TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science

Master Thesis

Querying Wikidata with GraphQL

Anas Shahab

Master Computational Logic

Matriculation number: 4827407

Supervisors:

Prof. Dr. Markus Krötzsch

Dr. Dörthe Arndt

Tutor:

Dipl.-Inf. Lukas Gerlach

Faculty of Computer Science

Institute of Theoretical Computer Science

Chair of Knowledge-Based Systems

Submission Date:

Declaration of originality

I hereby declare that I have written this Thesis on my own accord and any participation of others has been acknowledged. I have clearly marked all references to existing work. I have not submitted this work partly or as a whole anywhere else.

Dresden, XX.XX.XXXX

____

(signature)

Acknowledgements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Contents

1 Introduction

The term "knowledge graph" gained popularity in 2012 when Google launched its own *Google Knowledge Graph*. A knowledge graph is a collection of data represented as a graph. The collected data conveys knowledge of the real world, where the nodes represent entities of interest and edges the many different relations between those entities [3]. The entities are real world objects and abstract concepts. For example, "Helium has the chemical formula He", is a statement that can be represented using a knowledge graph. Here the nodes of the graph would represent "Helium" and "He", while the connecting edge between those nodes would represent the relation "chemical formula".

There are many ways of modelling data as a graph. The most commonly used ones are directed edge-labelled graphs, heterogeneous graphs, property graphs and graph dataset [3]. We will see in Section 2 how we can use RDF to specify directed edge-labelled graphs.

The term knowledge base is used synonymously with knowledge graph but there is a small difference [**?**]. Normally, a knowledge graph is viewed as a graphical structure. However, when defined semantically, it is considered to be a knowledge base for interpretation and inference from facts [**?**].

Many companies such as Amazon, Facebook, Uber, Google, etc., use knowledge graphs for their applications. Depending on the organization or community there are open or enterprise knowledge graphs [3]. Open knowledge graphs include Wikidata, DBpedia, Freebase, YAGO, etc. These are available online and freely accessible to the public. Enterprise knowledge graphs are generally used internally within a company and have their commercial specific use-cases [3].

The remainder of the report is structured as follows. In section 2, we provide an overview of RDF, Wikidata, SPARQL and GraphQL which will be crucial in the later sections. We also give a comparison between SPARQL and GraphQL. Section 3 gives the approaches used to query RDF graphs. The implementation of the approaches on Wikidata, along with the technicalities, and the differences in the SPARQL queries generated by the tools is provided in Section 4. The differences between the generated SPARQL queries and handwritten SPARQL queries is also shown here. This section also includes the performance and limitations of the tools. Lastly, we provide a conclusion and future work in Section 5.

Query language (only SPARQL) mentioned in the structure here

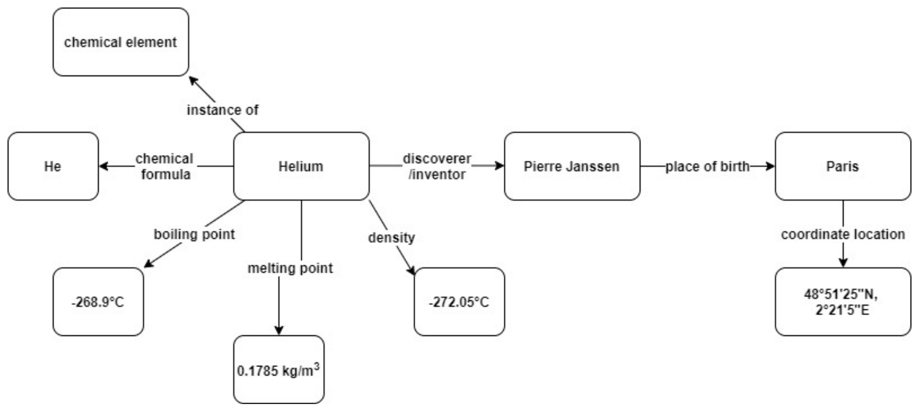

Figure 1: Directed edge-labelled graph describing Helium

2 Preliminaries

2.1 RDF

The Resource Description Framework (RDF) is a framework used to represent information available in the Web [4]. In the context of graphs, RDF is used for describing and exchanging graphs. The graphs specified by RDF are directed edge-labelled graphs. This means that the edges connect source nodes to target nodes, and have labels. It can be the case that there are multiple edges between the same nodes. However, these edges must have different labels. Figure 1 shows how knowledge about the chemical element Helium can be represented using directed edge-labelled graphs.

The graph shown in Fig 1 can be represented in an RDF graph. Formally, the building blocks of RDF graphs are IRIs, literals and blank nodes. They are defined as follows.

IRIs

To identify resources on the Web uniquely we use Uniform Resource Identifiers (URIs). An URI is a sequence of a subset of ASCII characters that have a scheme, authority, path, and optional query and fragment. An Internationalized Resource Identifier (IRI) is a generalized form of URI that helps to distinguish resources with Unicode. In RDF, an IRI is used as a name, or an equivalent of ID, for graph nodes [5]. Andy Seaborne has a good article that explains URIs and IRIs in detail [6].

RDF Literals

An RDF literal consists of three essential elements: a lexical value, a datatype IRI and an optional language tag. The lexical value is a string (footnote: RDF is based on Unicode strings) that corresponds to a particular literal value in the value space, where value space is the set of all possible values that a datatype can have. There are many datatypes in RDF some of which are string, Boolean, decimal and integer. A full list is available on the W3C's section on RDF datatypes: www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#section-Datatypes should I give as footnote?.

The datatype IRI refers to a datatype that defines which strings are valid (belong in the lexical space), the value space and the lexical-to-value mapping [**?**]. This mapping is essentially a function that maps each string from the lexical space to an element in the value space. The W3C standard XML Schema defines the datatypes and their IRIs. For example, decimals are identified by the IRI http://www.w3.org/2001/XMLSchema#decimal. W3C has a good documentation on the different XML Schema built-in datatypes [4].

The optional language tag helps to provide human-readable labels to RDF literals. A literal is a language-tagged string is of the form "string"@language (footnote: language is a well-formed language tag (after BCP47). The datatype IRI of such literals is http://www.w3.org/1999/02/22-rdf-syntax-ns#langString (footnote: It is never used in syntax).

RDF literals are used to represent resources that have values belonging to datatypes. Each literal can have only one datatype. For example, the boiling point of Helium would be a RDF literal represented as -268.98sd:decimal and its chemical formula as "He"@en, which is a language-tagged string.

change font

change font

Blank Nodes

Unlike an IRI or a literal, a blank node does not identify some specific resource [4]. It is used as a placeholder for some node, i.e., it is used to say that something with the given relationship exits at the position without specifying what the node is.

Definition 2.1 (RDF Graph)**.** An RDF graph is a set of triples that consists of the following elements:

- a subject (node) that is an IRI or a blank node;
- a predicate (edge) that is an IRI;
- an object (node) that is an IRI, a blank node, or a literal.

Figure 2 shows what an RDF graph could look like taking our example represented in Figure 1. We have added a label predicate to identify Helium. Our main interest is in querying the knowledge graph Wikidata, and so all the data correspond to the resources in its knowledge base. Literals are drawn as rectangular nodes. Here wd is a prefix for http://www.wikidata.org/entity/ and wdt for http://www.wikidata.org/prop/direct/. In Wikidata items (nodes) and properties

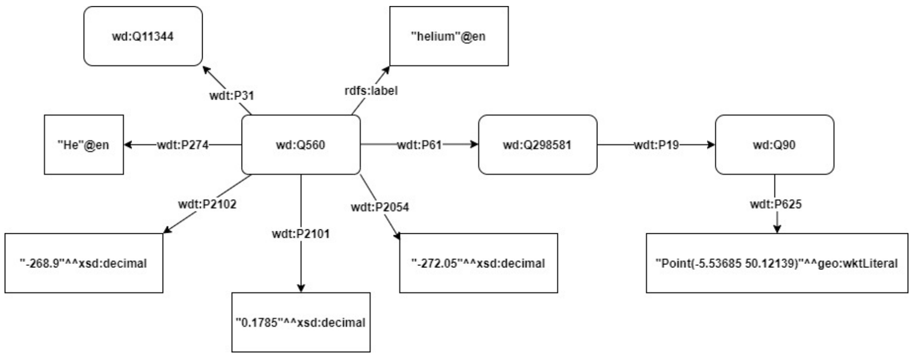

Figure 2: RDF graph describing Helium

(predicates) are identified uniquely by an entity ID. Table 1 helps to understand the RDF graph in Figure 2 better.

From Table 1 we can see that wd:Q560 is an abbreviation for the IRI http://www.wikidata.org/entity/Q560, where Q560 is the entity ID for Helium. The IRI helps is to identify the resource Helium on Wikidata. The rest of the nodes and edges in Figure 2 can be understood analogously. We can also see that for labelling we use rdfs:label which is used to give a human-readable version of a resource's name [7] . Also, geo:wktLiteral (http://www.opengis.net/ont/geosparql#wjktLiteral) is a datatype for representing geographic coordinates.

So far, we have shown an abstract syntax of RDF. For exchanging graphs, we need to have a concrete syntactic representation to encode RDF graphs [1]. The most relevant formats are N-Triples, Turtle, JSON-LD, RDF/XML and RDFa. W3C has provided a comprehensive ex-

Table 1: Abbreviation/IDs and their meanings.

wd	http://www.wikidata.org/entity/
wdt	http://www.wikidata.org/prop/direct/
rdfs	http://www.w3.org/2000/01/rdf-schema#
Q560	Helium
Q298581	discoverer/inventor
Q90	Paris
P31	instance of
P274	chemical formula
P2102	boiling point
P2101	melting point
P2054	density
P61	discoverer/inventor
P19	place of birth
P625	coordinate location

planation for each format mentioned here, along with other formats [8]. The RDF graph in Figure 2 can be represented in N-Triples format as follows:

```
<http://www.wikidata.org/entity/Q560> <http://www.wikidata.org/prop/direct/P31>
                                                <http://www.wikidata.org/entity/Q11344> .

<http://www.wikidata.org/entity/Q560> <http://www.w3.org/2000/01/rdf-schema#label>
                                                                          "helium"@en .

<http://www.wikidata.org/entity/Q560> <http://www.wikidata.org/prop/direct/P274>
                                                                              "He"@en .

<http://www.wikidata.org/entity/Q560>  <http://www.wikidata.org/prop/direct/P2102>
                                  "-268.9"^^<http://www.w3.org/2001/XMLSchema#decimal> .

<http://www.wikidata.org/entity/Q560> <http://www.wikidata.org/prop/direct/P2101>
                                  "0.1785"^^<http://www.w3.org/2001/XMLSchema#decimal> .

<http://www.wikidata.org/entity/Q560> <http://www.wikidata.org/prop/direct/P2054>
                                 "-272.05"^^<http://www.w3.org/2001/XMLSchema#decimal> .

<http://www.wikidata.org/entity/Q560> <http://www.wikidata.org/prop/direct/P61>
                                               <http://www.wikidata.org/entity/Q298581> .

<http://www.wikidata.org/entity/Q298581> <http://www.wikidata.org/prop/direct/P19>
                                                   <http://www.wikidata.org/entity/Q90> .

<http://www.wikidata.org/entity/Q90> <http://www.wikidata.org/prop/direct/P625>
         "Point(-5.53685 50.12139"^^<http://www.opengis.net/ont/geosparql#wjktLiteral> .
```

2.2 SPARQL

SPARQL Protocol and RDF Query Language (SPARQL) is a query language for RDF. It is based on matching graph patterns [9]. There can be different types of queries such as SELECT, ASK, CONSTRUCT and DESCRIBE. We will be talking about SELECT queries in this report. W3C provides a comprehensive document on SPARQL [10]. SELECT queries consist of the following blocks:

- Prologue: for declaring PREFIX and BASE;
- Select clause: SELECT keyword followed by either a list of variables and variable assignments, or by *;
- Where clause: WHERE keyword followed by a pattern
- Solution set modifiers: such as OFFSET [1].

To get a list of all chemical elements that has a chemical formula, boiling point, melting point, density, an inventor or discoverer, birth place of that inventor or discoverer and the coordinate location of the birth place we can a write a SPARQL query as follows:

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT ?element ?element_formula ?boiling_point ?melting_point ?density ?discoverer ?
    place_birth ?place_coordinate
WHERE {
  ?element wdt:P31 wd:Q11344.
  ?element wdt:P274 ?element_formula.
  ?element wdt:P2102 ?boiling_point.
  ?element wdt:P2101 ?melting_point.
  ?element wdt:P2054 ?density.
  ?element wdt:P61 ?discoverer.
  ?discoverer wdt:P19 ?place_birth.
  ?place_birth wdt:P625 ?place_coordinate.
}
```

We can run this query in Wikidata's query service (insert footnote about not having to use Prefix for wd and wdt since wikidata recognizes the abbreviations). Among the results, there will be also our element Helium (Q560) that we have used as an example in Figure 1 and Figure 2.

2.3 Wikidata

Wikidata is a free and publicly available open knowledge base. Editable by anyone, it acts as a central storage for the structured data for a variety of Wiki projects like Wikipedia, Wiktionary, Wikisource, etc [11]. It is more commonly known as the knowledge graph of Wikidata. Wikidata contains different data types like text, images, dates, etc., that can be queried using SPARQL [12]. There is a wide range of applications for Wikidata. We previously mentioned that Wikidata functions as a database for Wikimedia community like Wikipedia. According to the lecture given by Krötzsch [1], it also has external uses in many large organizations like Eurowings, Google, Apple and Amazon for tasks such as data integration, authority control, identity providing and data-driven journalism. The lecture also mentions that in the field of research, Wikidata is used for collecting test data for knowledge graph related algorithms and training data for machine learning projects. The easiest and most popular way to query Wikidata is through the Wikidata Query Service (WDQS). This is Wikidata's SPARQL endpoint. We can use this service two ways. Firstly, we can write queries in SPARQL directly on the web user interface of the service (footnote https://query.wikidata.org/) and obtain the results in different formats like table, tree, graph, etc. Secondly, the service can also be used pragmatically by submitting GET or POST requests (footnote https://query.wikidata.org/sparql) [13]. Another popular way to query Wikidata is by using the Wikidata API (footnote https://www.wikidata.org/wiki/Special:ApiSandbox). However, this API should mainly be used when we want to edit the contents of Wikidata or get data about entities like revision history. Wikidata dumps is useful when we know our result set will be significantly large or if we want to set up our own local query service. These dumps are full exports of all the available entities in Wikidata (footnote https://dumps.wikimedia.org/). To get started you should download the latest complete dump (footnote https://dumps.wikimedia.org/wikidatawiki/latest/). Wikidata also mentions some other ways to accessing Wikidata's data like Search and Linked Data Fragments endpoint, the complete list and usage of which can be found on Wikidata's Data Access webpage [13].

3 GraphQL

Stuff about GraphQL

3.1 GraphQL vs SPARQL

Stuff about GraphQL vs SPARQL

4 Literature Review

Stuff about GraphQL-LD and HypergraphQL and other tools do not have so many examples, and the ones that do are mainly for dbpedia.

5 GraphQL-LD

5.1 JSON-LD

Elaborate on concrete JSON-LD represetation

5.2 Using Graphql-LD with Wikidata

6 HypergraphQL

6.1 Schema

Elaborate on schema of HypergraphQL

6.2 Using HypergraphQL with Wikidata

7 GraphQL vs generated SPARQL queries

8 Differences between the generated SPARQL queries by both tools

9 Performance Evaluation

1. Evaluate performance of both approaches by writing SPARQL queries and equivalent GraphQL queries
2. Discuss in how far the generated Sparql queries differ from the handwritten ones.
3. analyze how well the approach scales with larger (deeper) GraphQL queries.

10 Effort for the setup

Evaluate/discuss the effort for the setup of both tools

11 Default context

12 Repository

13 Challenges faced and Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Bibliography

[1] Krötzsch, M.: Knowledge Graphs. Available at `https://iccl.inf.tu-dresden.de/web/Knowledge_Graphs_(WS2020)/en` 2022-10-01 (2020)

[2] Nonaka, I.: Takeuchi.(1995) the knowledge creating company. New York (1995)

[3] Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge Graphs. ACM Comput. Surv. **54**(4) (jul 2021). https://doi.org/10.1145/3447772, `https://doi.org/10.1145/3447772`

[4] R. Cyganiak, D.W., Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax. Available at `https://www.w3.org/TR/rdf11-concepts/` 2022-11-28 (2014)

[5] Tehnologies, O.S.: What is an IRI? (What does IRI mean?). Available at `https://www.oxfordsemantic.tech/fundamentals/what-is-an-iri-what-does-iri-mean` 2022-11-28 (2022)

[6] Seaborne, A.: RDF and IRI Syntax. Available at `https://afs.github.io/rdf-iri-syntax.html` 2022-11-28 (2020)

[7] Brickley, D., Guha, R.V.: RDF Schema 1.1. Available at `https://www.w3.org/TR/rdf-schema/` 2022-11-28 (2014)

[8] RdfSyntax. Available at `https://www.w3.org/wiki/RdfSyntax` 2022-12-04 (2011)

[9] E. Prud'hommeaux, A.S.: SPARQL Query Language for RDF. Available at `https://www.w3.org/2001/sw/DataAccess/rq23/` 2022-12-04 (2005)

[10] C. B. Aranda, O. Corby, S.D.L.F.P.G.B.G.S.H.S.H.I.H.N.h.N.M.C.O.M.P.A.P.A.P.E.P.A.S., Williams, G.T.: SPARQL 1.1 Overview. Available at `https://www.w3.org/TR/sparql11-overview/` 2022-12-04 (2013)

[11] Wikidata: Welcome to Wikidata. Available at `https://www.wikidata.org/wiki/Wikidata:Main_Page` 2022-12-04 (2019)

[12] Carpentry, L.: What is Wikidata? Available at `https://librarycarpentry.org/lc-wikidata/01-introduction/index.html` 2022-12-04 (2016)

[13] Wikidata: Wikidata:data access. Available at `https://www.wikidata.org/wiki/Wikidata:Data_access` 2022-12-04 (2022)