GraphQL vs SPARQL

GraphQL is a query language and server-side runtime (one has to run GraphQL first and then execute the code) for APIs of GraphQL-aware Web servers. The user writes a query in GraphQL structure (tree-based JSON format) and sends it to the API. The result of the query is obtained in the same format. A GraphQL schema constitutes of object types that tell what kind of object the user can request and what fields it has. When a query is received, GraphQL validates the queries against the schema. Then GraphQL executes the validated queries.

SPARQL is a query language for RDF graphs. Basically it can be used to retrieve or modify data represented by RDF. It is supported by many knowledge graphs like Wikidata and DBPedia. SPARQL makes use of RDF triples (subject, predicate, and object) by matching these patterns.

It is seen that many developers are more comfortable using GraphQL and its nested objects, rather than handling the RDF triples. Additionally, there are more libraries and frameworks that exist for GraphQL than SPARQL.

On the other hand, GraphQL is less expressive than SPARQL. It represent trees but not full graphs as in SPARQL. Also, GraphQL requires an interface specific schema. The query must validate against the schema set by the GraphQL server-side. So this makes it difficult to combine GraphQL data that comes from different sources. Lastly, GraphQL has no notion of global identifiers. Such identifiers exist in RDF as URIs that make the resources uniquely defined.

However, if we can use GraphQL on RDF graphs then we can simplify incorporating existing graph databases into web applications. It is an alternate to REST-based interfaces, and can be used to precisely define the data we want, replacing multiple REST requests with a single call.

GraphQL-LD and HypergraphQL

Since GraphQL cannot be used directly to query RDF graphs, there are 2 main approaches that make it possible:

1. **GraphQL-LD (Linked Data Querying with GraphQL)**: It enriches GraphQL queries with JSON-LD context to translate them (in the client side) into SPARQL queries. These queries are then accepted by RDF. JSON-LD provides a JSON syntax for RDF. It can be used to transform JSON documents to RDF.

   The results of the query can then be converted back in the tree structure JSON format that is expected by GraphQL. It is less expressive than SPARQL but can nevertheless achieve many data retrieval tasks that may be required by applications. Basically it does not use the GraphQL schema at all but exploits JSON-LD contexts to handle the conversion of GraphQL terms to RDF terms.

2. **HyperGraphQL:** It uses a different approach by exposing access to RDF sources using GraphQL queries. Basically it builds a GraphQL schema over an existing RDF graph. This is then accessible using GraphQL directly. The results are obtained as JSON-LD.

   HyperGraphQL requires a service to be set up that acts as an intermediary between the GraphQL client and the RDF sources**.** It uses a configuration file and an annotated GraphQL schema. The configuration file defines the RDF services form where the data will be fetched, and the schema is annotated with GraphQL directives that help to fetch the RDF service and the URIs.

   From the client perspective, the user sends plain GraphQL queries to an instance of HyperGraphQL. The returned result will be a GraphQL response enhanced with a JSON-LD context. The context can be used to transform the response into RDF.