GeoSPARQL: A Geographic Query Language for RDF

SemTech 2011
09 June 2011

Todd Pehle
tpehle@orbistechnologies.com

Matthew Perry, Ph.D.
matthew.perry@oracle.com

Agenda

- Semantic Web and Geo Web Convergence
- The Need for a Geographic Query Language for RDF
- Design Aspects of GeoSPARQL
  - GeoSPARQL Vocabulary
  - Query Functions
  - Query Rewrite Rules
- GeoSPARQL Examples
  - Sample GeoSPARQL Queries
  - GeoSPARQL in the Web of Linked Locations
- Future Work
- Summary

Convergence of the Semantic Web & Geo Web

- The Semantic Web and Geo Web traveled similar yet parallel paths over last decade
- With the rise of Linked Data and Neo Geography the lines between these two worlds are getting very blurry
- As the Mobile Web and Sensor Web continues to grow the distinctions will become even fuzzier
- The Web of Linked Locations represents a spatiotemporal projection of the Web of Linked Data
- The projection represents a global spatial knowledgebase that begins to merge the Semantic Web & Geo Web
- A new kind of “map” is evolving: one which links mapping data with non-mapping data
- A key missing ingredient:
  - ***Geographic Query for RDF***

Geo Web Representations & Query

- Spatial Representation in the Geo Web:
  - Geographic features modeled in GML/KML/WKT, etc.
  - Use of multiple CRS + multi-scale + multi-context = heterogeneity
  - Separation of geographic feature from geometric representations
  - Spatial relations such as topology networks sometimes pre-computed
  - Thematic attribution describes non-geometric aspects of feature
- Geo Web Query:
  - Index GML/KML files; non-standard hybrid IR/structured geo query
  - Spatial databases offer object-relational storage of geometry in addition to thematic attribution allowing SQL query extensions
  - Spatial relations computed “on-the-fly” since pre-computing explodes number of comparisons and size of storage for a single spatial relation
  - Due to query complexity approximations such as MBR, convex hulls and centroids are sometimes pre-computed
  - OGC Specifications such as WFS + Filter Encoding allow rich set of spatial, temporal and thematic filtering via web service

Semantic Web Representations & Query

- Semantic Web Geospatial Representations:
  - W3C Basic Geo Vocabulary + some GeoRSS
  - Typically uses WGS84 as default CRS
  - Very few linear and area features
  - Mainly “near” relationships and some topological “containedIn” relations
  - No common vocabularies yet for feature/geometry/spatial relations
- Semantic Web Geospatial Query:
  - Simplest, most popular types of SPARQL involve W3C Basic Geo lat/lon
  - Often doesn’t use a spatial index but instead is pure graph match
  - Goal is to get richer geometries and richer semantics
  - However, geometric computations not well suited for logic reasoning
  - Geospatial computations in triple stores pushed outside graph
  - Little combination of quantitative and qualitative spatial reasoning
- Moral of the Story:
  - Queries in Geo Web offer rich GIS functionality, but weaker data integration
  - Queries in Semantic Web offer rich query across data, but weaker geo query

Why the Need for a "*Standards-Based*" RDF Query Language?

- Lots of interest in querying geo data in RDF!
- Many triple stores now support a form of geospatial query
- Examples include: Oracle, BBN Parliament, Virtuoso, Franz, Big OWLIM, Big Data, Open Sahara, etc.
- Due to lack of standard query syntax and semantics clients lose uniformity of geo query across SPARQL endpoints
- Current de facto standard to query geometric data is via W3C Basic Geo Vocabulary (minimal expressivity)
- OGC successfully instituted GML/WFS/Filter Encoding in the GIS domain
- The same uniformity (common syntax & semantics) of geo querying across SPARQL endpoints in LOD is needed

orbis
TECHNOLOGIES, INC.

GeoSPARQL

OGC & ISO Geo Semantic Web Activities

- OGC
  - GeoSPARQL Standards Working Group
  - Geosemantics Domain Working Group
- ISO
  - ISO/RS 19150
    - Geographic information – Ontology
  - ISO 19101-1rev
    - Geographic information – Reference model - Part 1: Fundamentals
  - ISO/PT 19150-1
    - Geographic information – Ontology - Part 1: Framework
  - ISO/PT 19150-2
    - Geographic information – Ontology - Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)

About the GeoSPARQL SWG

- Open Geospatial Consortium standards working group
  - 13 voting members, 36 observers
  - First meeting June 17, 2010
  - Planned completion in 2011
- Submitting Organizations
  - Australian Bureau of Meteorology
  - Bentley Systems, Inc.
  - CSIRO
  - Defence Geospatial Information Working Group (DGIWG)
  - GeoConnections - Natural Resources Canada
  - Interactive Instruments GmbH
  - Oracle America
  - Ordnance Survey
  - Raytheon Company
  - Traverse Technologies, Inc.
  - US Geological Survey (USGS)

Goals for GeoSPARQL

- Work within SPARQL's extensibility framework
- Simple enough for general users but capable enough for GIS professionals
- Accommodate systems based on qualitative spatial reasoning and systems based on quantitative geometries
- Don't re-invent the wheel!

ISO 19107 – Spatial Schema
ISO 13249 – SQL/MM

Simple Features
Well Known Text (WKT)
GML
KML
GeoJSON

What Does GeoSPARQL Give Us?

- Standard Vocabulary for Spatial Information
  - 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, …
- Query Rewrite Rules
  - Expand feature-feature query into geometry query
  - Gives a common interface for qualitative and quantitative systems

A Simple Example

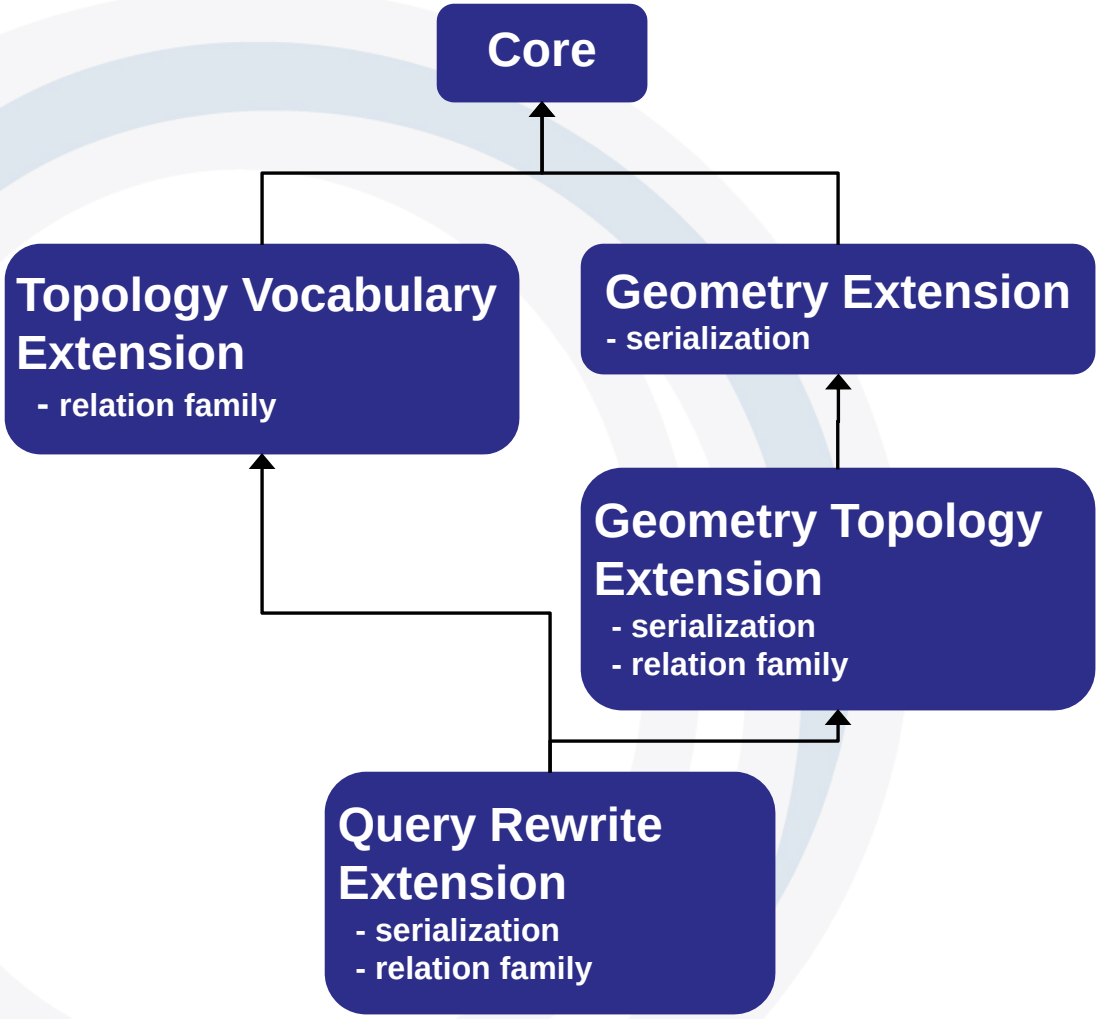
Example Data: SemTech 2011 is located in San Francisco

```
:SemTech2011   :locatedIn     :SanFrancisco .
:SanFrancisco  :hasPointGeometry [
        ogc:asWKT “POINT(-122.4192 37.7793)”^^ogc:WKTLiteral ] .
```

Example Query: Find airports within 100 KM of San Francisco

```
SELECT ?airport
WHERE { ?airport rdf:type :Airport .
        ?airport :hasPointGeometry [
                 ogc:asWKT ?aPointGeom ]
        FILTER(ogcf:distance(?aPointGeom,
                 “POINT(-122.4192 37.7793)”^^ogc:WKTLiteral,
                  ogc:km) <= 100) }
```

Components of the Spec



Parameters

- Serialization
  - *WKT*
  - *GML*

Determines geometry classes and geometry literal datatype

- Relation Family
  - *Simple Features*
  - *RCC8*
  - *Egenhofer*

Determines topology properties and topology functions

Why no Universal Geometry Ontology?

- Geometry types in existing standards are different abstractions
  - e.g. Polygon edges can be arcs in ISO 19107 / GML but must be straight lines in Simple Features
- Any hierarchy we get may be too restrictive
  - You should be able to conceptualize a geometry any way you want as long as you can support the required operations.
- All we're left with is a single root Geometry class

Assumptions and Design Aspects

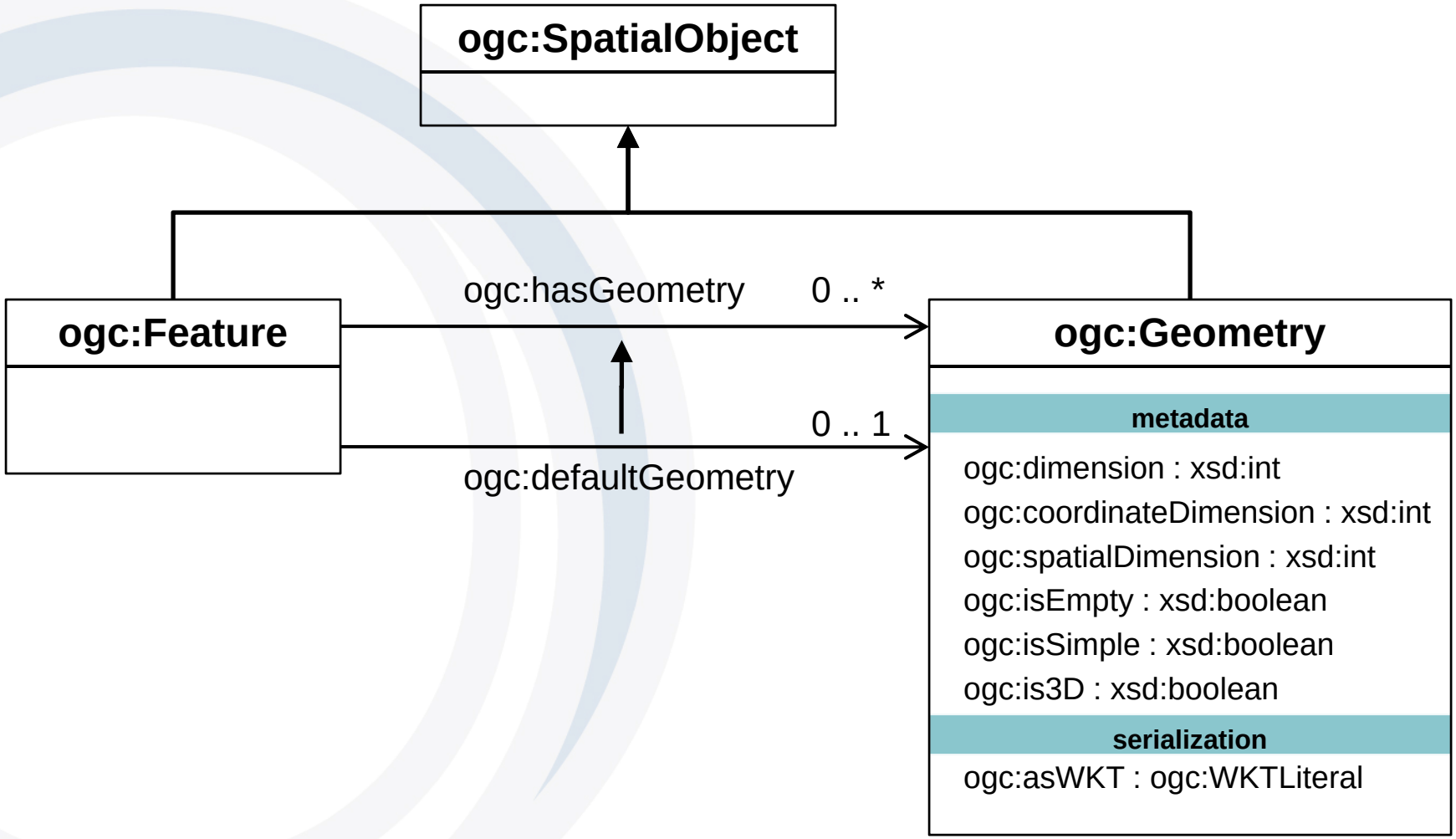
- Assumption:
  - Serialization = *WKT*
  - Relation family = *Simple Features*

- Major Design Aspects:
  - GeoSPARQL Vocabulary
  - Query Functions
  - Query Rewrite Rules

orbis
TECHNOLOGIES, INC.

GeoSPARQL Vocabulary

Basic Classes and Relations



Why Encode Geometry Data as a Literal?

Advantage: single self-contained unit

Consistent way to select geometry information

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

```
SELECT  ?water ?wWKT
WHERE { ?water      rdf:type             :WaterBody .
        ?water      :hasExactGeometry    ?wGeo .
        ?wGeo       ogc:asWKT            ?wWKT .
        :Route_3    :hasExactGeometry    ?r3Geo .
        :r3Geo      ogc:asWKT            ?r3WKT .
        FILTER(ogcf:distance(?r3WKT, ?wWKT,…) <= 1000)
}
```

Consistent way to pass geometry information around

Details of ogc:WKTLiteral

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.

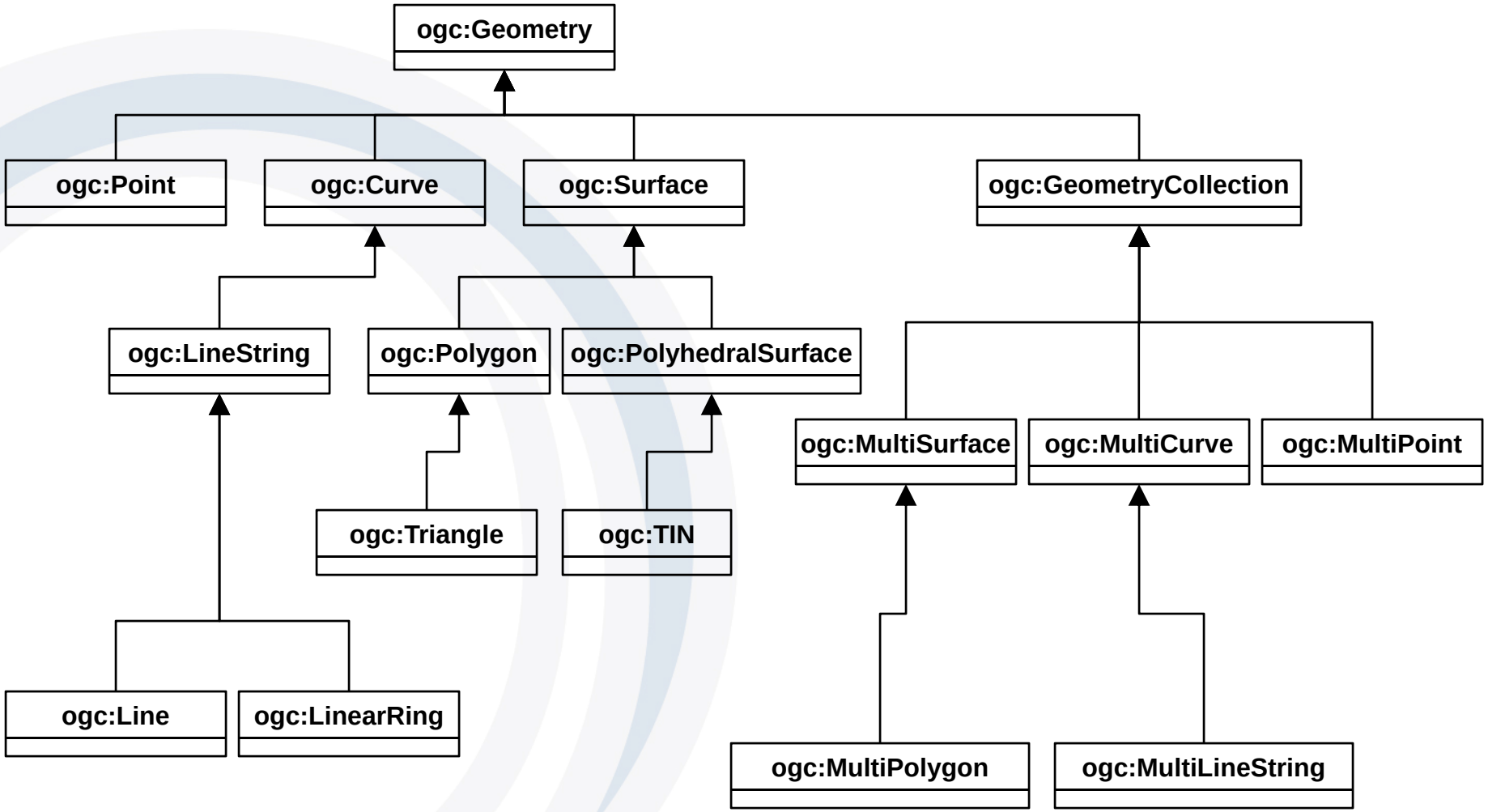
```
"<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
POINT(-122.4192 37.7793)"^^ogc:WKTLiteral
```

WGS84 longitude – latitude is the default CRS

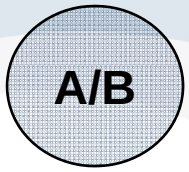
```
"POINT(-122.4192 37.7793)"^^ogc:WKTLiteral
```

European Petroleum Survey Group (EPSG) maintains a set of CRS identifiers.

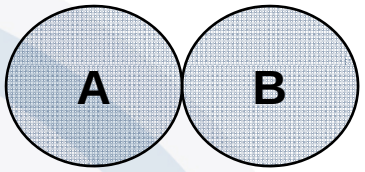
Simple Features Geometry Types



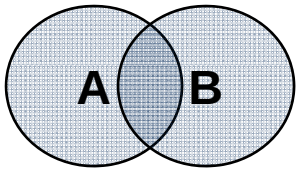
Topological Relations between ogc:SpatialObject



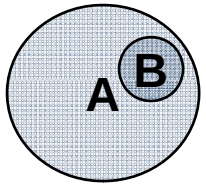
ogc:equals



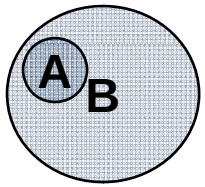
ogc:touches



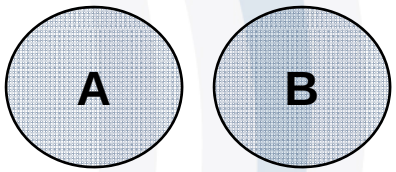
ogc:overlaps



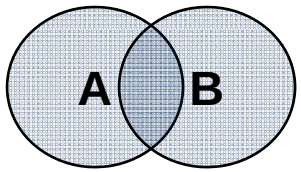
ogc:contains



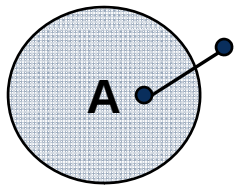
ogc:within



ogc:disjoint



ogc:intersects



ogc:crosses

RCC8, Egenhofer & Simple Features

Simple Features	Egenhofer	RCC8
equals	equal	EQ
disjoint	disjoint	DC
intersects	¬ disjoint	¬ DC
touches	meet	EC
within	inside+coveredBy	NTPP+TPP
contains	contains+covers	NTPPi+TPPi
overlaps	overlap	PO

Example Data

Meta Information

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

Non-spatial Properties

```
:Nashua                rdf:type    :City .
:MinesFallsPark        rdf:type    :Park .
:MinesFallsPark        :opened     “1950-03-01”^^xsd:date .
```

Spatial Properties

```
:MinesFallsPark :exactGeometry  :geo1 .
:geo1           rdf:type        ogc:Polygon  .
:geo1           ogc:asWKT       “Polygon((…))”^^ogc:WKTLiteral .

:Nashua         :exactGeometry  :geo2 .
:geo2           rdf:type        ogc:Polygon  .
:geo2           ogc:asWKT       “Polygon((…))”^^ogc:WKTLiteral .

:MinesFallsPark ogc:within      :Nashua .
```

orbis
TECHNOLOGIES, INC.

GeoSPARQL Query Functions

GeoSPARQL Query Functions

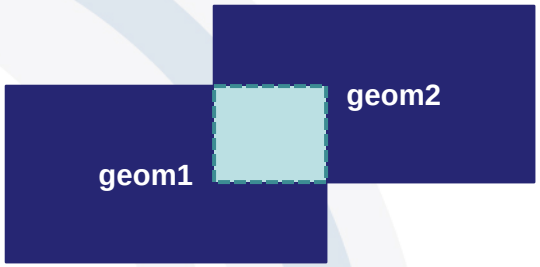
- **ogcf:distance(geom1: ogc:WKTLiteral, geom2: ogc:WKTLiteral, units: xsd:anyURI): xsd:double**

- **ogcf:buffer(geom: ogc:WKTLiteral, radius: xsd:double, units: xsd:anyURI): ogc:WKTLiteral**

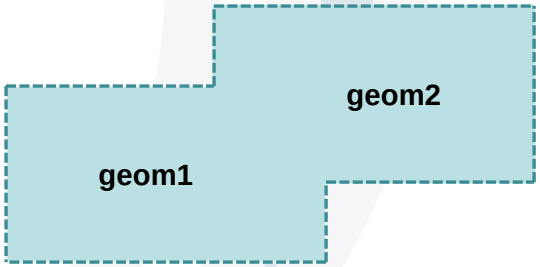
- **ogcf:convexHull(geom: ogc:WKTLiteral): ogc:WKTLiteral**

GeoSPARQL Query Functions

- `ogcf:intersection(geom1: ogc:WKTLiteral, geom2: ogc:WKTLiteral): ogcf:WKTLiteral`



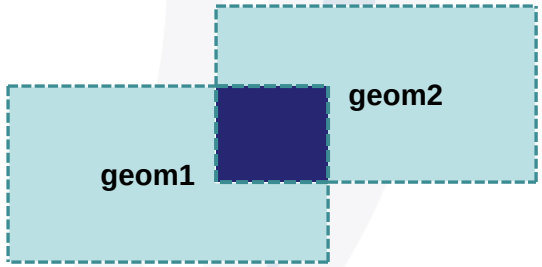
- `ogcf:union(geom1: ogc:WKTLiteral, geom2: ogc:WKTLiteral): ogc:WKTLiteral`



GeoSPARQL Query Functions

- `ogcf:difference(geom1: ogc:WKTLiteral, geom2: ogc:WKTLiteral): ogcf:WKTLiteral`

- `ogcf:symDifference(geom1: ogc:WKTLiteral, geom2: ogc:WKTLiteral): ogc:WKTLiteral`



GeoSPARQL Query Functions

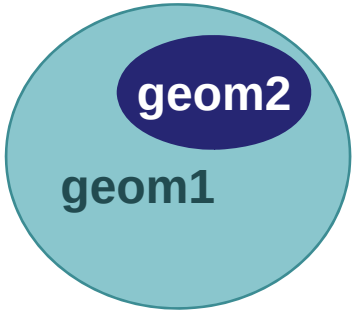
- **ogcf:envelope(geom: ogc:WKTLiteral): ogcf:WKTLiteral**

- **ogcf:boundary(geom1: ogc:WKTLiteral): ogc:WKTLiteral**

GeoSPARQL Query Functions

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

DE-9IM Intersection Matrix				
		geom2		
		Interior	Boundary	Exterior
geom1	Interior	T	T	T
	Boundary	F	F	T
	Exterior	F	F	T



ogc:contains

patternMatrix: TTTFFTFFT

GeoSPARQL Query Functions

- **ogcf:equals(geom1: ogc:WKTLiteral,**
  geom2: ogcf:WKTLiteral): xsd:boolean
- **ogcf:disjoint(geom1: ogc:WKTLiteral,**
  geom2: ogcf:WKTLiteral): xsd:boolean
- **ogcf:intersects(geom1: ogc:WKTLiteral,**
  geom2: ogcf:WKTLiteral): xsd:boolean
- **ogcf:touches(geom1: ogc:WKTLiteral,**
  geom2: ogcf:WKTLiteral): xsd:boolean
- **ogcf:crosses(geom1: ogc:WKTLiteral,**
  geom2: ogcf:WKTLiteral): xsd:boolean
- **ogcf:within(geom1: ogc:WKTLiteral,**
  geom2: ogcf:WKTLiteral): xsd:boolean
- **ogcf:contains(geom1: ogc:WKTLiteral,**
  geom2: ogcf:WKTLiteral): xsd:boolean
- **ogcf:overlaps(geom1: ogc:WKTLiteral,**
  geom2: ogcf:WKTLiteral): xsd:boolean

Example Queries

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

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

SELECT  ?water ?wWKT
WHERE { ?water      rdf:type             :WaterBody .
        ?water      :exactGeometry       ?wGeo .
        ?wGeo       ogc:asWKT            ?wWKT .
        :Route_3    :exactGeometry       ?r3Geo .
        :r3Geo      ogc:asWKT            ?r3WKT .
        FILTER(ogcf:distance(?r3WKT, ?wWKT,
                             ogc:km) <= 1)
}
```

Example Queries

Find all land parcels for sale within a constant search polygon

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

SELECT  ?parcel
WHERE { ?parcel  rdf:type          :Residential .
        ?parcel  :for_sale         "true"^^xsd:boolean .
        ?parcel  :exactGeometry    ?pGeo .
        ?pGeo    :asWKT            ?pWKT
        FILTER(ogcf:within(?pWKT,
                "Polygon((…))"^^ogc:WKTLiteral)) }
```

Example Queries

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

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

SELECT  ?parcel
WHERE { ?parcel      rdf:type         :Residential .
        ?parcel      :exactGeometry   ?pGeo .
        ?pGeo        ogc:asWKT        ?pWKT .
        :District1   :exactGeometry   ?dGeo .
        ?dGeo        ogc:asWKT        ?dWKT .
        :City1       :extent          ?cGeo .
        ?cGeo        ogc:asWKT        ?cWKT .
        FILTER(ogcf:within(?pWKT,
                ogcf:intersection(?dWKT,?cWKT)))}
```

Example Queries

Find the three closest Mexican restaurants

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

SELECT  ?restaurant
WHERE { ?restaurant  rdf:type         :Restaurant .
        ?restaurant  :cuisine         :Mexican .
        ?restaurant  :pointGeometry   ?rGeo .
        ?rGeo        ogc:asWKT        ?rWKT }
ORDER BY ASC(ogcf:distance("POINT(…)"^^ogc:WKTLiteral,
                           ?rWKT, ogc:KM))
LIMIT 3
```

orbis
TECHNOLOGIES, INC.

GeoSPARQL Query Rewrite Rules

Motivation for Query Rewrite Rules

Find all water bodies within New Hampshire

```
SELECT ?water
WHERE { ?water  rdf:type      :WaterBody .
        ?water  ogc:within    :NH }
```

Same Query Specification

Qualitative → RCC8 Backward Chaining

Quantitative

```
SELECT ?water
WHERE { ?water  rdf:type      :WaterBody .
        ?water  ogc:defaultGeometry      ?wGeo .
        ?wGeo   ogc:asWKT                ?wWKT .
        :NH     ogc:defaultGeometry      ?nGeo .
        ?nGeo   ogc:asWKT                ?nWKT .
        FILTER(ogcf:within(?wWKT, ?nWKT)) }
```

Query Rewrite

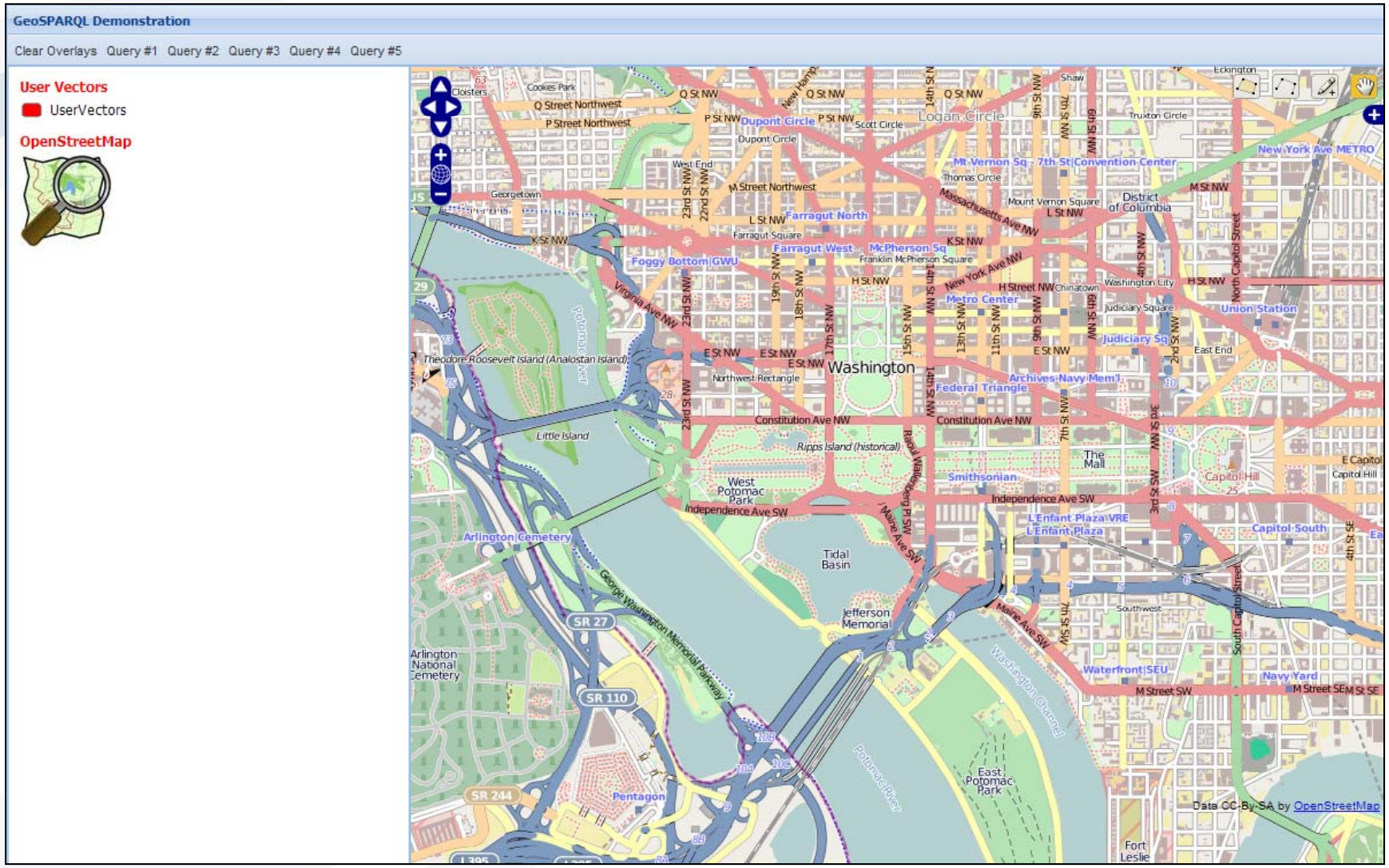
Specified with a RIF rule

Query Rewrite Rules

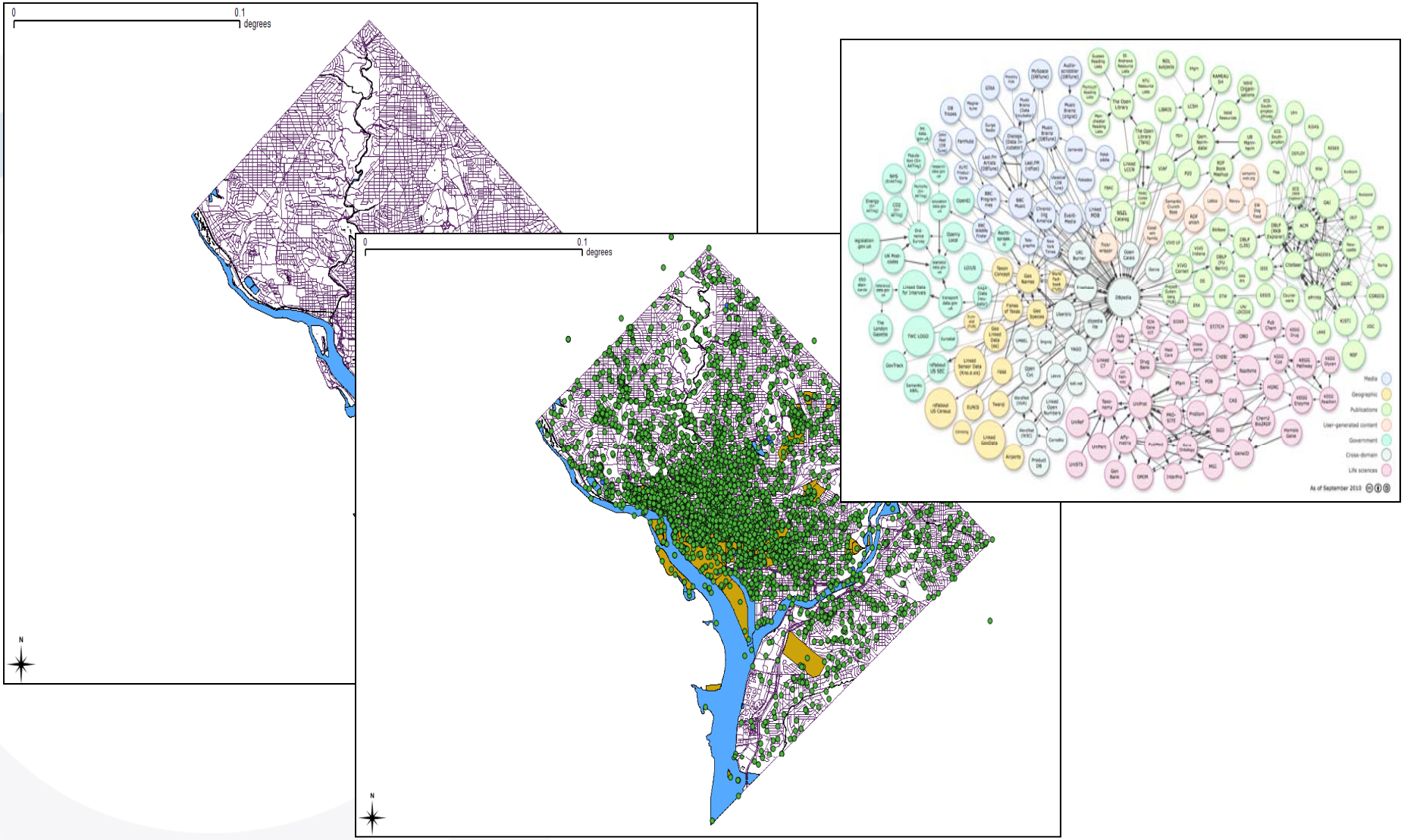
- Used to compute Feature-Feature spatial relations based on default geometries
- Specified as a collection of RIF rules
- Example: ogc:equals

```
(Forall ?f1 ?f2 ?g1 ?g2 ?g1Serial ?g2Serial
  (f1[ogc:equals->?f2] :-
    And
      (?f1[ogc:defaultGeometry->?g1]
       ?f2[ogc:defaultGeometry->?g2]
       ?g1[ogc:asWKT->?g1Serial]
       ?g2[ogc:asWKT->?g2Serial]
       External(ogcf:equals(?g1Serial,?g2Serial)))
  )
)
```

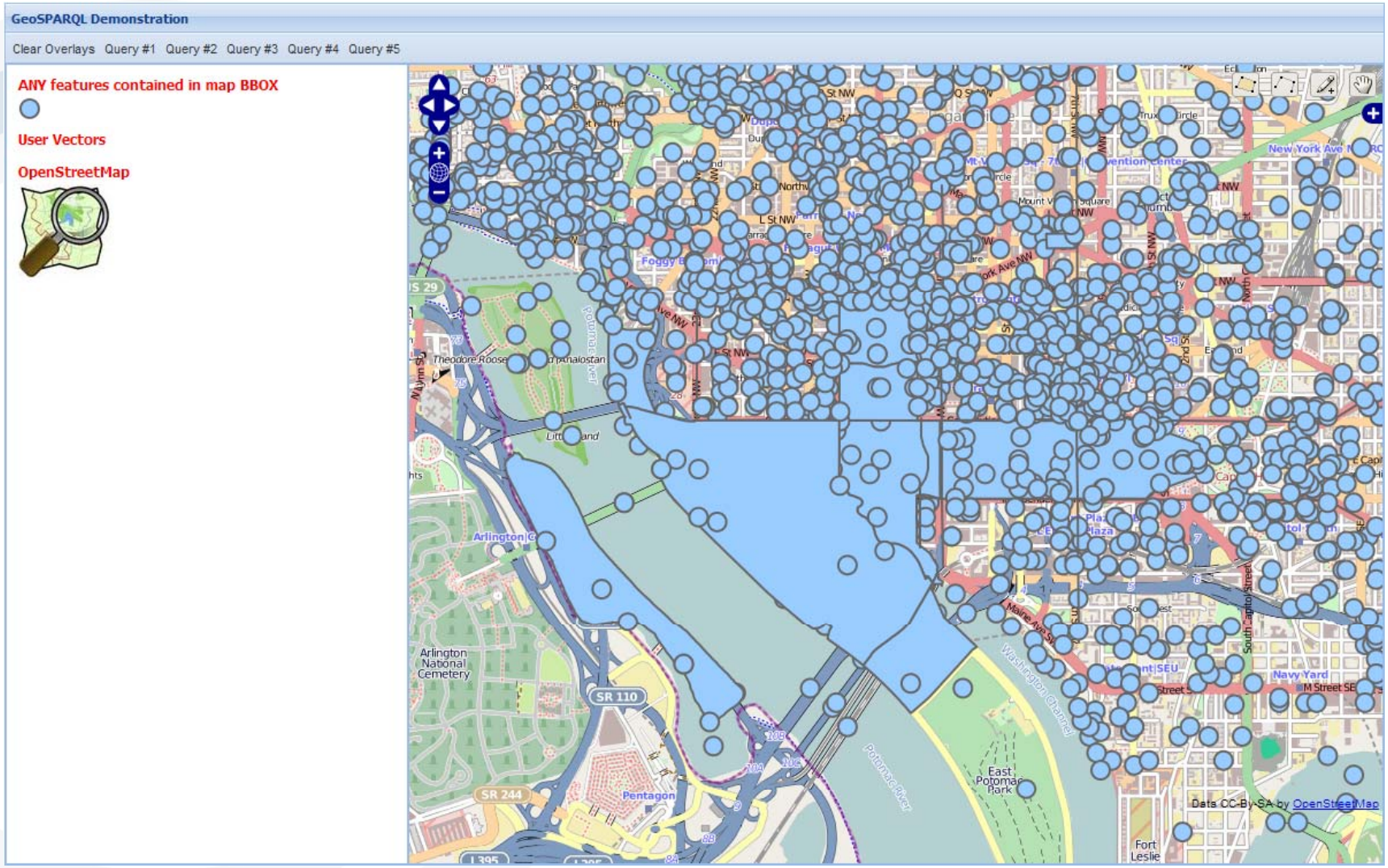
GeoSPARQL Demonstration



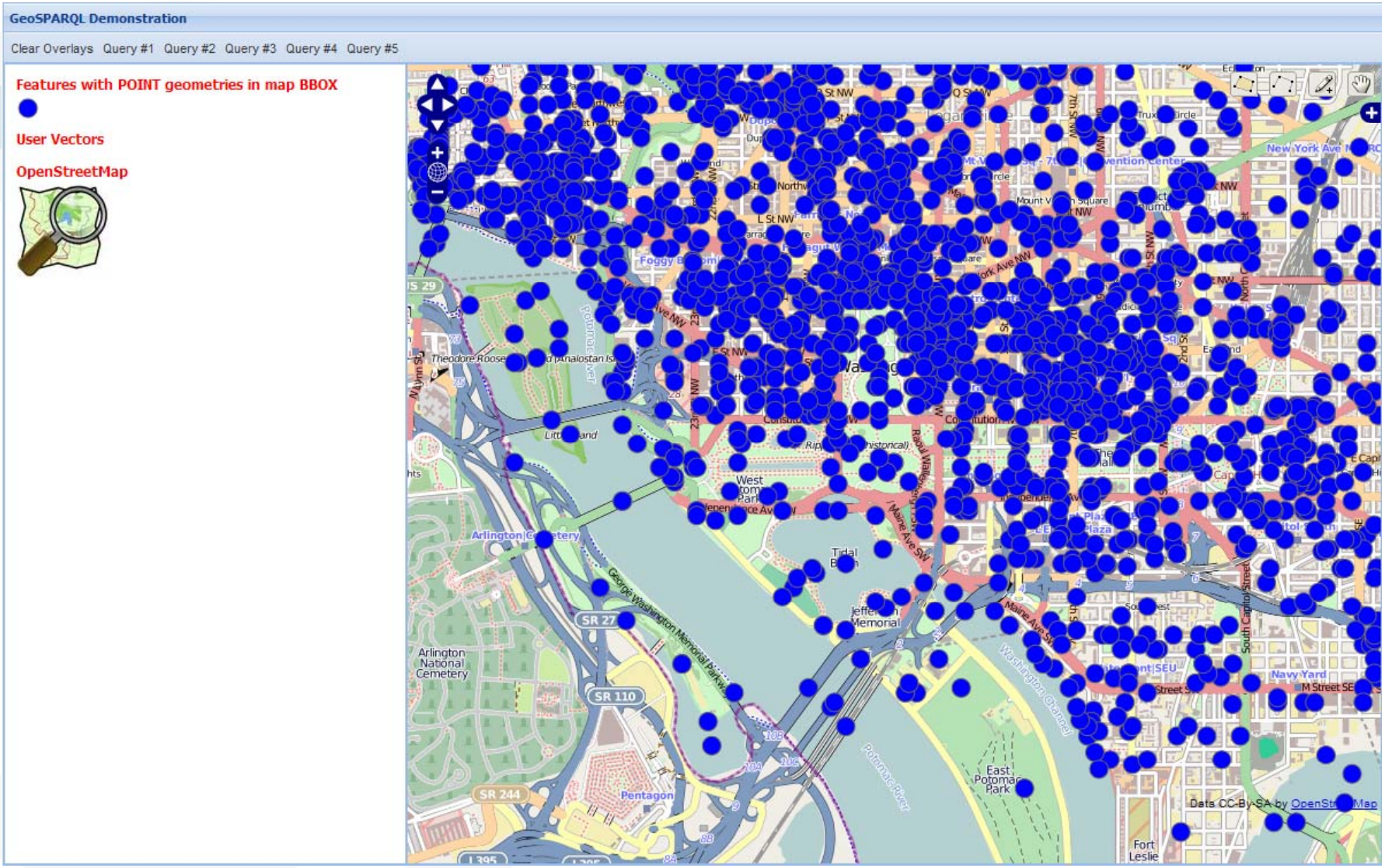
The (Good) Mess We're In



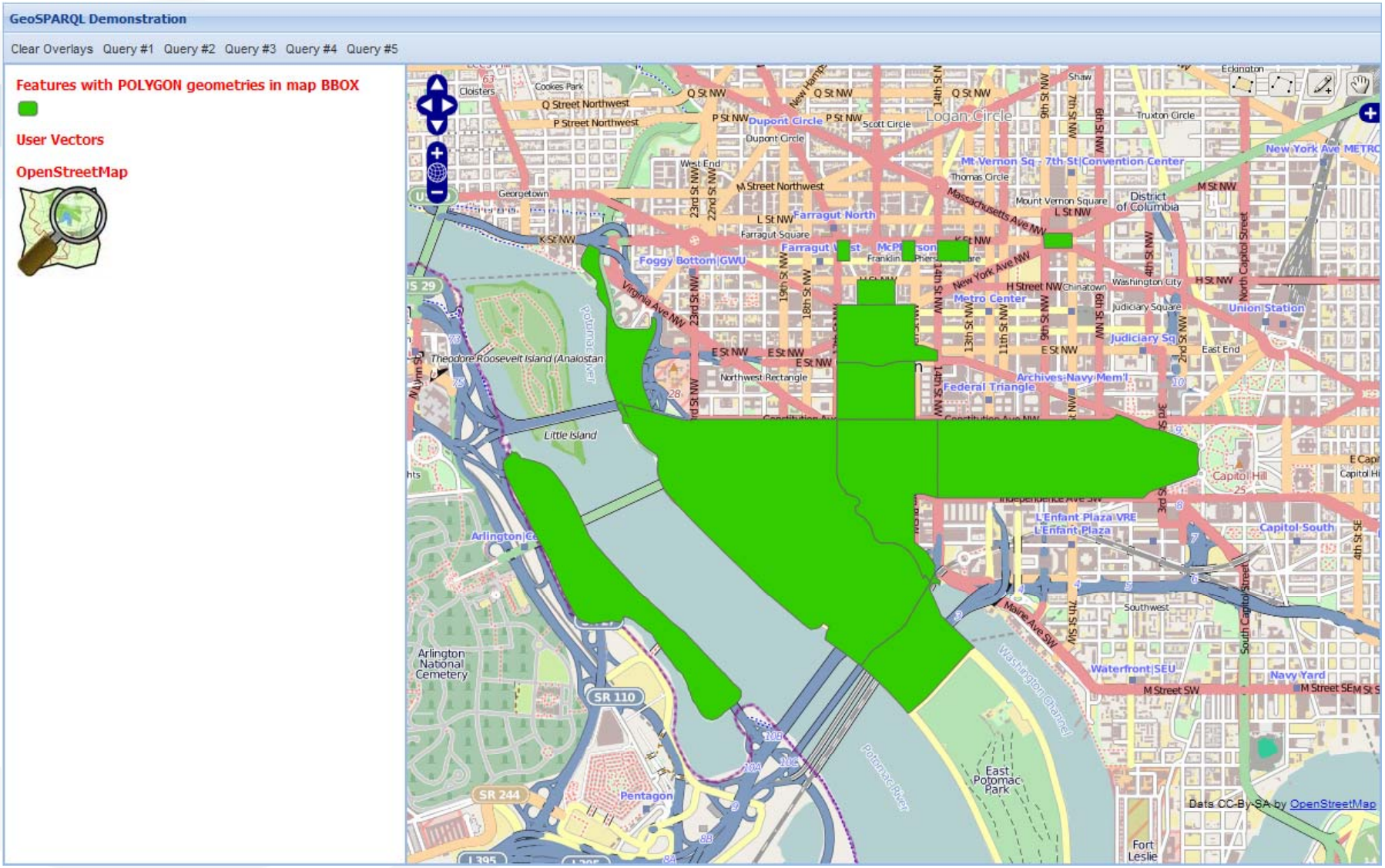
ANY Features “contained in” Map Bounding Box



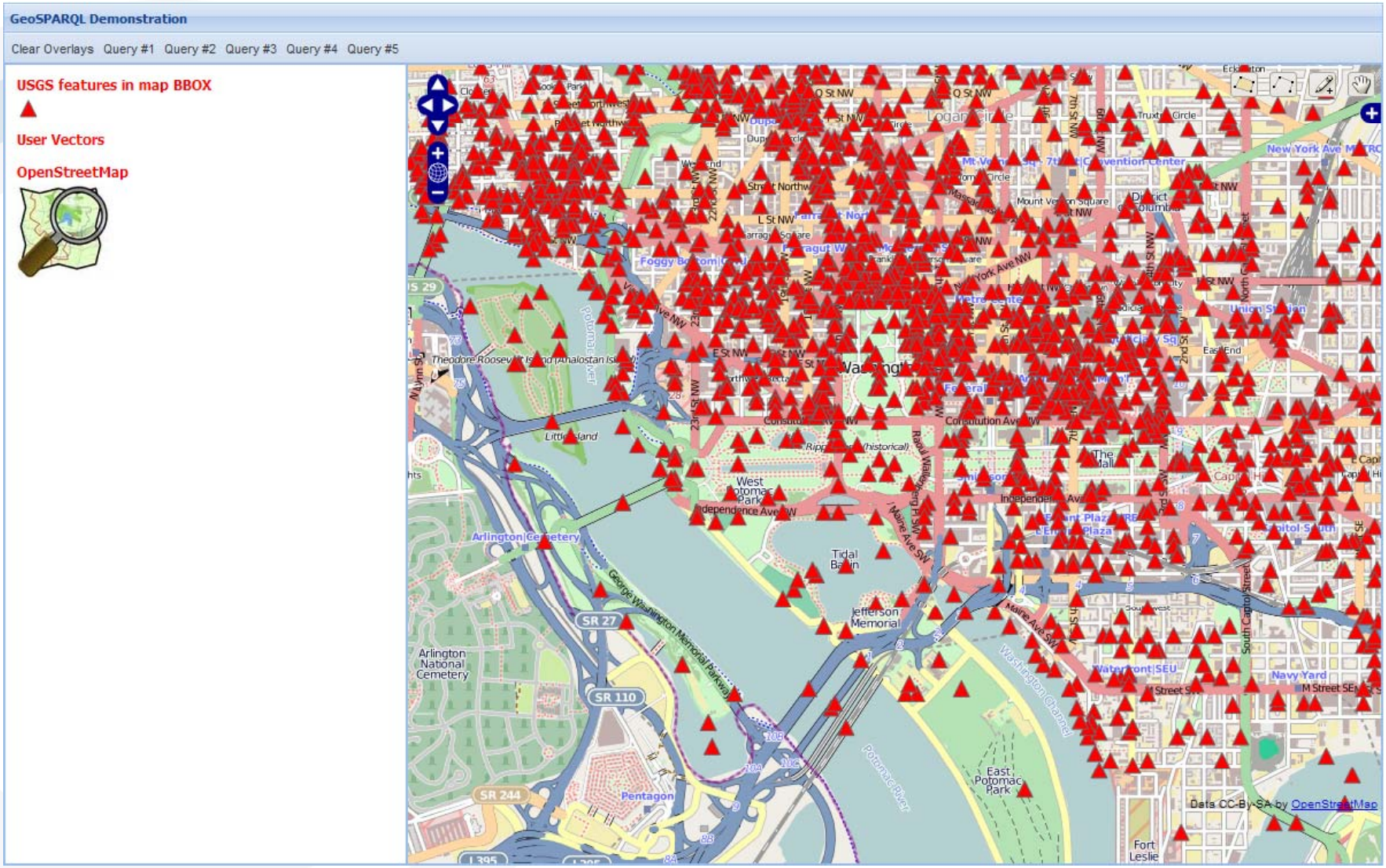
Point Features “contained in” Map Bounding Box



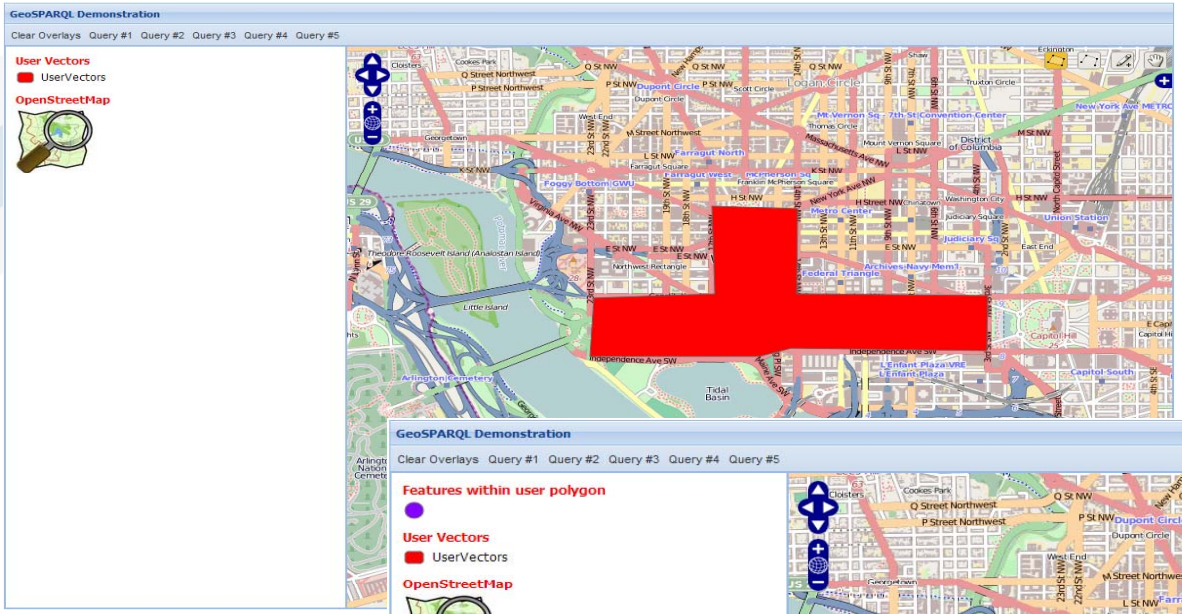
Polygon Features “contained in” Map Bounding Box

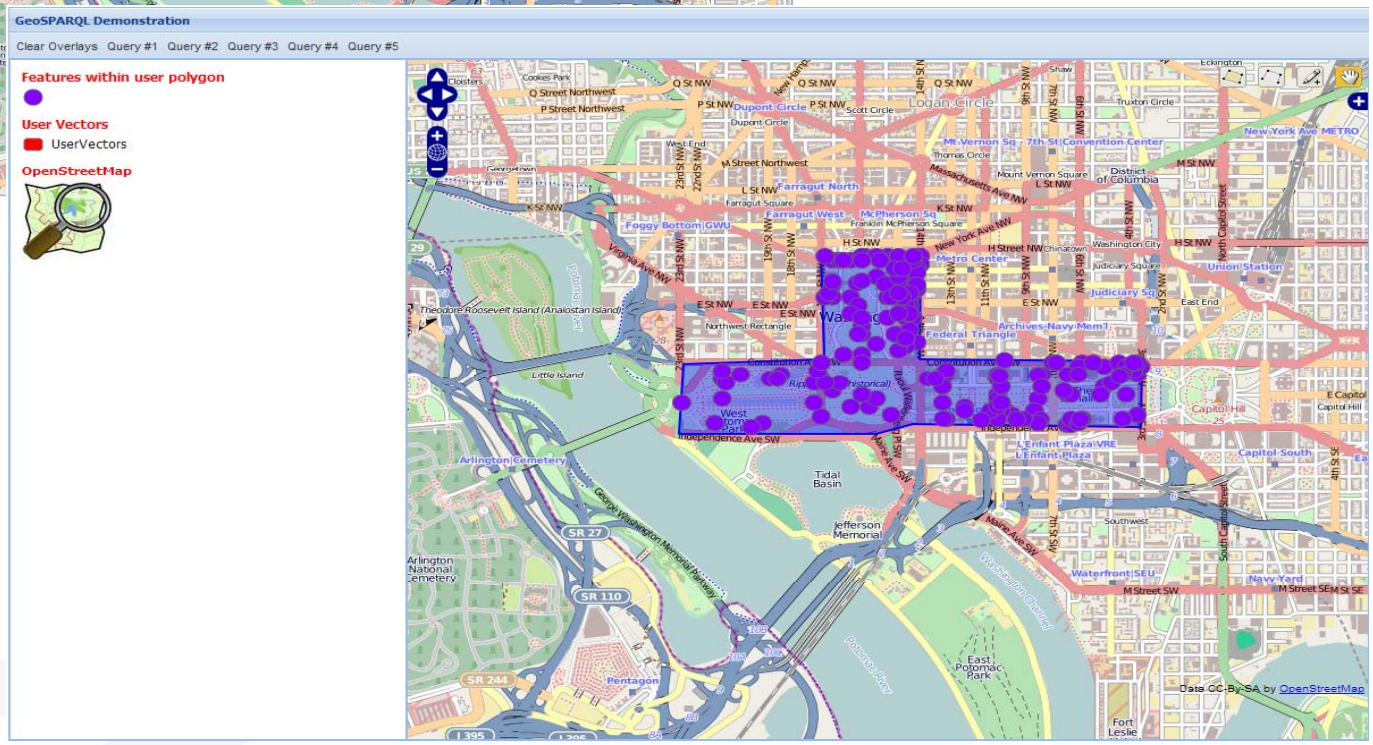


USGS GNIS "contained in" Map Bounding Box

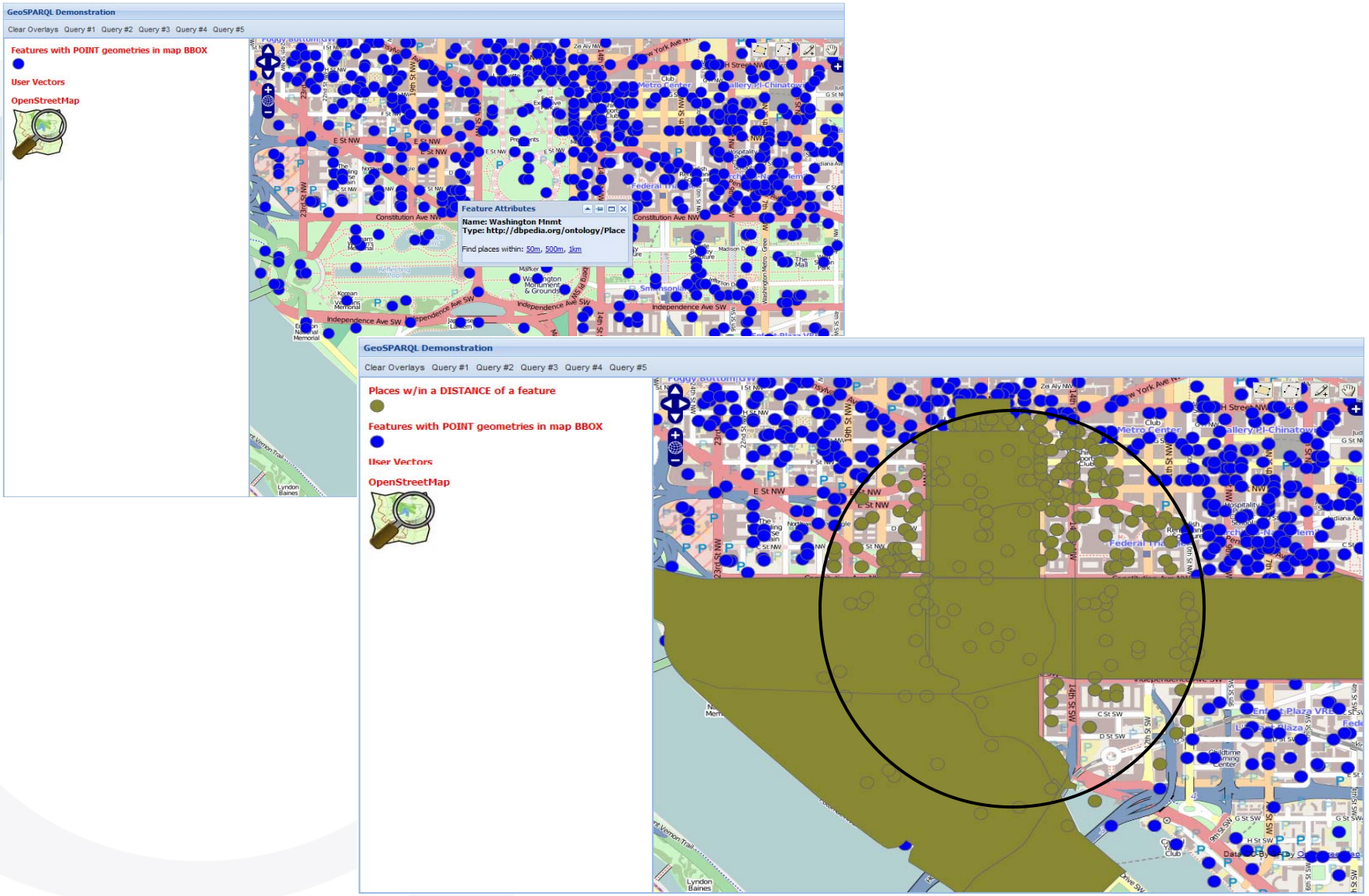


Point features “contained in” User-Defined Polygon

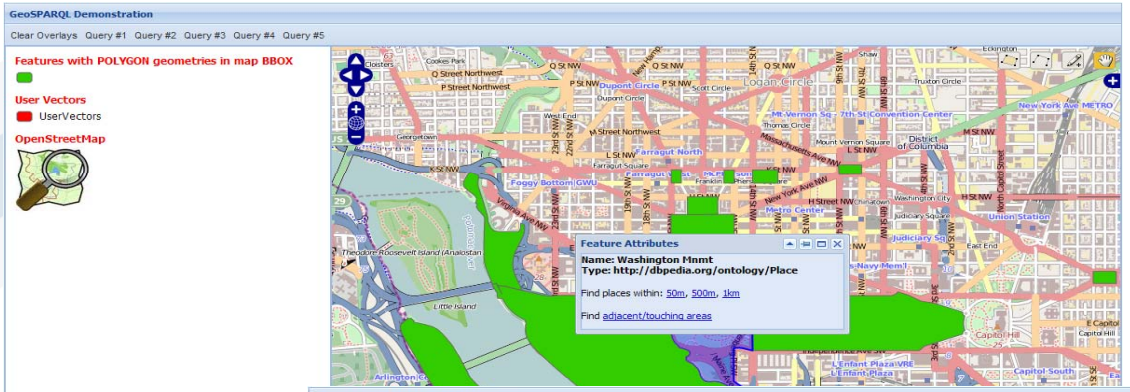


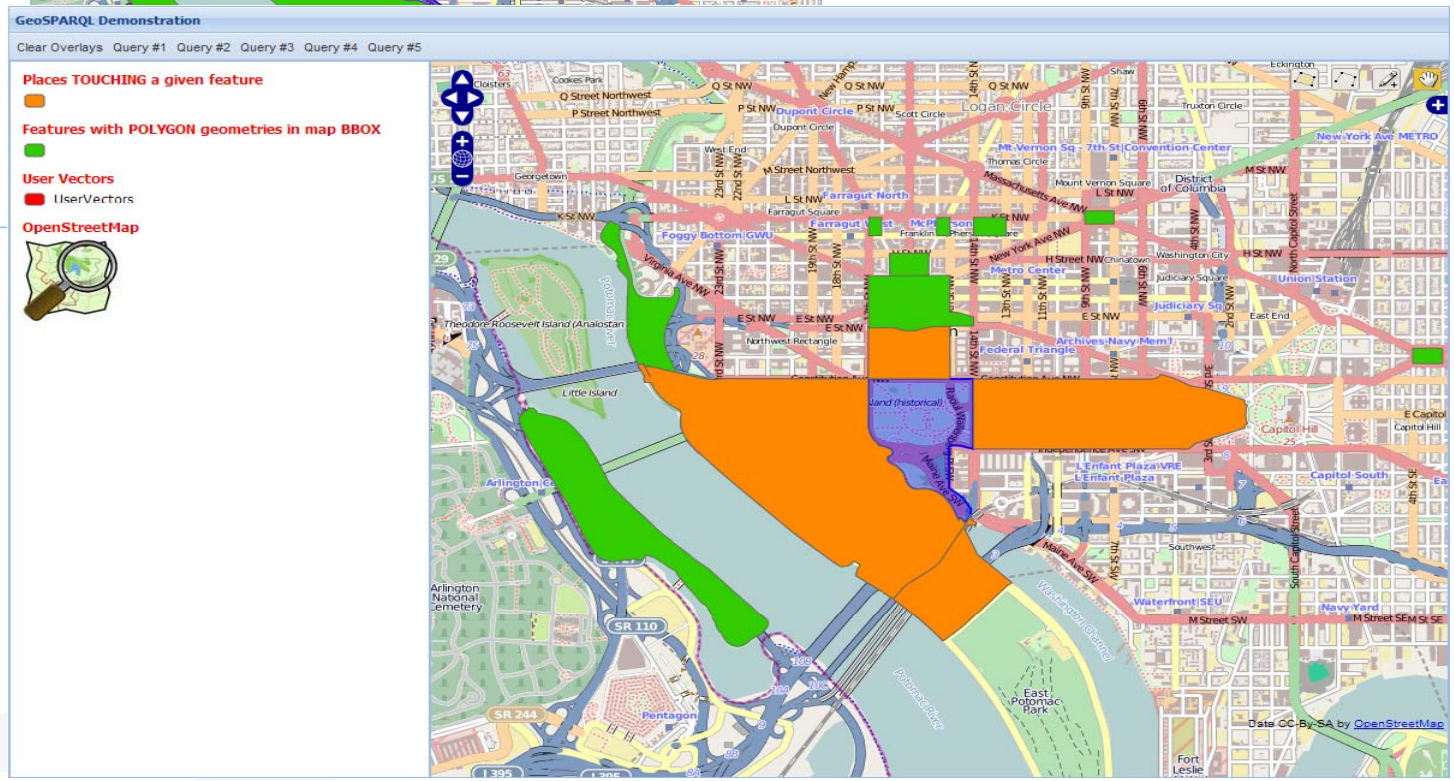


Point features “within 1km of” Washington Monument

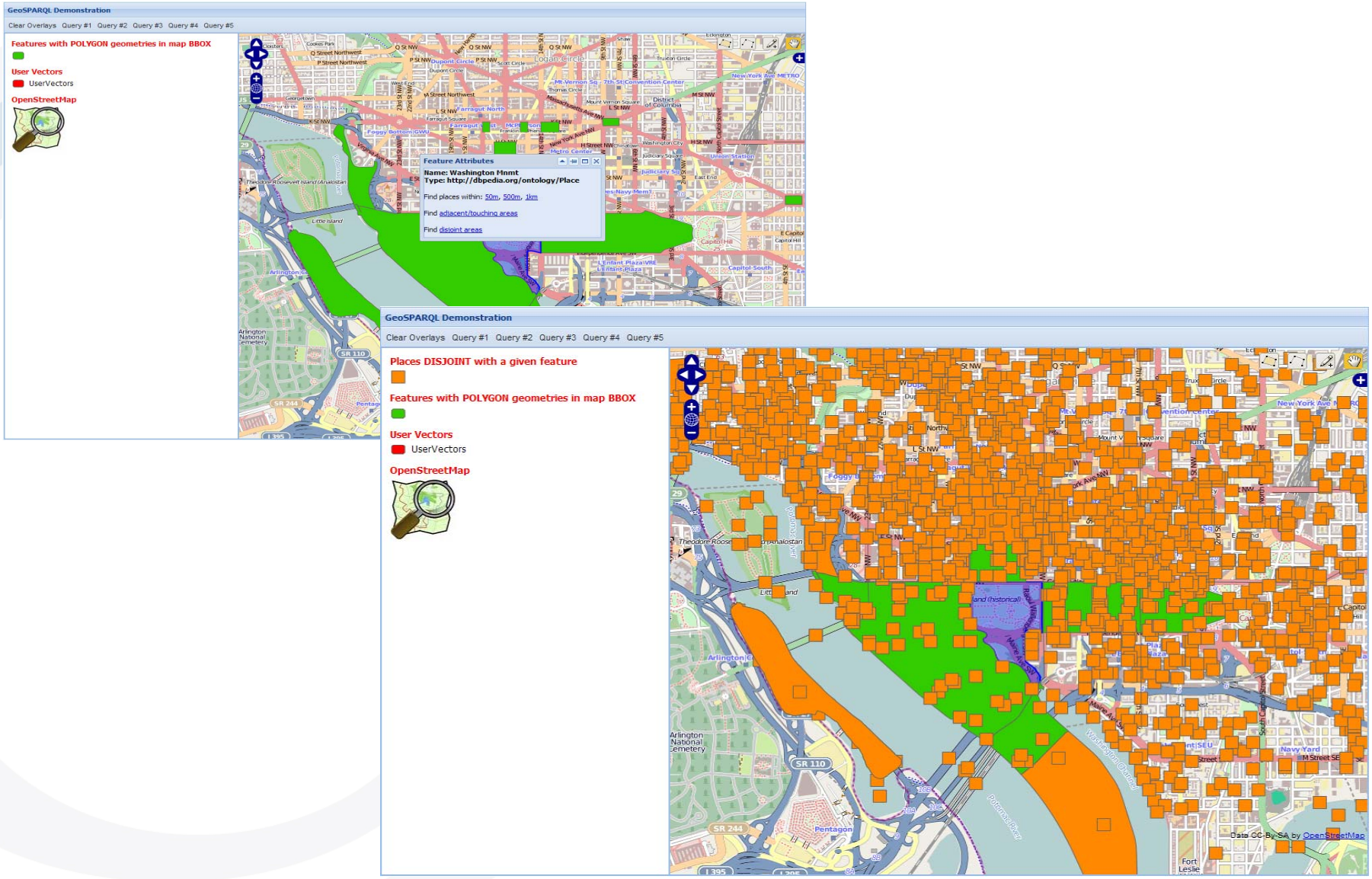


Features “touching” Washington Monument Park Area





Features “disjoint with” Washington Monument Park Area



orbis
TECHNOLOGIES, INC.

GeoSPARQL in the Web of Linked Locations

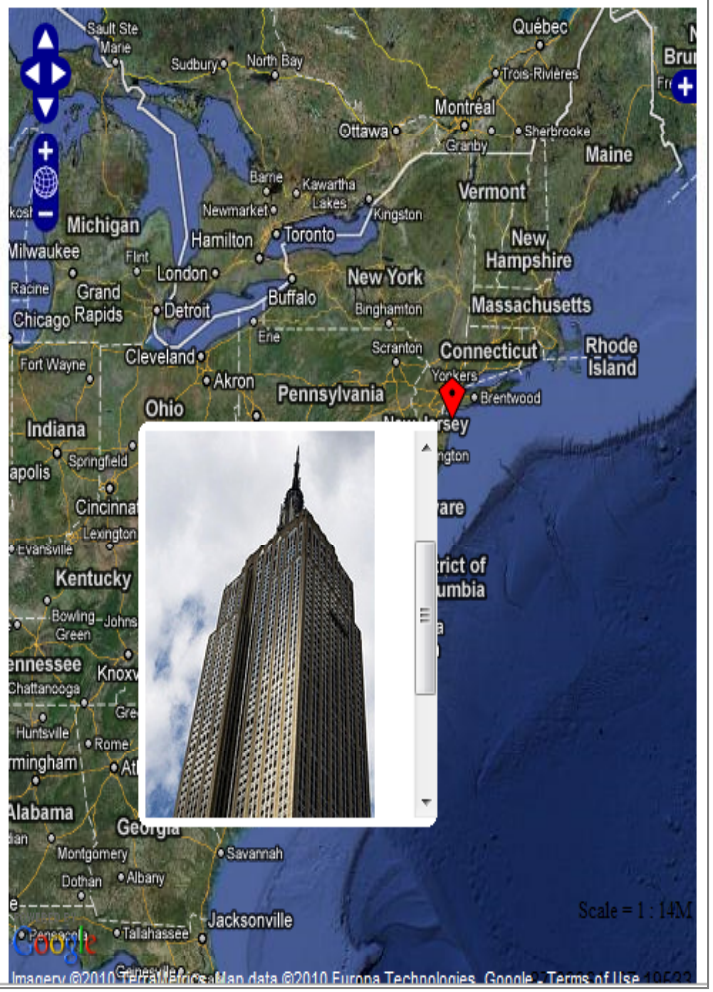
Browsing the Web of Linked Locations

http://dbpedia.org/resource/Empire_State_Building Open

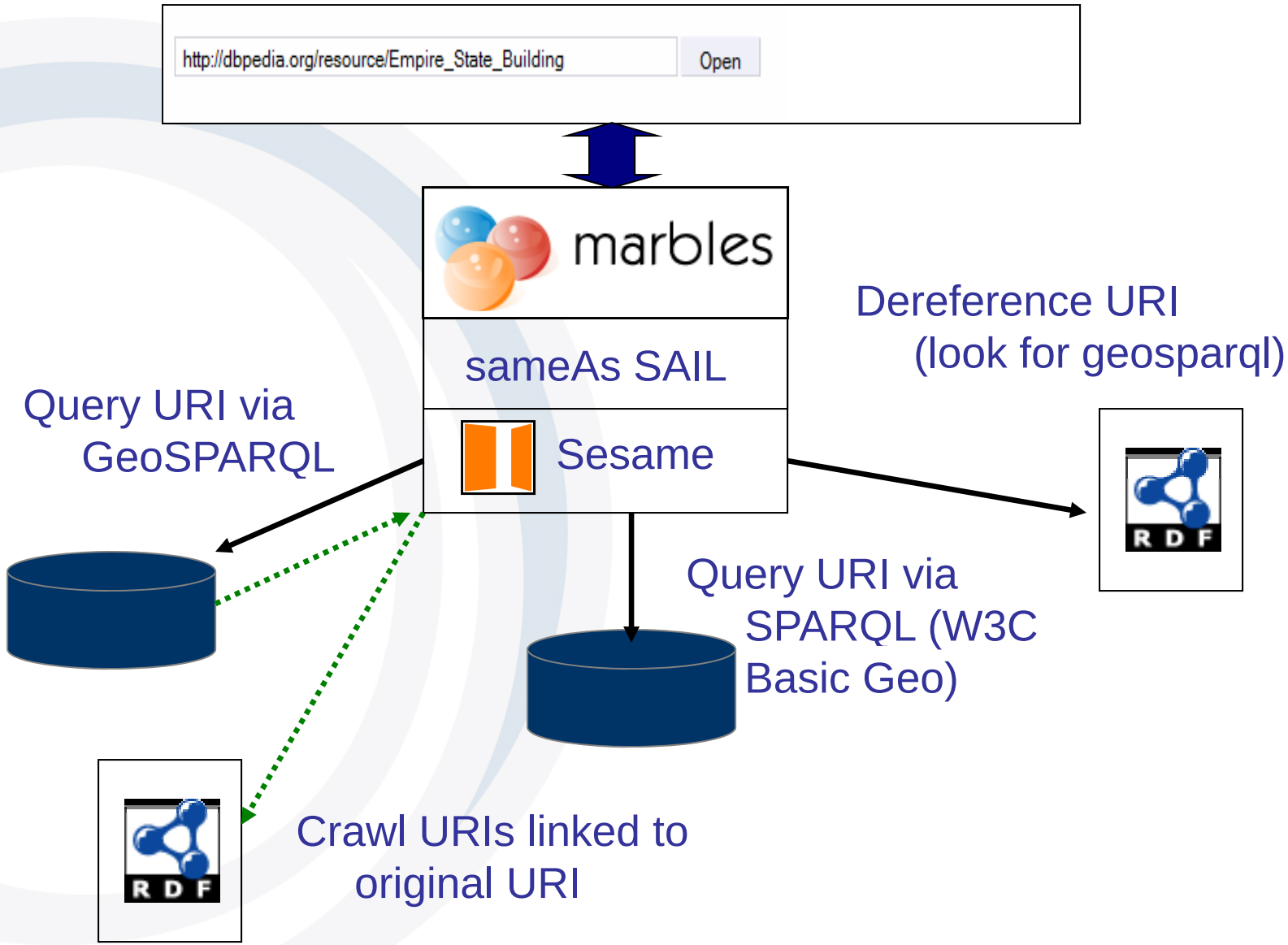
Empire State Building

comment

- Empire State Building er en skyskraper som ligger langs Fifth Avenue, mellom 33. og 34. gate på Manhattan, New York. Bygningen har 102 etasjer og er 381 meter høy (443 meter, inkludert antennen). Bygningen stod ferdig i 1931, og hadde den gang en prislapp på 45 millioner dollar. Bygningen var den første bygning på over 100 etasjer. Det har 6 500 vinduer og 73 heiser.
- El edificio Empire State es un rascacielos situado en la intersección de la Quinta Avenida y West 34th Street, en la ciudad de Nueva York, Estados Unidos. Su nombre deriva del apodo del Estado de Nueva York. Fue el edificio más alto del mundo durante más de cuarenta años, desde su finalización en 1931 hasta 1972, año en que se completó la construcción de la torre norte del World Trade Center.
- The Empire State Building is a 102-story landmark Art Deco skyscraper in New York City at the intersection of Fifth Avenue and West 34th Street. Its name is derived from the nickname for the state of New York, The Empire State. It stood as the world's tallest building for more than forty years, from its completion in 1931 until construction of the World Trade Center's North Tower was completed in 1972.
- O Empire State Building(no Brasil Conhecido Como CIDADE MAESTRA) é um arranha-céu de 102 andares de estilo Art déco localizado na intersecção da 5ª Avenida com a West 34th Street na cidade Nova York. Seu nome deriva do apelido do estado de Nova York. Foi considerada uma das estruturas mais altas do mundo por mais de quarenta anos, desde a sua conclusão em 1931 até que a construção da Torre Norte do World Trade Center foi concluída em 1972.
- Empire State Building on 102-kerroksinen Art Deco -tyylinen pilvenpiirtäjä, joka sijaitsee New Yorkin kaupungissa New Yorkissa Yhdysvalloissa. Vuonna 1931 valmistunut Empire State Building on yksi New Yorkin tunnetuimmista maamerkeistä ja maailman kuuluisimmista pilvenpiirtäjistä.
- Het Empire State Building is met 381 meter de hoogste wolkenkrabber in New York



GeoSPARQL "plugs into" LOD Architecture



Future Work

- Define new serializations
  - KML, GeoJSON
- Define standard methodology for (virtually) converting legacy feature data represented using the general feature model to RDF
- Define OWL axioms for qualitative spatial reasoning
  - ogc:within rdf:type owl:TransitiveProperty
- Qualitative + Quantitative reasoning that utilizes partial geometry information

Summary

- GeoSPARQL Defines:
  - Vocabulary
  - Query functions
  - Query rewrite rules
- 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)
- Thanks to all members of the GeoSPARQL SWG