The GeoSPARQL OGC Standard

Matthew Perry

5th International Terra Cognita Workshop 2012
In Conjunction with the 11th International Semantic Web Conference
November 12, 2012
Boston, MA
Agenda

• About the GeoSPARQL SWG
• Use Cases & Requirements
• GeoSPARQL Technical Details
• Implementation Considerations
• Live Demos
  – BBN Parliament (Dave Kolas)
  – Strabon (Kostis Kyzirakos)
Group Members

- Open Geospatial Consortium standards working group
  - 13 voting members, 36 observers
  - Editors: Matthew Perry and John Herring
  - Chairs: John Herring and Dave Kolas

- Submitting Organizations

  - Australian Government Bureau of Meteorology
  - Bentley
  - CSIRO
  - Interactive Instruments
  - Defence Geospatial Information Working Group
  - GeoConnections
  - Oracle
  - USGS
  - Ordnance Survey
  - Raytheon

  Traverse Technologies, Inc.
Standardization Process

- Form SWG (June 2010)
- OAB vote on candidate standard (June 2011)
- Process comments and update document (Feb. 2012)
- Standard Published (June 2012)

- Release candidate standard (May 2011)
- 30-day public comment period (July 2011)
- TC/PC vote (March 2012)
Implementations

Parliament™

strabon.di.uoa.gr

Oracle®
DATABASE

Open SAHARA | uSeekM
SOME USE CASES FOR GEOSPARQL
Linked Geo Data

• Many LOD datasets have geospatial components

• Barriers to integration
  – Vendor-specific geometry support
  – Different vocabularies
    • W3C Basic Geo, GML XMLLiteral, Vendor-specific
  – Different spatial reference systems
    • WGS84 Lat-Long, British National Grid

What DBPedia Historic Buildings are within walking distance?

What OpenStreetMap Dog Parks are inside Ordnance Survey Southampton Administrative District?
• GIS applications with semantically complex thematic aspects
  – Logical reasoning to classify features
    • land cover type, suitable farm land, etc.
  – Complex Geometries
    • Polygons and Multi-Polygons with 1000’s of points
  – Complex Spatial Operations
    • Union, Intersection, Buffers, etc.

Find parcels with an area of at least 3 sq. miles that touch a local feeder road and are inside an area of suitable farm land.
Gazetteers and Linked Open Data Services

- Provide common terms (place names) to link across existing spatial data resources
- Enable consolidated view across the map layers
- Reconcile differences in data semantics so that they can all “talk” and interoperate
- Resolving semantic discrepancies across databases, gazetteers, and applications
- Integrate full breath of enterprise content continuum (structured, spatial, email, documents, web services)
Towards Qualitative Spatial Reasoning

• Don’t always have geometry data
  – Textual descriptions
    • Next to Hilton hotel
    • Inside Union Square
  – Incomplete geometry data
    • Only have geometries for some features
    • Hybrid quantitative and qualitative spatial reasoning

• GeoSPARQL takes some steps in this direction
  – Vocabulary for asserting topological relations
  – Same query specification for qualitative and quantitative systems
Requirements for GeoSPARQL

• Provide a common target for implementers & users
  – Representation and query
• Work within SPARQL’s extensibility framework
• Simple enough for general users
  – Keep the common case simple (WGS 84 point data)
• Capable enough for GIS professionals
  – Multiple SRSs, complex geometries, complex operators
• Don’t re-invent the wheel!

ISO 19107 – Spatial Schema
ISO 13249 – SQL/MM

Simple Features
Well Known Text (WKT)
GML
KML
GeoJSON
FROM SPARQL TO GEOSPARQL
**SPARQL Query**

**RDF Data**

```
:res1 rdf:type :House .
:res1 :baths "2.5"^^xsd:decimal .
:res1 :bedrooms "3"^^xsd:decimal .

:res2 rdf:type :Condo .
:res2 :baths "2"^^xsd:decimal .
:res2 :bedrooms "2"^^xsd:decimal .

:res3 rdf:type :House
:res3 :baths "1.5"^^xsd:decimal .
```

**SPARQL Query**

```
SELECT ?r ?ba ?br
 ?r :bedrooms ?br }
```

**Result Bindings**

```
?r | ?ba | ?br
-----
:res1 | "2.5" | "3"
:res3 | "1.5" | "3"
```
RDF Data

```
:res1 rdf:type :House .
:res1 :baths "2.5"^^xsd:decimal .
:res1 :bedrooms "3"^^xsd:decimal .
:res2 rdf:type :Condo .
:res2 :baths "2"^^xsd:decimal .
:res2 :bedrooms "2"^^xsd:decimal .
:res3 rdf:type :House
:res3 :baths "1.5"^^xsd:decimal .
```

SPARQL Query

```
SELECT ?r ?ba ?br
WHERE {
  ?r :bedrooms ?br
  FILTER (?ba > 2)
}
```

Result Bindings

```
?r | ?ba | ?br
:res1 | "2.5" | "3"
```
Spatial RDF Data

```sparql
:res1 rdf:type :House .
:res1 :baths "2.5"^^xsd:decimal .
:res1 :bedrooms "3"^^xsd:decimal .
:res1 ogc:hasGeometry :geom1 .
:geom1 ogc:asWKT "POINT(-122.25 37.46)"^^ogc:wktLiteral .

:res3 :baths "1.5"^^xsd:decimal .
:res3 ogc:hasGeometry :geom3 .
:geom3 ogc:asWKT "POINT(-122.24 37.47)"^^ogc:wktLiteral .
```

GeoSPARQL Query

```
SELECT ?r ?ba ?br
    ?r ogc:hasGeometry ?g .
    ?g ogc:asWKT ?wkt
    FILTER(ogcf:sfWithin(?wkt, "POLYGON(...)"^^ogc:wktLiteral)) }
```
GEOSPARQL TECHNICAL DETAILS
Components of GeoSPARQL

• Vocabulary for Query Patterns
  – Classes
    • Spatial Object, Feature, Geometry
  – Properties
    • Topological relations
    • Links between features and geometries
  – Datatypes for geometry literals
    • ogc:wktLiteral, ogc:gmlLiteral

• Query Functions
  – Topological relations, distance, buffer, intersection, …

• Entailment Components
  – RDFS entailment
  – RIF rules to compute topological relations
GEOSPARQL VOCABULARY
GeoSPARQL Vocabulary: Basic Classes and Relations

- ogc:SpatialObject
  - ogc:hasGeometry 0 .. *
  - ogc:hasDefaultGeometry 0 .. 1

- ogc:Feature
  - Same as ISO GFI_Feature

- ogc:Geometry
  - metadata
    - ogc:dimension : xsd:int
    - ogc:coordinateDimension : xsd:int
    - ogc:spatialDimension : xsd:int
    - ogc:isEmpty : xsd:boolean
    - ogc:isSimple : xsd:boolean
  - serializations
    - ogc:asWKT : ogc:wktLiteral
    - ogc:asGML : ogc:gmlLiteral
    - ...

- Same as ISO GM_Object
  - Geometry encoded as a Literal
All RDFS Literals of type `ogc:wktLiteral` shall consist of an optional IRI identifying the spatial reference system followed by Simple Features Well Known Text (WKT) describing a geometric value [ISO 19125-1].

```
"<http://www.opengis.net/def/crs/OGC/1.3/CRS84> POINT(-122.4192 37.7793)"^^ogc:wktLiteral
```

WGS84 longitude – latitude is the default CRS

```
"POINT(-122.4192 37.7793)"^^ogc:wktLiteral
```

European Petroleum Survey Group (EPSG) maintains a set of CRS identifiers.
All ogc:gmlLiterals shall consist of a valid element from the GML schema that implements a subtype of GM_Object as defined in [OGC 07-036].

"<gml:Point
  srsName="http://www.opengis.net/def/crs/OGC/1.3/CRS84"
  xmlns:gml="http://www.opengis.net/gml">
  <gml:pos>-83.38 33.95</gml:pos>
</gml:Point>"^^ogc:GMLLiteral

Note that gmlLiterals are NOT rdf:XMLLiteral
Topological Relations between ogc:SpatialObject

- ogc:sfEquals
- ogc:sfTouches
- ogc:sfOverlaps
- ogc:sfContains
- ogc:sfWithin
- ogc:sfDisjoint
- ogc:sfIntersects
- ogc:sfCrosses

- Assumes Simple Features Relation Family
- Also support Egenhofer and RCC8
# RCC8, Egenhofer & Simple Features

<table>
<thead>
<tr>
<th>Simple Features</th>
<th>Egenhofer</th>
<th>RCC8</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals</td>
<td>equal</td>
<td>EQ</td>
</tr>
<tr>
<td>disjoint</td>
<td>disjoint</td>
<td>DC</td>
</tr>
<tr>
<td>intersects</td>
<td>¬ disjoint</td>
<td>¬ DC</td>
</tr>
<tr>
<td>touches</td>
<td>meet</td>
<td>EC</td>
</tr>
<tr>
<td>within</td>
<td>inside+coveredBy</td>
<td>NTPP+TPP</td>
</tr>
<tr>
<td>contains</td>
<td>contains+covers</td>
<td>NTPPi+TPPi</td>
</tr>
<tr>
<td>overlaps</td>
<td>overlap</td>
<td>PO</td>
</tr>
</tbody>
</table>
Example Data

Meta Information

:City rdfs:subClassOf ogc:Feature .
:Park rdfs:subClassOf ogc:Feature .
:exactGeometry rdfs:subPropertyOf ogc:hasGeometry .

Non-spatial Properties

:SanFrancisco rdf:type :City .
:UnionSquarePark rdf:type :Park .
:UnionSquarePark :commissioned "1847-01-01"^^xsd:date .

Spatial Properties

:geo1 ogc:asWKT "Polygon((...))"^^ogc:wktLiteral .
:geo2 ogc:asWKT "Polygon((...))"^^ogc:wktLiteral .
:UnionSquarePark ogc:sfWithin :SanFrancisco .
### Why Encode Geometry Data as a Literal?

**Advantage: single self-contained unit**

- Consistent way to select geometry information
- Consistent way to pass geometry information around

Find all water bodies that are within 1 km of Route 3

```sql
SELECT ?water ?wWKT
WHERE {
  ?water :hasExactGeometry ?wGeo .
  ?wGeo ogc:asWKT ?wWKT .
  :Route_3 :hasExactGeometry ?r3Geo .
  ?r3Geo ogc:asWKT ?r3WKT .
  FILTER(ogcf:distance(?r3WKT, ?wWKT,...) <= 1000)
}
```
Why don’t you have ogc:myFavoriteProperty?

• GeoSPARQL vocabulary is not comprehensive
  – Just enough to define a reasonable set of query patterns
  – More structural than semantic

• There are other efforts for more comprehensive vocabularies
  – ISO / TC 211
  – SOCoP
  – GeoVocamps

• GeoSPARQL vocabulary can easily be extended with other application/domain-specific vocabularies
Why don’t you support W3C Basic Geo?

- Too simple to meet our requirements
  - Can’t use different datums and coordinate systems
  - Limited number of geometry types

- W3C Basic Geo data can easily be converted to wktLiteral

PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX ogc: <http://www.opengis.net/ont/geosparql#>
SELECT (STRDT(CONCAT("POINT("?,long," ",?lat,")"),
  ogc:wktLiteral) AS ?wktLit)
WHERE { ?point geo:long ?long .
  ?point geo:lat ?lat }
GEOSPARQL QUERY FUNCTIONS
GeoSPARQL Query Functions


  ![Distance Diagram](geom1 -- geom2)


  ![Buffer Diagram](geom)


  ![Convex Hull Diagram](geom)
GeoSPARQL Query Functions


GeoSPARQL Query Functions


GeoSPARQL Query Functions


- `ogcf:getSRID(geom: ogc:wktLiteral): xsd:anyURI`
GeoSPARQL Topological Query Functions

- `ogcf:relate(geom1: ogc:wktLiteral,
  geom2: ogc:wktLiteral,
  patternMatrix: xsd:string): xsd:boolean`

<table>
<thead>
<tr>
<th>geom1</th>
<th>Interior</th>
<th>Boundary</th>
<th>Exterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interior</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Boundary</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>Exterior</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

DE-9IM Intersection Matrix

```
ogcf:relate(geom1: 'POLYGON ((0 0, 1 0, 1 1, 0 1, 0 0))',
  geom2: 'POLYGON ((0.5 0.5, 1 1, 1 0.5, 0.5 0.5))',
  patternMatrix: 'TTTFFTFFFTT')
```

ogc:contains
GeoSPARQL Topological Query Functions

Find all land parcels that are within the intersection of :City1 and :District1

```
PREFIX : <http://my.com/appSchema#>
PREFIX ogc: <http://www.opengis.net/ont/geosparql#>
PREFIX ogcf: <http://www.opengis.net/def/geosparql/functions/>
PREFIX epsg: <http://www.opengis.net/def/crs/EPSG/0/>

SELECT ?parcel
WHERE {

?parcel rdf:type :Residential .

:District1 :exactGeometry ?dGeo .
?dGeo ogc:asWKT ?dWKT .

:City1 :extent ?cGeo .
?cGeo ogc:asWKT ?cWKT .

FILTER(ogcf:sfWithin(?pWKT, ogcf:intersection(?dWKT, ?cWKT)) )
```
Basic graph pattern matching shall use the semantics defined by the **RDFS Entailment Regime** [W3C SPARQL Entailment]

Implementations shall support graph patterns involving terms from an **RDFS/OWL class hierarchy of geometry types** consistent with the one in the specified *version* of Simple Features / GML
GeoSPARQL Query Rewrite Extension

Find all water bodies within New Hampshire

```
SELECT ?water
  ?water ogc:rcc8Within :NH }
```

Same Query Specification

```
SELECT ?water
  ?water ogc:hasDefaultGeometry ?wGeo .
  ?wGeo ogc:asWKT ?wWKT .
  :NH ogc:hasDefaultGeometry ?nGeo .
  ?nGeo ogc:asWKT ?nWKT .
  FILTER(ogcf:rcc8Within(?wWKT, ?nWKT)) }
```

RCC8 Backward Chaining

Quantitative

Query Rewrite

Specified with a RIF rule
Query Rewrite Rules

• Used to compute Feature-Feature spatial relations based on default geometries
• Specified as a collection of RIF rules
• Example: ogcr:sfEquals

```
(Forall ?f1 ?f2 ?g1 ?g2 ?g1Serial ?g2Serial
  (f1[ogc:sfEquals->?f2] :-
   And
   (?f1[ogc:hasDefaultGeometry->?g1]
   ?f2[ogc:hasDefaultGeometry->?g2]
   ?g1[ogc:asWKT->?g1Serial]
   ?g2[ogc:asWKT->?g2Serial]
   External(ogcf:sfEquals(?g1Serial,?g2Serial)))
  )
  )
```
Summary of Conformance Classes

Parameters

- Serialization
  - WKT
  - GML

- Relation Family
  - Simple Features
  - RCC8
  - Egenhofer

Parameters

- Serialization
  - WKT
  - GML

- Relation Family
  - Simple Features
  - RCC8
  - Egenhofer

Determines geometry classes and geometry literal datatype

Determines topology properties and topology functions
Implementing Spatial Operations

- These are standard OGC operators that have been around for some time
- Lots of infrastructure available
  - Open Source
    - GEOS
    - pysal
    - PostGIS
    - JTS Topology Suite
  - Commercial
    - esri
    - ORACLE DATABASE
Other Considerations

• Have to handle geometries from multiple Spatial Reference Systems simultaneously
  – Normalize to common SRS on-the-fly during computation
  – Pre-normalize ahead of time

• Spatial Indexing very important for performance
  – Normalize to common SRS during indexing
Summary

• GeoSPARQL Defines:
  – Basic vocabulary, Query functions, Entailment component

• Based on existing OGC/ISO standards
  – WKT, GML, Simple Features, ISO 19107

• Uses SPARQL’s built-in extensibility framework

• Modular specification
  – Allows flexibility in implementations
  – Easy to extend

• Accommodates qualitative and quantitative systems
  – Same query specification for qualitative (core + topology vocabulary) and quantitative (all components, incl. query rewrite)
Future Work

• Define new conformance classes
  – KML, GeoJSON

• Define OWL axioms for qualitative spatial reasoning
  – `ogc:sfWithin rdf:type owl:TransitiveProperty`

• Hybrid qualitative / quantitative spatial reasoning

• Define standard methodology for (virtually) converting legacy feature data represented using the general feature model to RDF (RDB2RDF for spatial)
Thanks to all members of the GeoSPARQL SWG!

QUESTIONS?