Efficiently making technical decisions using knowledge graph

Sandeep Kunkunuru* [1,2]

[1]IIPH Hyderabad
[2]VaidhyaMegha Private Limited

* Correspondence: Sandeep Kunkunuru <hi18sk@iiphh.org>

Abstract

Knowledge graphs, semantic web and related topics have garnered considerable interest recently. Knowledge graphs are considered generally as a natural/intuitive form of knowledge representation and also as a queryable repository of knowledge. Several knowledge graphs have been built in several domains including medicine and its subdomains. On technical design especially technical decisions, the creative engine, of software products and solutions, very few attempts have been made. This paper presents methods and techniques to use a curated knowledge graph on technologies. Such that, answering questions sequentially, we can arrive at optimal design choices for every layer in software architecture. This paper demonstrates ways to implement and query this graph using SQL from command-line or SparQL from command-line or GraphQL using any API client tool ex: Postman or curl.

Introduction

A knowledge graph could be defined as below per [2]

```
...
Herein we adopt an inclusive definition, where we view a knowledge graph as a graph
of data intended to accumulate and convey knowledge of the real world, whose nodes
represent entities of interest and whose edges represent relations between these
entities. The graph of data (aka data graph) conforms to a graph-based data model,
which may be a directed edge-labelled graph, a propertygraph, etc
...
By knowledge, we refer to something that is known. Such knowledge may be accumulated
from external sources, or extracted from the knowledge graph itself.
```

The objective of this project was to curate a knowledge graph with dimensional and sometimes subjective facts of technologies at the heart and below feature list in addition :

Feature list

- Using GraphQL API knowledge graph can be queried using any API client tool ex: curl or Postman.

Sources

Methods

Knowledge graph will be designed as a directed-edge-labelled graph.

- It will be serialized as an adjacency list with below structure directly in a database table.

```
A question->a user chosen response-> Next question
A question->a user chosen response-> Final technical choice as a literal
```

- The choice of persisting graph as a database table, instead of RDF, was made to allow curation through an 'admin' interface.
- All entries in the adjacency table will be numeric ids. Each numeric id will in turn be a primary key in a different 'master' table.

Results

Querying knowledge graph using SparQL

Querying knowledge graph using GraphQL

Design for possibly bridges between GraphQL and RDF [3] datasets have been explored and implemented extensively since GraphQL [1] was originally published as an alternative API form.

Start server

From Postman

- With ntriples response

- With json response

Start client

In a separate terminal execute GraphQL query using curl (alternatively use Postman)

Discussions

This design utility framework can be extended for application code generation ex: through JHipster.

Acknowledgements

Declarations

Tables

References

[1] Byron, L. 2015. GraphQL: A data query language. *FACEBOOK Engineering, Core Data, Developer Tools*. (2015).

[2] Hogan, A. et al. 2021. Knowledge graphs. *Synthesis Lectures on Data, Semantics, and Knowledge*. 12, 2 (2021), 1–257.

[3] Taelman, R. et al. 2019. Bridges between graphql and rdf. *W3C workshop on web standardization for graph data. W3C* (2019).