GeoSPARQL Query Tool
A Geospatial Semantic Web Visual Query Tool

Ralph Grove[1], James Wilson[2], Dave Kolas[3] and Nancy Wiegand[4]
[1]*Department of Computer Science, James Madison University, Harrisonburg, Virginia, U.S.A.*
[2]*Department of Integrated Science and Technology, James Madison University, Harrisonburg, Virginia, U.S.A.*
[3]*Raytheon BBN Technologies, Columbia, Maryland, U.S.A.*
[4]*Space Science and Engineering Center, University of Wisconsin, Madison, Wisconsin, U.S.A.*

Keywords: Semantic Web, Geographic Information Systems, GeoSPARQL, SPARQL, RDF.

Abstract: As geospatial data are becoming more widely used through mobile devices and location sensitive applications, the potential value of linked open geospatial data in particular has grown, and a foundation is being developed for the Semantic Geospatial Web. Protocols such as GeoSPARQL and stSPARQL extend SPARQL in order to take advantage of spatial relationships inherent in geospatial data. This paper presents GeoQuery, a graphical geospatial query tool that is based on Semantic Web technologies. GeoQuery presents a map-based user interface to geospatial search functions and geospatial operators. Rather than using a proprietary geospatial database, GeoQuery enables queries against any GeoSPARQL endpoint by translating queries expressed via its graphical user interface into GeoSPARQL queries, allowing geographic information scientists and other Web users to query linked data without knowing GeoSPARQL syntax.

1 INTRODUCTION

The Semantic Web has the potential to greatly increase the usability of publicly available data by allowing access to open data sets in linked format over the Web. W3C standards such as RDF (Manola 2004) and SPARQL (Prud'hommeaux, 2008) enable standard access to data stored in triple stores that are accessible over the Web at SPARQL endpoints. The Linked Open Data (Bizer, 2009) movement adds best practices for publishing data in order to maximize availability and usability.

As geospatial data are becoming more widely used through mobile devices and location sensitive applications, the potential value of linked open geospatial data in particular has grown, and a foundation is being developed for the Semantic Geospatial Web (Egenhofer, 2002). Protocols such as GeoSPARQL (Perry, 2010) and stSPARQL (Kyzirakos, 2012) extend SPARQL, the standard RDF Semantic Web query language, in order to take advantage of spatial relationships inherent in geospatial data.

In this paper we present the GeoSPARQL Query Tool (GeoQuery)[1], a graphical geospatial query tool that is based on Semantic Web technologies. GeoQuery translates queries expressed through its graphical user interface into GeoSPARQL queries, which can then be executed against any GeoSPARQL endpoint. With GeoQuery, geographic information scientists can query linked data and see map output without knowing GeoSPARQL syntax.

1.1 Motivation

The availability of linked open geospatial data and the Semantic Web will offer the potential for enhancing the value of geospatial data to the user in several ways.

- *Data Search and Discovery*: Semantic Web protocols could be used to dynamically discover and examine geospatial datasets over the Web, so that newly created or revised datasets will be of immediate value. The geospatial features of GeoSPARQL could also allow spatial operations to be incorporated into queries over metadata contained in catalogs of available open data during

[1] http://geoquery.cs.jmu.edu

a search.

- *Integration of Data Sets*: SPARQL provides the ability to integrate data services within queries, so that a user can perform logical queries of data from multiple servers without regard to the details of accessing multiple datasets and their respective formats. One query, for example, might perform spatial operations on data obtained from an open map service, a public repository, and a corporate data store.
- *Potential Applications of Semantic Web Technologies*: Beyond direct geospatial queries (e.g., show me what public buildings exist with the bounds of this city), techniques associated with the Semantic Web such as ontological reasoning and machine intelligence offer the potential for smarter user interfaces that can anticipate queries and integrate geospatial reasoning into emerging technologies such as automated vehicles and location aware phones.

The last ten years have seen tremendous growth in the development and utilization of the Internet in Geographic Information Systems (GIS). This growth is taking place in standard desktop software that is able to access geospatial data, maps, and geoprocessing services through the Internet, as well as web-based frontends to these same Internet based resources.

Currently, geospatial data are just starting to be represented in RDF (e.g., Varanka, 2012, and work by the Ordnance Survey), and the full potential of geospatial data available over the Web has yet to be realized. Open standards such as those developed by the Open Geospatial Consortium (OGC)[2] have played an important role in facilitating these developments. Most of these developments have been an evolution of existing GIS technologies, with only minor explorations into semantic technologies. The most successful developments have been in simplifying the access to distributed environments, allowing non-specialists to take advantage of mapping on the Internet. The small forays into geospatial semantics on the Internet have mostly been useable only by experts.

Conventional GIS allow users to explore, via a graphical user interface, datasets that are stored in proprietary or specialized data storage formats. Now, with linked data in RDF, the data representation format is open and standard. The GeoSPARQL query language is a new standard with which anyone can pose queries to data over a provided web-based endpoint. However, SPARQL and GeoSPARQL queries are not easy to write correctly without training. Prior attempts to overcome the difficulty of learning SPARQL include, for example, a visual SPARQL editor (Collustra, 2013). It might also be possible to approach this problem through a natural language interface to GeoSPARQL queries. However, natural language processing remains a difficult, not completely solved problem.

GeoQuery is a first step towards developing a universal query tool for the Geospatial Semantic Web. GeoQuery demonstrates that it is feasible to execute geospatial queries based upon the Semantic Web infrastructure using a graphical user interface. This ability is of value to geospatial professionals who need access to the data but are not trained in Semantic Web technologies. Because GeoSPARQL queries are viewable in GeoQuery, users can also learn about the GeoSPARQL language.

2 RELATED WORK

The Ordnance Survey in Great Britain pioneered Semantic Web work for geospatial data, including the use of linked data (e.g., The Linked Data Web, 2013) and the building of geospatial ontologies (e.g., Denaux et al., 2011). The Ordnance Survey held the first Terra Cognita geospatial workshop at the 2006 International Semantic Web Conference (ISWC) to add spatial data to the Semantic Web. The Spatial Ontology Community of Practice (SOCoP), along with others, have continued the series with the fifth one (Terra Cognita, 2012) being held with ISWC 2012.

In the United States, the Geological Survey (USGS) is doing work on linked data and ontologies (Geospatial Semantics and Ontology, 2013), including a recent workshop (Varanka, 2012). The USGS has translated some of The National Map data into RDF format. Transferring spatial data into RDF is a new area that the authors are also working on. Meanwhile, to handle nonspatial RDF data, leading database companies, such as Oracle and DB2, have added RDF storage and processing to their relational database systems e.g., (Das et al., 2004, Ma et al., 2008). Because query-processing of geospatial data in RDF is still new, Garbis et al. (2013) recently developed a benchmark to judge the performance of several RDF stores for geospatial querying. The database community is also interested in temporal aspects of the Semantic Web, and a bibliography has

[2] http://www.opengeospatial.org/

been compiled that also includes spatial references (Grandi, 2012).

Koubarakis et al. (2012) delineate areas of research for linked geospatial data, of which one area is user interfaces. They pose questions as to whether user interfaces should be based on natural language or be graphical, what high level APIs would ease rapid development, and whether interfaces could be built using existing platforms such as Google Maps, Bing Maps, or OpenStreetMap. In our work, we developed a graphical interface and used OpenStreetMap and Web Map Service for map display.

Our work uses the GeoSPARQL model as an extension to SPARQL. There is another spatial extension to SPARQL, stSPARQL, which is implemented in Strabon (Kyzirakos, 2012). Strabon extends Sesame, which has the ability to have PostGIS as a backend DBMS and spatial query processor. stSPARQL and GeoSPARQL do not overlap perfectly in functionality: stSPARQL includes aggregate functions and update capabilities (without which stSPARQL is a subset of GeoSPARQL), while GeoSPARQL includes an ontology and allows for topological relations as triples. A query language in addition to SPARQL and stSPARQL that incorporates spatial considerations is SPOTL (SPO + Time and Location) in the YAGO2 project (Hoffart et al., 2012). Time and location of facts are represented through reification.

3 SEMANTIC WEB TECHNOLOGIES FOR GEOSPATIAL DATA

Most early efforts to add geospatial data to the Semantic Web focused on very simple geospatial data, i.e., points represented by latitude and longitude. The W3C Geo ontology is popular for representing such points. Though this is sufficient for many domains and use cases, more complicated geospatial domains require the ability to use multiple coordinate systems and to store polygons and other shapes. This led to development of GeoSPARQL and its support in Parliament (see section 3.2).

3.1 GeoSPARQL

GeoSPARQL provides a unifying vocabulary for geospatial data on the Semantic Web. GeoSPARQL has two key parts: a small ontology for representing geospatial entities, and a set of query functions for processing relationships between the geospatial entities. The ontology is derived from well-used and well-understood concepts from the OGC and uses much of the same terminology as other OGC standards. The ontology is intentionally small so that it can be easily understood and easily attached to an appropriate domain ontology.

There are two key classes in the GeoSPARQL ontology: Feature and Geometry. A Feature is simply any entity (physical or abstract) with some spatial location. This could be a park, airport, monument, restaurant, etc. A Geometry is any geometric shape, such as a point, polygon, or line, and is used as a representation of a feature's spatial location. A third class, SpatialObject, is a superclass of both Feature and Geometry.

A Feature has only one primary property, *hasGeometry*. This property links the Feature to a Geometry that represents where it is in space. A Feature can have multiple Geometries, in which case it may specify one of these as the defaultGeometry to be used for spatial reasoning.

A Geometry has a number of properties, but the most important ones are those that relate the Geometry to a concrete spatial representation. These are *asWKT* and *asGML*, depending on whether the representation is in Well Known Text (WKT) (Open Geospatial Consortium, 2011) or Geography Markup Language (GML)[3] respectively. The properties point to an RDF literal with a data type of wktLiteral or gmlLiteral. Within these literals are the points that delineate the geometry: for example, the corners of a polygon.

The general usage of this ontology is to attach it to the ontology of the domain. If a domain ontology includes classes with relevant geospatial locations, those classes are declared subclasses of Feature. In this way they inherit the hasGeometry property and its link to the Geometry class.

The query functions in GeoSPARQL are used to relate the Features and Geometries to one another. The functions include binary topological relationships, set combinations of Geometries (ex. union, intersection), and other calculations such as distance. When possible, GeoSPARQL provides multiple sets of terminology for these functions. For example, the topological relations can be expressed in the terminology of the 9-intersection model (Egenhofer, 1990), RCC-8 (Randell, 1992), or OGC Simple Features (Open Geospatial Consortium, 2011). While implementations of GeoSPARQL do

[3] http://www.opengeospatial.org/standards/gml

not have to support all of these vocabularies, it is expected that most will. The binary Boolean topological relations can be expressed as either functions in a FILTER clause or as triples between SpatialObjects.

The following is an example of a GeoSPARQL query, which looks for monuments within parks, where both classes are subclasses of Feature. For more detailed examples see the GeoSPARQL specification (Perry, 2010) or (Battle, 2012). (The prefix "geo:" in this example refers to www.opengis.net/ont/geosparql, while the prefix "ex:" refers to an arbitrary example domain.)

```
SELECT ?m ?p
WHERE{
 ?m a ex:Monument ;
   geo:hasGeometry ?mgeo .
 ?p a ex:Park ;
   geo:hasGeometry ?pgeo .
 ?mgeo geo:within ?pgeo .
}
```

3.2 Parliament

Parliament[4] is an open-source RDF triple store (Figure 1). The outer layers are based on the open-source RDF toolkit Jena[5], which connects to a novel indexing scheme for RDF triples (Kolas 2009). Parliament provides a SPARQL endpoint for storing and querying RDF triples.

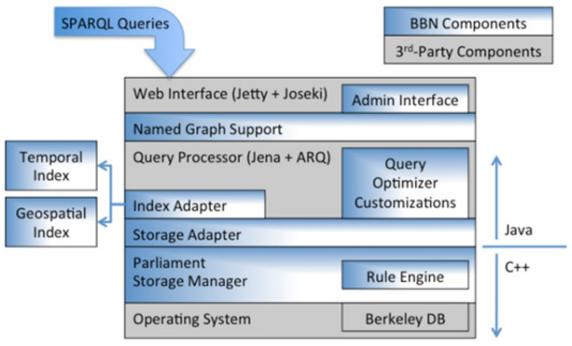


Figure 1: Parliament Architecture.

As the importance of the Geospatial Semantic Web grew, Parliament added spatial and temporal indexing. Initially this was a purely in-memory index designed as a proof of concept. It used OWL-Time[6] and an ontology based on GeoRSS[7] as the vocabularies for indexing. More recently the indices were updated to be persistent. The spatial index can use either an R-tree implementation or an external instance of Postgres. The temporal index uses Berkeley DB. The result is the ability to store and query spatial and temporal RDF efficiently. As the GeoSPARQL standard matured, Parliament was updated to include support for the standard.

4 GeoQuery

GeoQuery is being developed to provide GIS professionals an easy way to explore geospatial semantics in a familiar mapping interface, and to provide the ability to see how the semantics queries are actually built and executed.

The user interface is similar to many web-based mapping sites, where the user can turn layers on and off, and navigate around the map by panning and zooming. The interface also includes the ability to execute two separate queries using pick lists and text boxes, and to perform spatial operations on the results of the two queries. The onscreen areas for performing these operations have separate colors, and the results for each operation are displayed on the map with the same color to make it easy to identify the results for each operation.

Because one of the design goals of GeoQuery was to help explain GeoSPARQL, the full text of each GeoSPARQL query is saved and all of the query text can be viewed at any time.

The GeoSPARQL endpoint we are using for development is an instance of Parliament that has been populated with vector data extracted from the USGS National Map[8]. The data was extracted using the boundaries of the Shenandoah River (Virginia, USA) watershed, and was converted from an ESRI Personal Geodatabase to .N3 files using a custom tool developed by the USGS[9].

4.1 Design and Operation

Development of GeoQuery began by establishing a set of six general use cases that were gathered by examining example queries from literature. Development followed a prototyping process, the starting point for which was a tool for visualizing GeoSPARQL query results developed by the USGS. The ultimate user interface was developed with the

[4] http://parliament.semwebcentral.org/
[5] http://jena.apache.org/
[6] http://www.w3.org/TR/owl-time/
[7] http://www.georss.org

[8] See "Prodxucts and Services" at http://nationalmap.gov/index.html
[9] USGS NationalMap2RDF conversion tool: http://cegis.usgs.gov/ontology_userguide.html

intention of reproducing some controls and functions commonly found in existing GIS software.

Architecturally, GeoQuery is a web application that uses the web browser as its execution platform (Figure 2). As such, it is written entirely in HTML and JavaScript. The map display functionality is based upon OpenLayers[10], which provides a robust API for obtaining, displaying, and manipulating map tiles. JQuery provides a variety of coding shortcuts that reduce the overall amount of custom JavaScript required. The interface to the GeoSPARQL endpoint is based on Ajax and JSON. All GeoSPARQL queries are generated in JavaScript and then sent via Ajax to the Parliament server. Responses are returned in JSON format.

The current GeoSPARQL endpoint is built with Parliament, but GeoQuery should be compatible with any GeoSPARQL server.

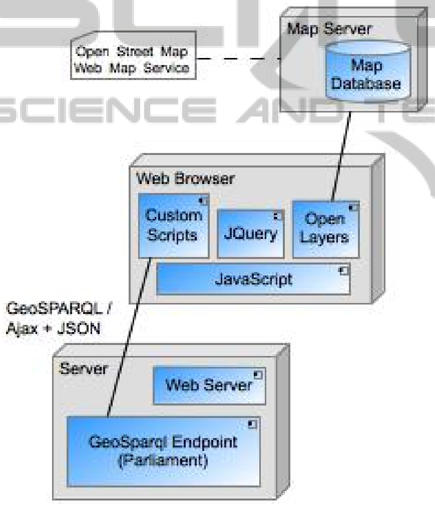


Figure 2: GeoQuery Architecture.

At startup, GeoQuery launches a predefined GeoSPARQL query to request map bounds (which are pre-loaded into the endpoint server) in order to select the initial map display. Other than this initialization, the system is fully event-driven. Each time the user launches a query or executes a spatial operation, the request is translated into GeoSPARQL and delivered via Ajax to the GeoSPARQL endpoint server. Query results, which consist of sets of spatial objects with WKT encoding, are returned as JSON objects, which must be decoded in order to be displayed on the map when appropriate. Some queries also return textual results, which are displayed in pop-up windows. No query optimization is performed in this version, but that would be an obvious next step in development.

[10] http://openlayers.org

4.2 User Queries

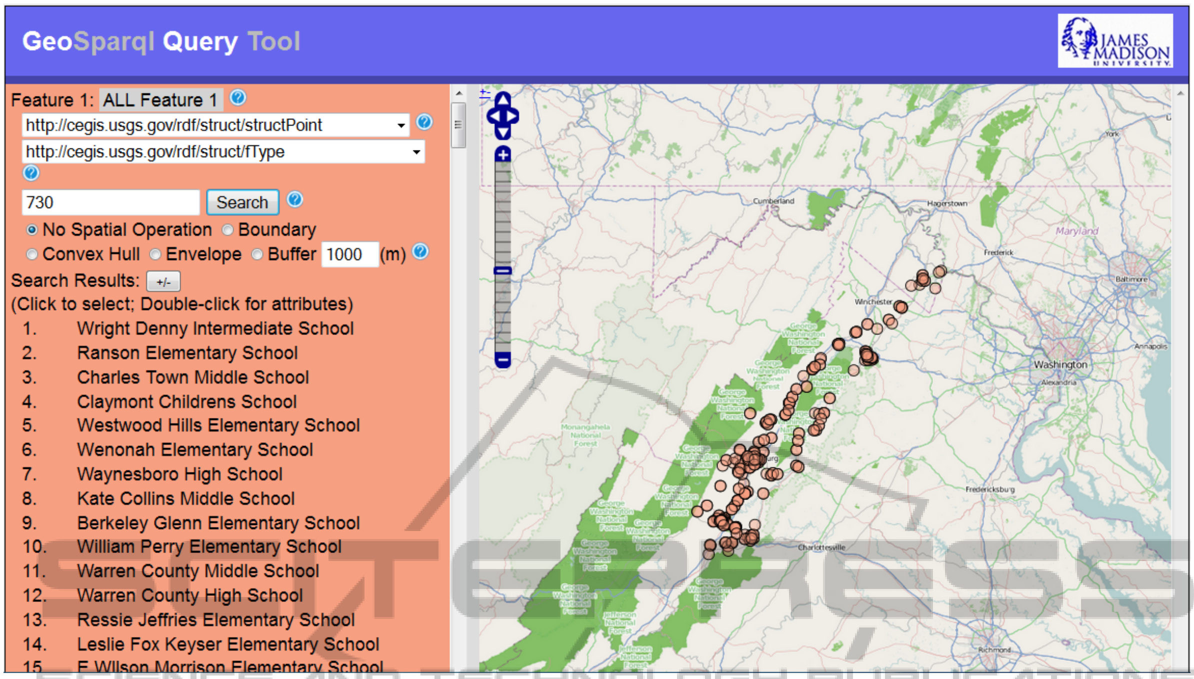
The user interface includes a map with basic navigation tools on the right side of the screen and user options for interacting with the data on the left side (Figure 3). The interface allows for two distinct queries to be executed, and for a spatial operation to be performed on the combined results.

For example, finding all of the schools that are present in a particular county (using the USGS National Map data) can be accomplished by defining the two queries and then applying a spatial operation.

First, to find all of the schools, the query can be defined in the Feature 1 query section on the user interface (Figure 3 - query area and results displayed in orange). In the USGS data, the point locations for different types of structures are stored in the class called *structPoint*, with different types of structures coded in a field called *fType*. Schools can be selected by selecting the feature type of *structPoint*, and the feature property *fType* with an *fType* value of 730 (Figure 3, Appendix: Query 1). Then, the Feature 2 query area can be used to select a particular county (query area and results displayed in cyan). Shenandoah County can be selected by choosing the feature type of *countyOrEquivalent*, then selecting to search on the label "Shenandoah" (Query 2). Invoking the Spatial Relationship tool (selection area and results displayed in purple) allows for any of the GeoSPARQL supported spatial operations to be applied to the results of the two searches, such as determining the schools that are within Shenandoah County (Figure 4, Appendix: Query 3).

Queries are created based on predefined patterns, using terms selected by the user from those derived from the data store. GeoQuery does not apply heuristics or make inferences from input, rather it responds in a straightforward way to user selections. Terms used to describe features and their properties are derived from the data store, and GeoQuery does not interpret or modify them. These abilities could be extensions to the tool in future versions.

Providing a complete interface to GeoSPARQL was not a design goal of this project. The set of query forms generated by GeoQuery is a small subset of the (infinite) set of query forms that could be generated in GeoSPARQL. Though more complex query options could be added to GeoQuery to extend the range of resulting queries, it is not


GeoSparql Query Tool

Feature 1: ALL Feature 1

http://cegis.usgs.gov/rdf/struct/structPoint

http://cegis.usgs.gov/rdf/struct/fType

730 Search

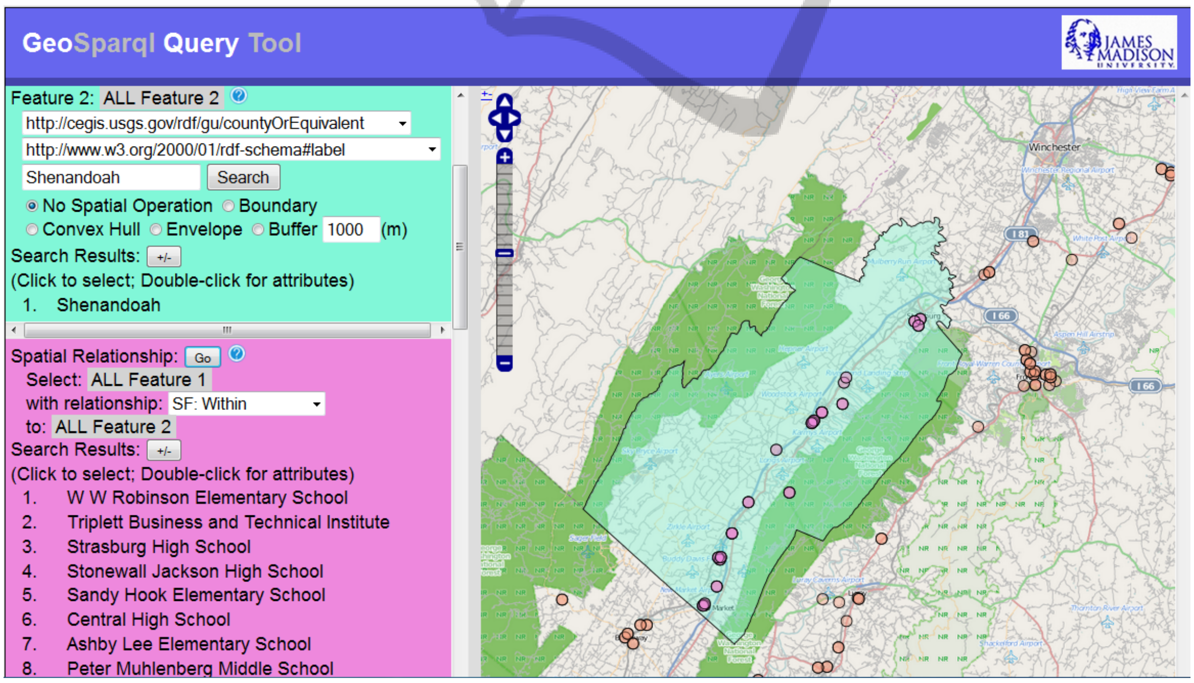
[✓ No Spatial Operation] Boundary Convex Hull Envelope Buffer 1000 (m)

Search Results: +/-

(Click to select; Double-click for attributes)

1. Wright Denny Intermediate School
2. Ranson Elementary School
3. Charles Town Middle School
4. Claymont Childrens School
5. Westwood Hills Elementary School
6. Wenonah Elementary School
7. Waynesboro High School
8. Kate Collins Middle School
9. Berkeley Glenn Elementary School
10. William Perry Elementary School
11. Warren County Middle School
12. Warren County High School
13. Ressie Jeffries Elementary School
14. Leslie Fox Keyser Elementary School
15. E Wilson Morrison Elementary School

Figure 3: Schools.


GeoSparql Query Tool

Feature 2: ALL Feature 2

http://cegis.usgs.gov/rdf/gu/countyOrEquivalent

http://www.w3.org/2000/01/rdf-schema#label

Shenandoah Search

[✓ No Spatial Operation] Boundary Convex Hull Envelope Buffer 1000 (m)

Search Results: +/-

(Click to select; Double-click for attributes)

1. Shenandoah

Spatial Relationship: Go

Select: ALL Feature 1

with relationship: SF: Within

to: ALL Feature 2

Search Results: +/-

(Click to select; Double-click for attributes)

1. W W Robinson Elementary School
2. Triplett Business and Technical Institute
3. Strasburg High School
4. Stonewall Jackson High School
5. Sandy Hook Elementary School
6. Central High School
7. Ashby Lee Elementary School
8. Peter Muhlenberg Middle School

Figure 4: Schools within Shenandoah County.

clear that completeness could be gained without exposing the user to elements of GeoSPARQL syntactic structure, which is counter to the design goals. Instead, we have provided an interface to support commonly used types of queries.

5 CONCLUSIONS

This work illustrates how a geospatial query tool can be successfully implemented based on Semantic

Web technologies such as RDF, SPARQL, and GeoSPARQL. Users can effectively query an RDF geospatial database over the Web, execute spatial operators on the results, and then visualize the results on a map in a familiar format, without knowing a formal query language. This is a first step towards bringing the value of the Semantic Web and open data to geospatial data and users. GeoQuery is an integral step in provided needed query access to the Geospatial Semantic Web.

The current GeoQuery tool is an initial proof of concept. The tool could be improved by replacing USGS National Map URIs and codes with more user-understandable synonyms. We used RDF data directly from The National Map and did not re-code it to be understandable by the general user. This is a limitation of using data directly converted to RDF, but adding definitions or links to ontologies is beyond the scope of this project. The innovation of our work is to take a new paradigm (RDF and GeoSPARQL) and make the data and querying accessible to any Web user using a graphical interface.

Testing to improve the tool could include formal user testing, testing of the tool against other SPARQL endpoints, testing with multiple endpoints simultaneously, and comparing its use against conventional tools. Additional extensions of the tool could include query optimization, the addition of more complex query forms through additional user interface options, and automatic clustering of results. We are also working on methods to automate converting general spatial data to RDF to make more spatial data accessible and to further test the tool.

ACKNOWLEDGEMENTS

A This work was partially supported by the National Science Foundation's Office of Cyberinfrastructure (OCI), INTEROP Grant No. 0955816.

Early versions of GeoQuery were based in part on a prototype query tool developed by USGS

REFERENCES

Battle, R., Kolas, D., 2012. Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL, *Semantic Web Journal* 3(4): 355-370.

Bizer, C., Heath, T., and Berners-Lee, T., 2009. Linked Data – The Story So Far, *International Journal on Semantic Web and Information Systems*, Special Issue on Linked Data, 5:1–22.

Collustra, 2013. (Online). Available: http://tw.rpi.edu/web/event/TWeD/2013/Fall/Collustra (30 Sep 2013).

Das, S., Chong, E. I., Eadon, G., Srinivasan, J., 2004. Supporting Ontology-based Semantic Matching in RDBMS, *Proceedings of the 30th VLDB Conference*, Toronto, Canada, pp. 1054-1065, 2004.

Denaux, R., Dolbear, C., Hart, G., Dimitrova, V., and Cohn, A., 2011. Supporting Domain Experts to Construct Conceptual ontologies: A Holistic Approach, *Web Semantics: Science, Services and Agents on the World Wide Web*, 9 (2011), pp. 113-127.

Egenhofer, M. J., and Herring, J. R., 1990. Categorizing binary topological relations between regions, lines, and points in geographic databases. *Technical report, Department of Surveying Engineering,* University of Maine.

Egenhofer, M. J., 2002. Toward the semantic geospatial web, *Proceedings of the ACM GIS02*, pp. 1-4.

Garbis, G., Kyzirakos, K., and Koubarakis, M., 2013. Geographica: A Benchmark for Geospatial RDF Stores, accepted at the *2013 International Semantic Web Conference*.

Geospatial Semantics and Ontology, 2013. Center for Excellence in Geospatial Science, [Online], Available: http://cegis.usgs.gov/ontology.html [15 Sep 2013].

Grandi, F., 2012. Introducing an Annotated Bibliography on Temporal and Evolution Aspects in the Semantic Web, SIGMOD Record, December 2012 (Vol. 41, No. 4), pp. 18-21, Available: http://www-db.deis.unibo.it/~fgrandi/TWbib/TSWbib.html (15 Sep 2013).

Hoffart, J., Suchanek, F. M., Berberich, K., and Weikum, G., 2012. YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia (preprint), (Online), Available: http://www.mpi-inf.mpg.de/yago-naga/yago/publications/aij.pdf (29 Sep 2013).

Kolas, D., 2009. Supporting Spatial Semantics with SPARQL, *Transactions in GIS, Vol. 12,* Issue s1, pp. 5-18, December 2008.

Koubarakis, M., Karpathiotakis, M., Kyzirakos, K., Nikolaou, C, and Sioutis, M., 2012. Data Models and Query Languages for Linked Geospatial Data, *Reasoning Web 2012*, LNCS 7487, T. Eiter and T. Krennwallner (Eds.) pp. 220-328, 2012.

Kyzirakos, K., Karpathiotakis, M., and Koubarakis, M., 2012. Strabon: A Semantic Geospatial DBMS, *Proceedings of the 11th International Semantic Web Conference* (ISWC 2012), Boston, USA, November 11th-15th.

The Linked Data Web, (Online), Available: http://www.ordnancesurvey.co.uk/education-research/research/linked-data-web.html (16 Sep 2013).

Ma, L., Wang, C., Lu, J., Cao, F., Pan, Y., Yu, Y., 2008. Effective and Efficient Semantic Web Data Management over DB2, Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1183-1194.

Manola, F., Miller, E. (Ed.s), 2004. RDF Primer, (Online), Available: http://www.w3.org/TR/rdf-primer (25 Dec 2013).

Open Geospatial Consortium, 2011. OpenGIS Implementation Standard for Geographic information - Simple feature access - *Part 1: Common architecture*, (Online), Available: http://www.opengeospatial.org/standards/sfa (25 Dec 2013).

Perry, M., Herring, J. (Eds.), 2010. OGC GeoSPARQL - A Geographic Query Language for RDF Data, Open Geospatial Consortium, (Online), Available: http://www.opengis.net/doc/IS/geosparql/1.0 (16 Sep 2013).

Prud'hommeaux, E. ,Seaborne, A. (Eds.), 2008. SPARQL Query Language for RDF, (Online), Available: http://www.w3.org/TR/rdf-sparql-query (25 Dec 2013).

Randell, D. A., Cui, Z., and Cohn, A. G., 1992. A spatial logic based on regions and connection. In Proceedings of the *3rd International Conference on Knowledge Representation and Reasoning*.

Terra Cognita, 2012. (Online), Available: http://iswc2012.semanticweb.org/workshops/TerraCognita.html (15 Sep 2013).

Varanka, D. (Ed.), 2012. *Introduction to Geospatial Semantics and Technology Workshop Handbook*, 2012 University Consortium for Geographic Information Science Symposium, (Online), Available: http://pubs.usgs.gov/of/2012/1109/ (15 Sep 2013).

APPENDIX

Query 1: select structPoint with fType=730

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?feature ?label
WHERE {
  # Select features of the specified type:
  ?feature rdf:type <http://cegis.usgs.gov/rdf/struct/structPoint> .
  ?feature rdfs:label ?label .
  # Filter features by property:
  ?feature <http://cegis.usgs.gov/rdf/struct/fType> ?obj .
  FILTER( regex(str(?obj), "730", "i" ) ) .
  # Eliminate the group of features ending in "/None"
  FILTER(! regex(str(?feature), "/None$", "i" ) ) .
}
```

………..Not showing individual queries to obtain geometry and map the features….

Query 2: Select & draw Shenandoah from countyOrEquivalent

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?feature ?label
WHERE {
  # Select features of the specified type:
  ?feature rdf:type
<http://cegis.usgs.gov/rdf/gu/countyOrEquivalent> .
  ?feature rdfs:label ?label .
  # Filter features by property:
  ?feature <http://www.w3.org/2000/01/rdf-schema#label> ?obj .
  FILTER( regex(str(?obj), "Shenandoah", "i" ) ) .
  # Eliminate the group of features ending in "/None"
  FILTER(! regex(str(?feature), "/None$", "i" ) ) .
}
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
SELECT ?wkt
WHERE {
  <http://cegis.usgs.gov/rdf/gu/Features/1673918>
geo:hasGeometry ?g .
  ?g geo:asWKT ?wkt .
}
```

Query 3: Schools within Shenandoah County

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX units: <http://www.opengis.net/def/uom/OGC/1.0/>
SELECT DISTINCT ?feature ?label
WHERE {
  # Feature 1:
  # Select features of the specified type:
  ?feature rdf:type <http://cegis.usgs.gov/rdf/struct/structPoint> .
  ?feature rdfs:label ?label .
  # Filter features by property:
  ?feature <http://cegis.usgs.gov/rdf/struct/fType> ?obj1 .
  FILTER( regex(str(?obj1), "730", "i" ) ) .
  # Eliminate the group of features ending in "/None"
  FILTER(! regex(str(?feature), "/None$", "i" ) ) .
  ?feature geo:hasGeometry ?g1 .
  ?g1 geo:asWKT ?wkt1 .
  # Feature 2:
  # Select features of the specified type:
  ?feature2 rdf:type
<http://cegis.usgs.gov/rdf/gu/countyOrEquivalent> .
  # Filter features by property:
  ?feature2 <http://www.w3.org/2000/01/rdf-schema#label> ?obj2
.
  FILTER( regex(str(?obj2), "Shenandoah", "i" ) ) .
  # Eliminate the group of features ending in "/None"
  FILTER(! regex(str(?feature2), "/None$", "i" ) ) .
  ?feature2 geo:hasGeometry ?g2 .
  ?g2 geo:asWKT ?wkt2 .

  # spatial relationship
  FILTER (geof:sfWithin(?wkt1, ?wkt2)) .
}
```

(Note: Some of the queries contain a filter term intended to eliminate features ending in "/None". These are features that have incomplete definitions, an anomaly of the test dataset that was used.)