Getting Started with GeoSPARQL

Dave Kolas
Matt Perry & John Herring

May 7 2013

The OGC: Making Location Count
Outline

• Introduction to GeoSPARQL
• GeoSPARQL Vocabulary
• GeoSPARQL Query Relations
• Worked Example:
  – Ontology
  – Instance data
  – Query
• Conclusion
Purpose of GeoSPARQL

• Many problems for which semantic solutions are relevant have an inherent geospatial context
  – Which hospitals within 20 miles have appropriate treatment centers for my patient?
  – What airports within 50 miles of a mission objective can support a C5?

• In order to efficiently perform geospatial reasoning, special indexing is required
  – We cannot take away the ability to do semantic reasoning though
GeoSPARQL is a minimal RDF/OWL/SPARQL vocabulary for storage and query of geospatial information

- Should be able to be easily attached to ontologies with a need for spatial information
- Represents *only* geometries and the concept of a Feature (a thing with a geometry) and the geospatial relationships between them

Result: triple store implementations can spatially index information in the vocabulary, and perform spatial reasoning

GeoSPARQL intends to be:
- Robust enough to be used for ‘serious’ geospatial data
- Simple enough for Linked Open Data
• W3C Recommendation for querying RDF/OWL
• Syntax based on RDF/Turtle

PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE {
    OPTIONAL {
        ?x ns:price ?price .
        FILTER (?price < 30)
    }
}
• Uses W3C’s SPARQL’s extensibility framework
• Contains a shared vocabulary
  – Feature Model and Geometry Model
  – Minimal set of classes and properties supporting triple patterns
• Contains datatypes based on text-based geometry serializations: WKT (augmented for CRS); GML
  – geo:wktLiteral, geo:gmlLiteral, …
  – Different conformance class for each serialization
• Contains shared set of FILTER functions
  – For each Simple Features function
• Contains Query Rewrite Rules for spatial relations between Features — both explicit (triples) and implicit (geometry)
GeoSPARQL vs W3C Geo

• Unlike GeoSPARQL, W3C Geo:
  – Only supports point geometries
  – Only supports one coordinate reference system
  – Has no spatial relationships

• Have data in W3C Geo format?
  – No problem. One query converts W3C Geo into GeoSPARQL information (see paper “Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL”)

• W3C Geo is very simple and useful, but GeoSPARQL offers significantly more functionality
GeoSPARQL is a foundation for spatial domain ontologies.
Parts of GeoSPARQL

Ontology
- Features
- Geometries
- Geometry Descriptions

Domain Ontology

Instance Data

Query Terms
- Topological Relation Properties
- Filter Functions

SPARQL Queries
Ontology - Basic Classes

Explicitly typed as OWL classes and RDFS classes to accommodate both types of systems

geo:SpatialObject

geo:Feature

d:Park
d:City
d:Monument

geo:Geometry

Domain Classes
Datatype Properties for geo:Geometry

- Explicitly typed as owl:DatatypeProperty and rdf:Property
  - geo:dimension
  - geo:coordinateDimension
  - geo:spatialDimension
  - geo:isEmpty
  - geo:isSimple
  - geo:is3D

- geo:asWKT
- geo:asGML

Only one of these -- based on the conformance class

- Implementations may do both.
Datatype for Geometry Serialization

• Serialization defines the conformance class
• Initially conformance classes for WKT and GML
  – GML is used as-is for geo:gmlLiteral
  – Spatial Reference System Identifier (SRID) URI is added for geo:wktLiteral

```
"<http://www.opengeospatial.org/def/crs/EPSG/0/4326>
  Point(-83.38 33.95)"^^geo:wktLiteral

"Point(33.95 -83.38)"^^geo:wktLiteral
```

(Default CRS is http://www.opengeospatial.org/def/crs/OGC/1.3/CRS84)
Topological Spatial Relations

Can be used in two ways:
- A property in the triple pattern
- A filter function

**Simple Features**
- geo:sfEquals
- geo:sfDisjoint
- geo:sfIntersects
- geo:sfTouches
- geo:sfWithin
- geo:sfContains
- geo:sfOverlaps

**9-IM**
- geo:ehEquals
- geo:ehDisjoint
- geo:ehMeet
- geo:ehOverlap
- geo:ehCovers
- geo:ehCoveredBy
- geo:ehInside
- geo:ehContains

**RCC8**
- geo:rcc8Eq
- geo:rcc8dc
- geo:rcc8po
- geo:rcc8tppi
- geo:rcc8tp
- geo:rcc8ntpp
- geo:rcc8ntppi
Transformational FILTER Functions

- geof:relate
- geof:distance
- geof:buffer
- geof:convexHull
- geof:intersection
- geof:union
- geof:difference
- geof:symDifference
- geof:envelope
- geof:boundary
Implementations

• BBN Parliament
  – http://parliament.semwebcentral.org/
  – Open Source triplestore
  – Jena/Joseki front end

• Oracle Database

• Strabon
  – http://www.strabon.di.uoa.gr/about
Example Ontology

Feature

PointOfInterest

Attraction

Monument  Park  Museum

Service

Restaurant
Example Ontology

@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix ex: <http://www.example.org/POI#> .

ex:Restaurant a owl:Class;
   rdfs:subClassOf ex:Service .
ex:Park a owl:Class;
   rdfs:subClassOf ex:Attraction .
ex:Museum a owl:Class;
   rdfs:subClassOf ex:Attraction .
ex:Monument a owl:Class;
   rdfs:subClassOf ex:Attraction .
ex:Service a owl:Class;
   rdfs:subClassOf ex:PointOfInterest .
ex:Attraction a owl:Class;
   rdfs:subClassOf ex:PointOfInterest .
ex:PointOfInterest a owl:Class;
   rdfs:subClassOf geo:Feature .
Example Instance Data
Example Instance Data

@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix ex: <http://www.example.org/POI#> .
@prefix sf: <http://www.opengis.net/ont/sf#> .

ex:WashingtonMonument a ex:Monument;
  rdfs:label "Washington Monument";
  geo:hasGeometry ex:WMPoint .
ex:WMPoint a sf:Point;
  geo:asWKT "POINT(-77.03524 38.889468)"^^geo:wktLiteral.

ex:NationalMall a ex:Park;
  rdfs:label "National Mall";
  geo:hasGeometry ex:NMPoly .
ex:NMPoly a sf:Polygon;
  geo:asWKT "POLYGON((-77.050125 38.892086, -77.039482 38.892036, -77.039482 38.895393, -77.033669 38.895508, -77.033585 38.892052, -77.031906 38.892086, -77.031883 38.887474, -77.050232 38.887142, -77.050125 38.892086 ))"^^geo:wktLiteral.
Example Query

• Which monuments are spatially within which parks?

```sparql
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX ex: <http://www.example.org/POI#>

SELECT ?m ?p
WHERE {

  ?m a ex:Monument ;
  geo:hasGeometry ?mgeo .

  ?p a ex:Park ;
  geo:hasGeometry ?pgeo .

}
```
Tip of the Iceberg

• This is the simplest of simple examples
• GeoSPARQL is about adding geospatial reasoning to Semantic Web applications
  – Most domain ontologies are much richer than this example, enabling interesting reasoning
  – Data integration applications can have a geospatial component

• Some potential queries:
  – Which hospitals within 20 miles have appropriate treatment centers for my patient? (reasoning)
  – What hotels with 3 star ratings are within 10km of at least 3 attractions with 4 star ratings? (data integration)
Conclusion

• Do you need to use geospatial locations in your ontology?
  – Don’t reinvent, use GeoSPARQL.

Questions?