/Elastic for FTS & faceting, Mongo for storing large amounts of data of which only some parts need to be RDFized, etc. Many RDF stores integrate to such additional stores, but that is not standardized (GraphDB Connectors, Virtuoso FTS and Jena Text all work differently). So how can you describe the shape of all this data in a unified model? GraphQL Federation can help you query several GraphQL endpoints in concert, but how can you dispatch GraphQL mutations to several stores?

Generating more artefacts

If you can map RDF to GraphQL, you can probably also generate JSONLD contexts, create shapes for validation etc. The mapping and flow between such artifacts is a very interesting topic: whether you use a dedicated schema description language, or can you use one artefact to generate some of the others… (eg SHEX from RDF examples)

Mapping between shape languages

Citing from https://github.com/w3c/EasierRDF/issues/64:
Shapes are important to describe business objects (entities) in RDF. There are several shape or shape-like languages:

- SHACL
- ShEx
- JSONLD Frames
- GraphQL

Each of these shape languages has its adherents and users, each has some strengths and weaknesses. It will be very useful for the community if some conceptual or technical mappings were created between the shape languages. This can help efforts such as:

- using shapes in one language to generate artefacts in another language, thus serving a different set of APIs and another user population. Eg from SHACL to JSONLD Frames
- converting shape libraries between languages
- generating shapes from editing tools or simpler representations
- shape visualizations

By no means am I claiming that shape languages are equivalent or have the same strength, or that lossless mappings are possible in all cases. Exploring the detailed correspondences between these technologies can be very interesting and fruitful. Not least because it can suggest extensions to some of the languages.
There are multi-shape implementations (Shaclex) that I believe have a lot to contribute to this discussion.