GEOYASGUI: THE GEOSPARQL QUERY EDITOR AND RESULT SET VISUALIZER

W. Beek[a,c,d], E. Folmer[a,b], L. Rietveld[a,c], J. Walker[a,e]

[a] Kadaster, Apeldoorn, Netherlands
[b] University of Twente, Twente, Netherlands (e.j.a.folmer@utwente.nl)
[c] Triply, Amsterdam, Netherlands
[d] VU University Amsterdam, Amsterdam, Netherlands (w.g.j.beek@vu.nl)
[e] Semaku, Eindhoven, Netherlands (john.walker@semaku.com)

Commission VI, WG VI/4

KEY WORDS: GeoSPARQL, IDE, Linked Open Data, Open Government, Semantic Web

ABSTRACT:

The Netherlands' Cadastre, Land Registry and Mapping Agency – in short Kadaster – collects and registers administrative and spatial data on property and the rights involved. This includes for ships, aircraft and telecommunications networks. Doing so, Kadaster protects legal certainty. The Kadaster publishes many large authoritative datasets including several key registers of the Dutch Government (Topography, Addresses and Buildings). Furthermore Kadaster is also developing and maintaining the PDOK shared service, in which about 100 spatial datasets are being published in several formats, including an incredible amount of detailed geospatial objects. Geospatial objects include all plots of land, all buildings, all roads and all lampposts. These objects are spatially and/or conceptually related, but are maintained by different data curators. As a result these datasets are syntactically and architecturally disjoint, and using them together currently requires non-trivial human labor.
In response to this, Kadaster is currently publishing its geo-spatial data assets as Linked Open Data. The standardized query language for Linked Open Geodata is GeoSPARQL. Unfortunately, current tooling does not support writing and evaluating GeoSPARQL queries. This paper presents GeoYASGUI, a GeoSPARQL editor and result-set viewer with IDE capabilities. GeoYASGUI is not a new software product, but an integration of and a collection of updates to existing Open Source libraries. With GeoYASGUI it becomes possible to query the rich Open Data assets of the Kadaster.

1. INTRODUCTION

The Netherlands' Cadastre, Land Registry and Mapping Agency – in short Kadaster[1] – is undertaking the ambitious development of an integrated approach towards publishing many of its data assets as Linked Open Data. Linked Data is used to integrate heterogeneous geospatial datasets, and make the available through the Kadaster Data Platform (https://data.pdok.nl). The primary query language for accessing such Linked Geospatial data is GeoSPARQL (Perry and Herring, 2012), a query language that is standardized by the OGC[2] and that extends the SQL-inspired SPARQL query language for RDF data (Garlik et al., 2013).

Needless to say, most of the SPARQL queries that are run over the enormous Kadaster Linked Data collection include at least several geospatial objects and at least some geo-spatial relations and/or functions. Unfortunately, there is no existing tooling that allows such GeoSPARQL queries to be edited, run, evaluated, and then re-edited and re-run. Several tools exist that support this kind of interaction for SPARQL, but these solutions do not deal with the idiosyncrasies of geo-spatial data (syntax) formats, and they do not support the geo-spatial extensions that GeoSPARQL adds to the SPARQL query language.

We present **GeoYASGUI**, a combined REPL for GeoSPARQL that reuses and extends existing Open Source libraries. We show how GeoYASGUI is used by the Kadaster to disseminate its Linked Open Data assets through a standards-compliant GeoSPARQL endpoint that is fully powered by Open Source software.

[1] See http://kadaster.nl
[2] See http://www.opengeospatial.org/ogc

The rest of this paper is structured as follows. In Section 2 we discuss our approach towards supporting the iterative process of query writing. Section 3 explains how our approach is concretely implemented in a collection of Open Source libraries. Section 4 sketches the context in which GeoYASGUI is being used by the Kadaster. Section 5 discusses the impact of (Geo)YASGUI and why we believe that the here presented implementation will result in uptake by the Linked Open Geodata community. Section 6 concludes.

2. APPROACH

Typical interaction with a (Geo)SPARQL endpoint follows the read-eval-print loop (REPL) principle. The endpoint first reads a query string sent by a client and parses it into an algebraic object (in a formalism akin to relational algebra). Queries are sent in a standards-compliant way, according to the SPARQL 1.1 Protocol specification (Feigenbaum et al., 2013). Secondly, the endpoint evaluates the query algebra against its data collection, preferable using indices that are pre-computed over the data and a heuristics-based query planner, in order to ensure a speedy query evaluation (Schmidt et al., 2010). Thirdly, the endpoint writes the results in a standardized result-set format. This includes standardized formats for XML, JSON, and CSV/TSV (Hawke et al., 2013, Seaborne et al., 2013, Seaborne, 2013).

A query typically does not return the intended result immediately. Indeed, writing queries is a form of declarative programming, and is an inherently iterative practice. As a result, the user has to go

through the query evaluation loop several times before the final query is written.

The problems with existing SPARQL query editors is that they provide only little support to the user. Even simple features like syntax checking, that are widely available for programming languages, are not broadly implemented in SPARQL endpoints. As a result, the user often has to go through multiple iterations just to make the query syntactically conforming. For this reason Triply[3] develops three products that support the use of SPARQL endpoints: **YASQE**, a query editor that provides direct feedback to the user; **YASR**, a versatile query results visualizer; and **YASGUI**, an integrated web service that combines YASQE and YASR (Rietveld and Hoekstra, 2017).

GeoSPARQL (Battle and Kolas, 2011) extends the SPARQL query language in several ways. Specifically, it adds the following components:

Core High-level schema for spatial objects (e.g., `geo:SpatialObjec`

Topology Vocabulary Properties for asserting and querying topological relations between spatial objects (e.g., `geo:sfIntersect`

Geometry RDFS datatypes that allow geometry data to be serialized, RDF properties for geometric relationships, and non-topological spatial query functions that can be applied to geometry objects.

Geometry Topology Topological query functions (e.g., `geof:relate`)

RDFS Entailment Calculates entailment results under the class hierarchy closure for geometries. For example, a WKT triangle is also a WKT polygon, surface, and geometry.

Query Rewrite Allows expressions that contain GeoSPARQL relations to be translated to and from expressions that contain GeoSPARQL functions.

3. IMPLEMENTATION

The here presented GeoYASGUI is not a new library that is built from scratch. Rather, it is a collection of updates to existing libraries, together with the necessary cross-library integration, that results in a GeoSPARQL editor with IDE-like capabilities. The existing libraries that are extended are YASQE (Section 3.1) and YASR (Section 3.2). In the following we give a detailed explanation of how these two libraries are updated to support GeoYASGUI.

3.1 GeoYASQE

YASQE (Figure 1) is a JavaScript library that, when added to a web page, takes native HTML text areas, and turns them into full-featured, IDE-like SPARQL query editors.

YASQE is based on the CodeMirror JavaScript library[4] for advanced HTML-based text editing. Using CodeMirror and the JavaScript SPARQL grammar from the Flint SPARQL Editor[5], YASQE is able to tokenize, highlight, validate, and dissect SPARQL queries. If needed, users are presented with immediate validation

[3] See `https://triply.cc`
[4] See `http://codemirror.net`
[5] See `http://openuplabs.tso.co.uk/demos/sparqleditor`

Figure 1. The YASQE query editor for GeoSPARQL. At the top of the query prefixes that are used in the rest of the query are declared. The projection clause (`select`) states that each result row consists of a shape (`?wkt`), a label (`?wktLabel`), and a color (`?wktColor`). The drop-down list shows auto-completion options for the property the user is currently typing.

errors of their queries, together with information about the type of validation error. In addition, YASQE provides several completion services, where completions are automatically suggested while typing. Firstly, because IRIs can be long and difficult to read, YASQE supports IRI prefix aliasing. Common alias-to-IRI mappings are retrieved from the Prefix.cc web service[6]. Secondly, properties and classes are auto-completed by using the Linked Open Vocabularies API (Vandenbussche et al., 2015).

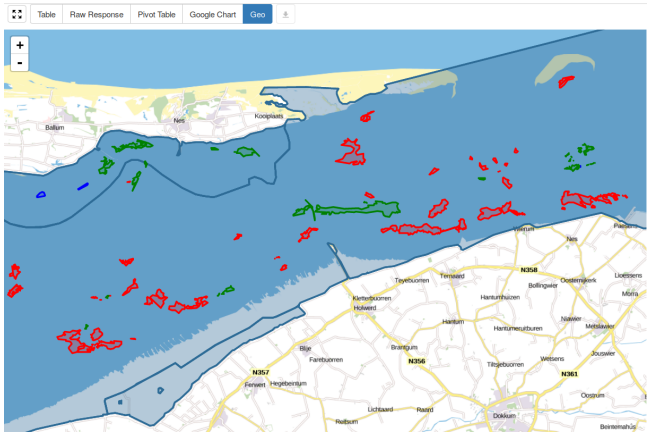
Because a user should not loose her work when a browser tab (accidentally) closes, YASQE uses HTML 5 functionality to store the application state. This makes the editing experience persistent between user sessions. As a result, a returning user will always see the screen as it was when the browser page was last closed.

YASQE was extended in the following ways to support the editing of GeoSPARQL queries:

- Prefix auto-completion mappings for GeoSPARQL namespaces were added. For example, `geof` aliases the IRI namespace `http://www.opengis.net/def/function/geosparql/`.
- Term auto-completion was extended to cover GeoSPARQL terms. This includes terms that denote geometric concepts (e.g., `geo:Geometry`), relationships (e.g., `geo:sfIntersects`), datatypes (e.g., `sf:Polygon`), functions (e.g., `geof:intersection`), and units of measure (e.g,, `uom:metre`).
- Syntax highlighting was extended to cover GeoSPARQL terms.
- A simple templating language was devised that allows geometric concepts to be styled. The templating language uses standard SPARQL 1.1 and GeoSPARQL constructs exclusively and links each shape variable (`?wkt`) to a label value (`?wktLabel`) with a particular color (`?wktColor`). Use of the templating language is optional.

[6] See `http://prefix.cc`

Figure 2. The YASR result-set viewer for GeoSPARQL. The map view shows WKT shapes that are displayed on a Leaflet map. The shapes denote areas where different protected species are living. The colors denote the kind of species that are living there. For example, red area only contain mussels, but green areas contain mussels and oysters.



3.2 GeoYASR

YASR (Figure 2) is a JavaScript library that parses and visualizes any SPARQL query response. The W3C specifies several SPARQL result formats, including XML, JSON, CSV, and TSV. To decrease the load on the publisher or developer, YASR consumes any of these data formats, by parsing the results and wrapping them in an internal data representation. A first parse attempt is based on the Media Type that is described by the `Content-Type` header of the HTTP response message. When this Media Type is missing or erroneous, YASR tries to parse the SPARQL results based on heuristics and on a best-effort basis.

YASR was extended in the following ways to support the visualization of GeoSPARQL result-sets:

- Literals with datatype IRI `geo:wktLiteral` are now parsed according to the Well-Known Text (WKT) grammar into (nested) JavaScript arrays. For this the Wicket library[7] for parsing WKT strings is used.
- A new map view was added that shows the parsed WKT geometries on a map. For this the popular web-based map widget Leaflet[8] is used. The map view supports the default tiles from Open Street Maps[9], but also allows the high-quality tiles of the Kadaster to be used instead.
- For label expressions in the YASQE templating language (Section 3.1), the map view displays markers at the centroid of shapes. Clicking a marker displays an HTML popup that shows additional content for the selected geometry.
- Similarly, color expressions in the YASQE templating language (Section 3.1) are used to style markers and polygons on the map. The color expressions are parsed using the Color[10] library, which supports the set of color definitions that are supported by the CSS standard.
- Geometry serializations, i.e., nested sequences of floating-point numbers, can be lengthy. Specifically, geometry serializations can be much longer than regular RDF data expressions. For this reason several tweaks need to be made, especially to the tabular views, in order to be able to conveniently display and browse result-sets that include geospatial terms.

[7] See `https://github.com/arthur-e/Wicket`
[8] See `http://leafletjs.com`
[9] See `https://www.openstreetmap.org`
[10] See `https://www.npmjs.com/package/color`

4. CONTEXT

The Kadaster[11] manages an enormous collection of heterogeneous datasets that describe every stationary geospatial object in the Netherlands in great detail. Geospatial objects include all plots of land, all buildings, all roads and all lampposts. These objects are spatially and/or conceptually related, but are maintained by different data curators. As a result these datasets are syntactically and architecturally disjoint, and using them together currently requires non-trivial human labor.

For these reasons, the Kadaster is now publishing its data assets as Linked Open Data. This makes it possible to query multiple datasets at once, without requiring query-specific and manual data integration. Datasets are published as RDF in according with the most recent standards and best practices, and by reusing existing Linked Vocabularies. In line with the GeoSPARQL standard, Linked Geospatial data is serialized as Well-Known Text (WKT).

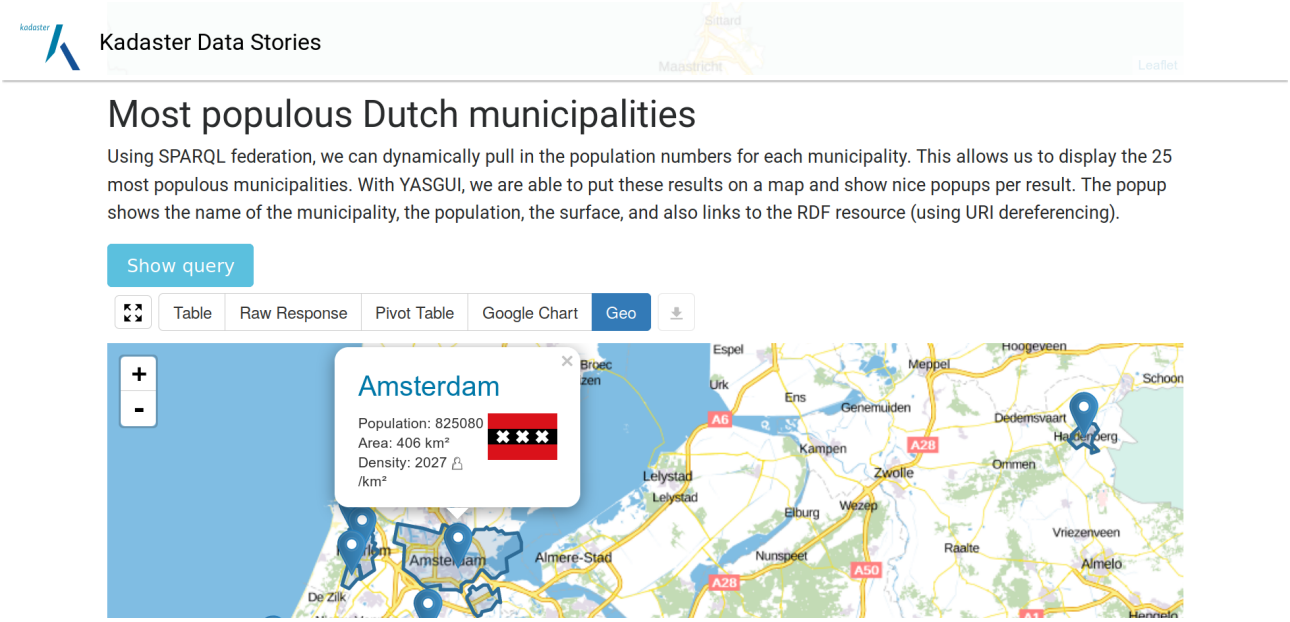
Figure 3 gives an example of how the integrated GeoYASGUI solution is currently being used by the Kadaster. The figure shows the YASR result-set visualizer in map mode. For GeoSPARQL results that are geometries, the WKT polygons of those geometries are displayed. Other properties that are part of the projection are displayed in a popup. In this example the popup includes the population of the selected municipality, together with its area and population density. Moreover, the title of the popup is a link that refers to the full RDF description of the resource that is served by the Linked Data Theater[12]. The user can also view the same result-set as a (regular) table, a pivot table, or a Google Chart (see the corresponding buttons in Figure 3). If the user knows GeoSPARQL, she can also click the “Show query” button, to ‘fold out’ the YASQE query editor. The combination of YASR and YASQE in the same widget is very powerful, because it allows a user to alter the query and immediately observe the results of running it.

5. IMPACT

YASGUI is already the most used SPARQL REPL. It is included in state-of-the-art triple stores like Apache Jena[13], OntoText GraphDB[14], Eclipse RDF4J[15], previously: OpenRDF Sesame), and ClioPatria (Wielemaker et al., 2016). In addition, YASGUI is used as a library in several data tools, such as Gosparqled[16], Snapper[17], Viso[18], Brwsr[19], and Trifid-LD[20]. Finally, many data publishers

[11] See `http://kadaster.nl`
[12] See `https://github.com/architolk/Linked-Data-Theatre`
[13] See `https://jena.apache.org`
[14] See `http://ontotext.com/products/graphdb/`
[15] See `http://rdf4j.org`
[16] See `https://github.com/scampi/gosparqled`
[17] See `https://github.com/jiemakel/snapper`
[18] See `https://github.com/jiemakel/visu`
[19] See `https://github.com/Data2Semantics/brwsr`
[20] See `https://github.com/zazukoians/trifid-ld`

Figure 3. Example of a GeoYASGUI widget that is used by the Kadaster. The widget appears in an HTML page about Dutch municipalities and their population size. The user can browse the data by clicking municipalities on the map. A popup displays the population size, area, and population density of each municipality (Amsterdam in this example). Clicking the name of the municipality traverses to the full RDF resource description for that municipality (implementing IRI dereferencing).



use it as the SPARQL entry point to their online data collection. This includes HealthData.gov[21], Smithsonian American Art Museum[22], German National Library of Economics[23], Linked Open Vocabularies[24], LOD Laundromat[25], MetaLex[26], and CEDAR[27].

Because GeoYASGUI integrates GeoSPARQL support into existing libraries that are already widely used, it will become available to will become available to a large number of users relatively quickly. A concrete example of this is the Swiss Federal Office of Topography[28] who are already using these GeoYASGUI features in their SPARQL endpoint.

6. CONCLUSION

The GeoYASGUI integration, the underlying libraries (YASQE, YASR), as well as the support services and libraries that are used are all published as Open Source code. (Geo)YASGUI, YASQE, and YASR are distributed under the MIT License. The Kadaster hosts a deployment of the GeoYASGUI web service through which its rich data assets can be queried. It currently exposes a data collection that consists of billions of triples and that describes tends of millions of geo-spatial objects. Please visit `https://data.pdok.nl/yasgui` to write your GeoSPARQL query and try out the GeoYASGUI query editor and result-set visualizer!

[21] See `http://www.healthdata.gov/sparql/`
[22] See `http://americanart.si.edu/collections/search/lod/about/sparql.cfm`
[23] See `http://zbw.eu/labs/en/blog/publishing-sparql-queries-live`
[24] See `http://lov.okfn.org/dataset/lov/sparql`
[25] See `http://lodlaundromat.org/sparql`
[26] See `http://doc.metalex.eu/query`
[27] See `http://lod.cedar-project.nl/cedar/data.html`
[28] See `https://ld.geo.admin.ch/sparql`

REFERENCES

Battle, R. and Kolas, D., 2011. GeoSPARQL: Enabling a geospatial Semantic Web. *Semantic Web Journal* 3(4), pp. 355–370.

Feigenbaum, L., Williams, G. T., Clark, K. G. and Torres, E., 2013. SPARQL 1.1 protocol. Recommendation, W3C.

Garlik, S. H., Seaborne, A. and Prud'hommeaux, E., 2013. SPARQL 1.1 query language. *World Wide Web Consortium*.

Hawke, S., Beckett, D. and Broekstra, J., 2013. SPARQL query results XML format (second edition). Technical report, W3C.

Perry, M. and Herring, J., 2012. OGC GeoSPARQL: A geographic query language for RDF data. *OGC Implementation Standard*.

Rietveld, L. and Hoekstra, R., 2017. The YASGUI family of SPARQL clients. *SWJ* 8(3), pp. 373–383.

Schmidt, M., Meier, M. and Lausen, G., 2010. Foundations of SPARQL query optimization. In: *Proc. of the 13th Int. Conf. on Database Theory*, ACM, pp. 4–33.

Seaborne, A., 2013. SPARQL query results CSV and TSV formats. Technical report, W3C.

Seaborne, A., Clark, K. G., Feigenbaum, L. and Torres, E., 2013. SPARQL query results JSON format. Technical report, W3C.

Vandenbussche, P.-Y., Atemezing, G. A., Poveda-Villalón, M. and Vatant, B., 2015. Linked open vocabularies (lov): a gateway to reusable semantic vocabularies on the web. *Semantic Web* pp. 1–16.

Wielemaker, J., Beek, W., Hildebrand, M. and van Ossenbruggen, J., 2016. ClioPatria: A SWI-Prolog infrastructure for the Semantic Web. *SWJ* pp. 529–541.