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

Ordnance Survey

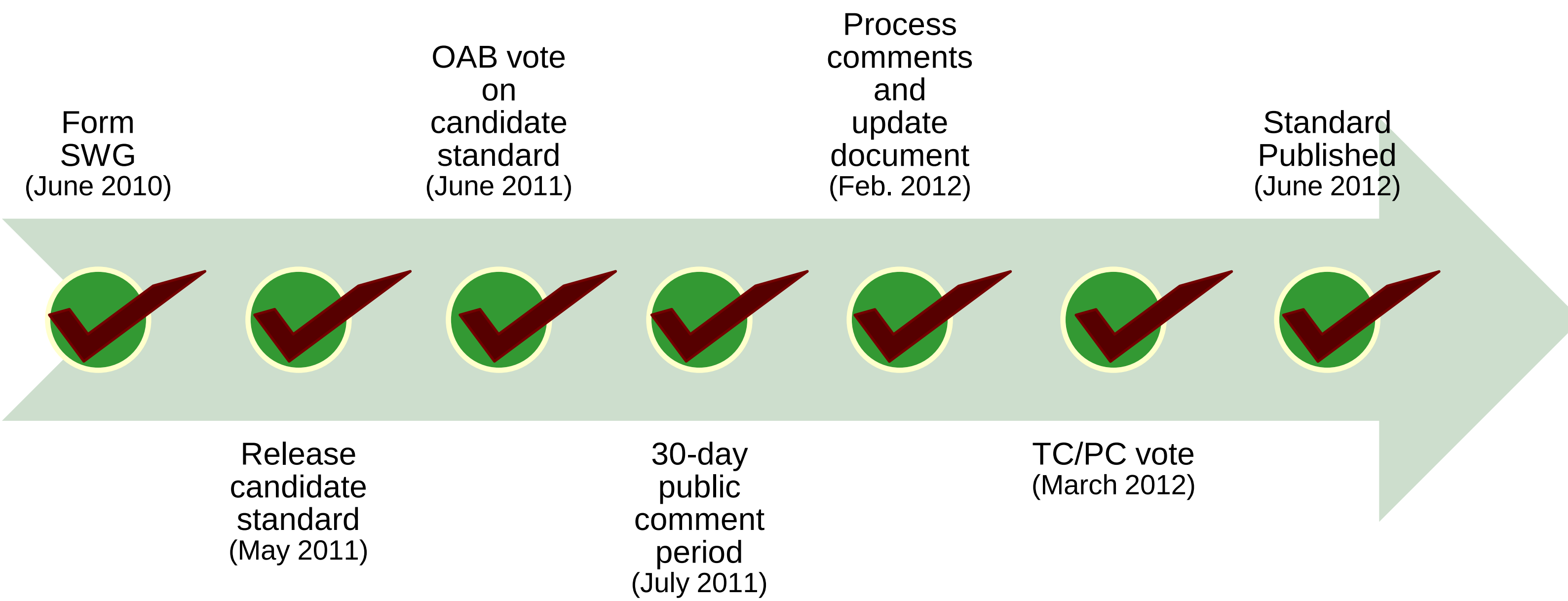
Raytheon

USGS science for a changing world

Traverse Technologies, Inc.

OGC®

Standardization Process



OGC®

Implementations

Raytheon
BBN Technologies

Parliament™

strabon.di.uoa.gr

ORACLE®
DATABASE

OPEN SAHARA | uSeekM

SOME USE CASES FOR GEOSPARQL

OGC®

Linked Geo Data

- Many LOD datasets have geospatial components

Linked GeoData · DBpedia · GeoNames · OpenStreetMap The Free Wiki World Map

Ordnance Survey · data.gov.uk BETA · YAHOO! GEO TECHNOLOGIES · LinkedSensorData

- 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?

Semantic GIS

- 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.

OGC®

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)

OGC®

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

OGC®

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 .
:res3 :bedrooms "3"^^xsd:decimal .
```

SPARQL Query

```
SELECT ?r ?ba ?br
WHERE { ?r rdf:type :House .
        ?r :baths ?ba .
        ?r :bedrooms ?br }
```

Result Bindings

```
?r     | ?ba   | ?br
===================
:res1 | "2.5" | "3"
:res3 | "1.5" | "3"
```

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 .
:res3 :bedrooms "3"^^xsd:decimal .
```

SPARQL Query

```
SELECT ?r ?ba ?br
WHERE { ?r rdf:type :House .
        ?r :baths ?ba .
        ?r :bedrooms ?br
        FILTER (?ba > 2)}
```

Result Bindings

```
?r     | ?ba    | ?br
===================
:res1 | "2.5" | "3"
```

OGC®

Spatial SPARQL QUERY

Spatial RDF Data

```
: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  rdf:type         :House
:res3  :baths           "1.5"^^xsd:decimal .
:res3  :bedrooms        "3"^^xsd:decimal .
:res3  ogc:hasGeometry  :geom3 .
:geom3 ogc:asWKT        "POINT(-122.24 37.47)"^^ogc:wktLiteral .
```

This is what we are standardizing

Vocabulary & Datatypes

GeoSPARQL Query

Find houses within a search polygon

```
SELECT ?r ?ba ?br
WHERE { ?r rdf:type :House .
        ?r :baths ?ba .
        ?r :bedrooms ?br .
        ?r ogc:hasGeometry ?g .
        ?g ogc:asWKT ?wkt
        FILTER(ogcf:sfWithin(?wkt,
                  "POLYGON(…)"^^ogc:wktLiteral)) }
```

Extension Functions

OGC®

GEOSPARQL TECHNICAL DETAILS

OGC®

Components of GeoSPARQL

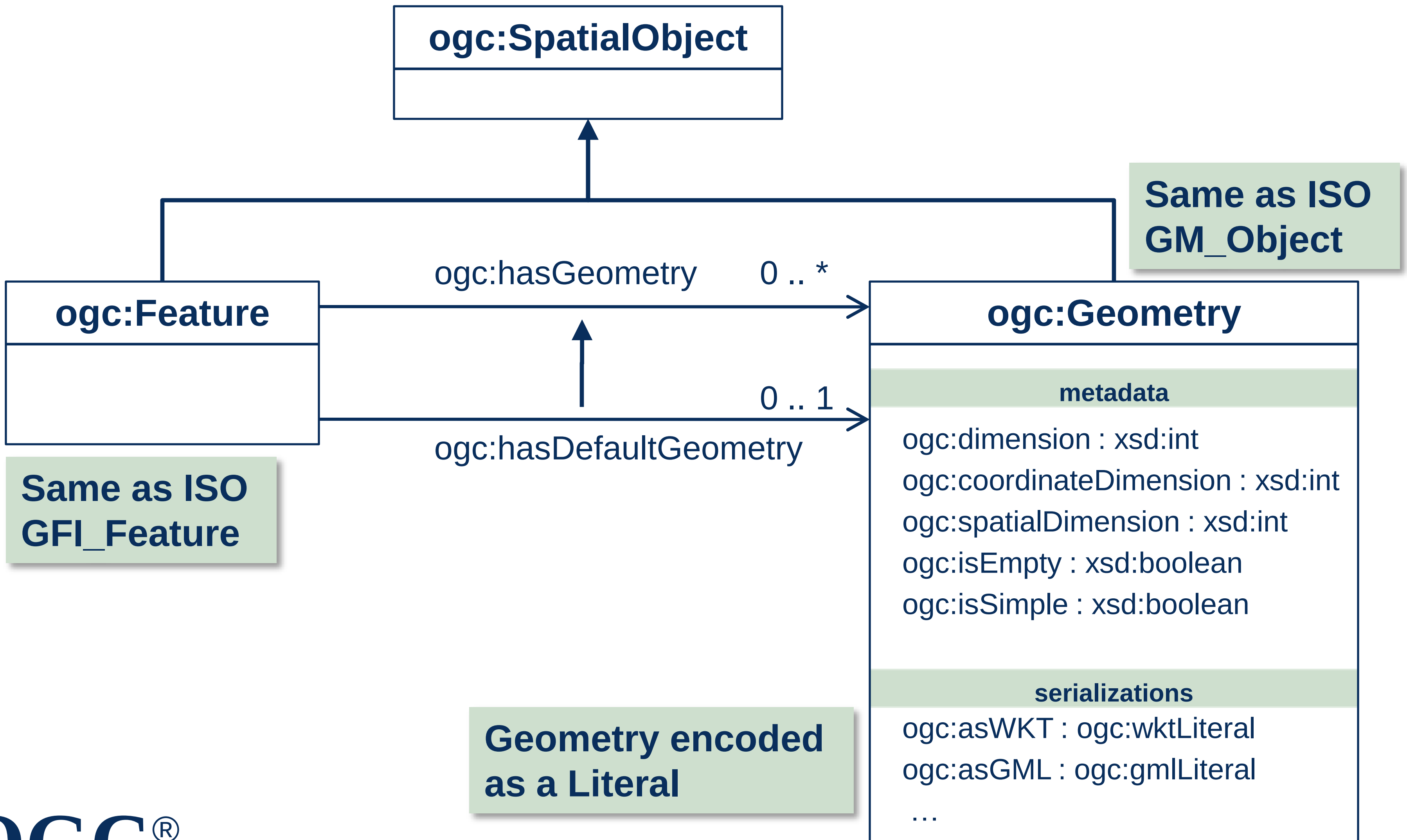
- Vocabulary for Query Patterns
  - Classes
    - Spatial Object, Feature, Geometry
  - Properties
    - Topological relations
    - Links between features and geometries
  - Datatypes for geometry literals
    - ogc:wktLiteral, ogc:gmlLiteral
- Query Functions
  - Topological relations, distance, buffer, intersection, …
- Entailment Components
  - RDFS entailment
  - RIF rules to compute topological relations

OGC®

GEOSPARQL VOCABULARY

OGC®

GeoSPARQL Vocabulary: Basic Classes and Relations



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 [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.

Details of ogc:gmlLiteral

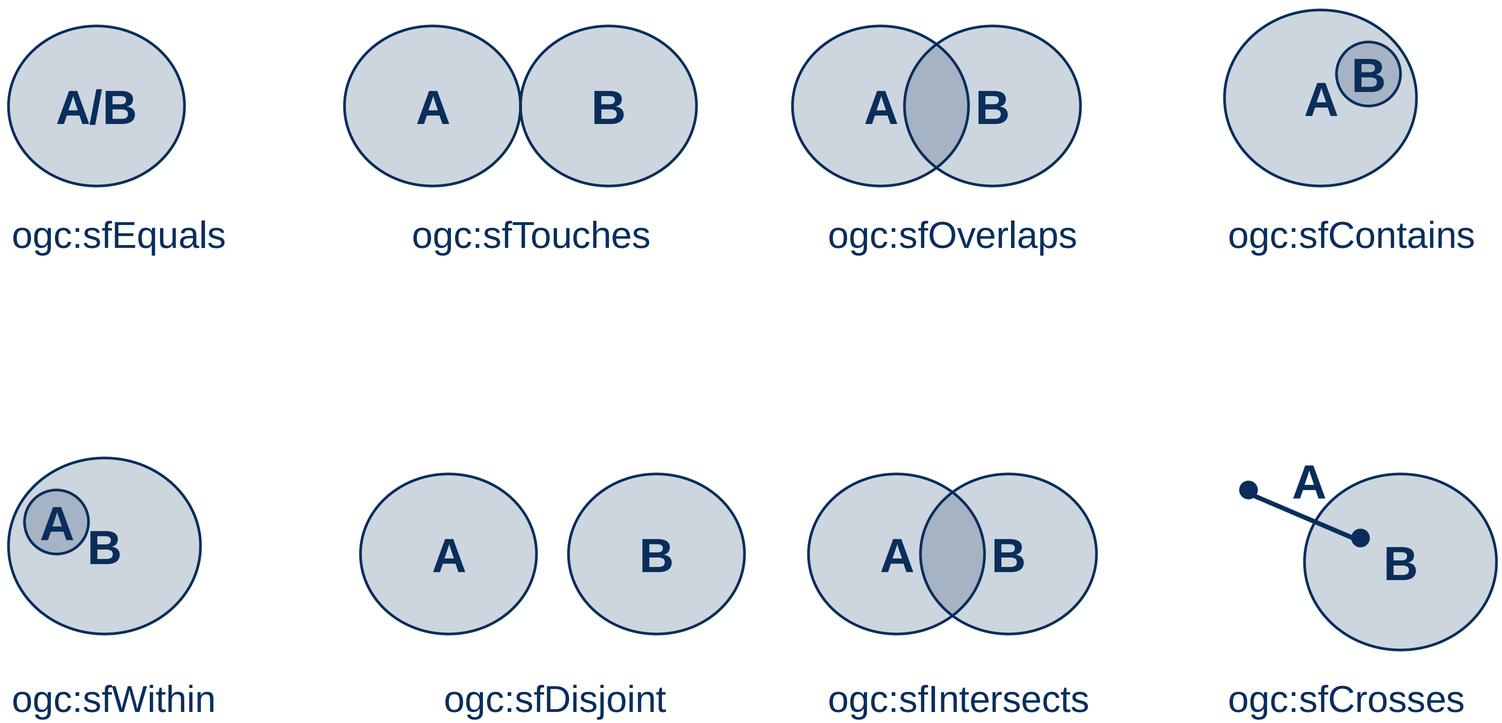
All ogc:gmlLiterals shall consist of a valid element from the GML schema that implements a subtype of GM_Object as defined in [OGC 07-036].

```
"<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:XMLLiterals

OGC®

Topological Relations between ogc:SpatialObject



- Assumes Simple Features Relation Family
- Also support Egenhofer and RCC8

OGC®

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

```
:SanFrancisco          rdf:type      :City .
:UnionSquarePark       rdf:type      :Park .
:UnionSquarePark       :commissioned “1847-01-01”^^xsd:date .
```

Spatial Properties

```
:UnionSquarePark :exactGeometry  :geo1        .
:geo1            ogc:asWKT      “Polygon((…))”^^ogc:wktLiteral .

:SanFrancisco    :exactGeometry  :geo2        .
:geo2            ogc:asWKT      “Polygon((…))”^^ogc:wktLiteral .

:UnionSquarePark ogc:sfWithin   :SanFrancisco .
```

OGC®

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

OGC®

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 }
```

OGC®

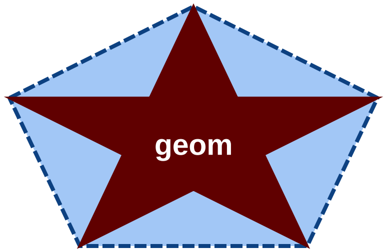
GEOSPARQL QUERY FUNCTIONS

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`

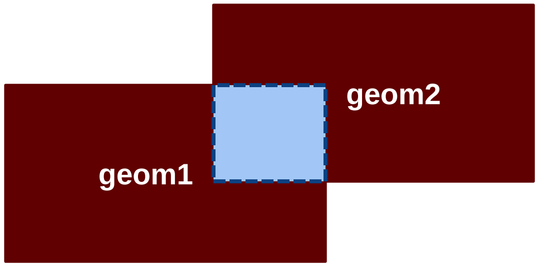
- `ogcf:convexHull(geom: ogc:wktLiteral): ogc:wktLiteral`



OGC®

GeoSPARQL Query Functions

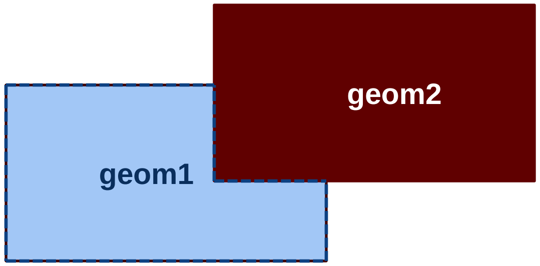
```
– ogcf:intersection(geom1: ogc:wktLiteral,
                    geom2: ogc:wktLiteral): ogc:wktLiteral
```



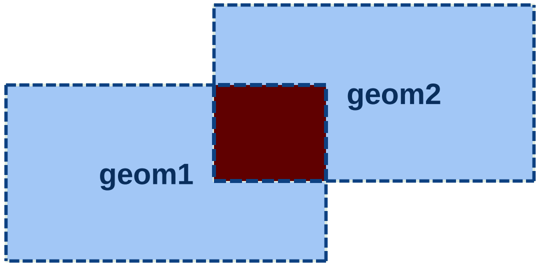

```
– ogcf:union(geom1: ogc:wktLiteral,
             geom2: ogc:wktLiteral): ogc:wktLiteral
```

GeoSPARQL Query Functions

```
– ogcf:difference(geom1: ogc:wktLiteral,
                  geom2: ogc:wktLiteral): ogc:wktLiteral
```




```
– ogcf:symDifference(geom1: ogc:wktLiteral,
                     geom2: ogc:wktLiteral): ogc:wktLiteral
```



OGC®

GeoSPARQL Query Functions

– `ogcf:envelope(geom: ogc:wktLiteral): ogc:wktLiteral`

– `ogcf:boundary(geom1: ogc:wktLiteral): ogc:wktLiteral`

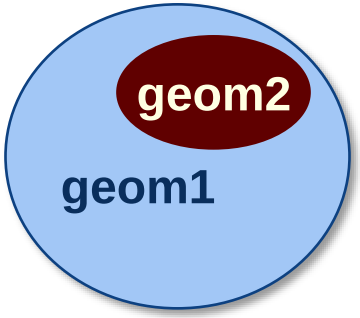
– `ogcf:getSRID(geom: ogc:wktLiteral): xsd:anyURI`

OGC®

GeoSPARQL Topological 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 Topological Query Functions

```
– ogcf:sfEquals(geom1: ogc:wktLiteral,
                geom2: ogc:wktLiteral): xsd:boolean
– ogcf:sfDisjoint(geom1: ogc:wktLiteral,
                  geom2: ogc:wktLiteral): xsd:boolean
– ogcf:sfIntersects(geom1: ogc:wktLiteral,
                    geom2: ogc:wktLiteral): xsd:boolean
– ogcf:sfTouches(geom1: ogc:wktLiteral,
                 geom2: ogc:wktLiteral): xsd:boolean
– ogcf:sfCrosses(geom1: ogc:wktLiteral,
                 geom2: ogc:wktLiteral): xsd:boolean
– ogcf:sfWithin(geom1: ogc:wktLiteral,
                geom2: ogc:wktLiteral): xsd:boolean
– ogcf:sfContains(geom1: ogc:wktLiteral,
                  geom2: ogc:wktLiteral): xsd:boolean
– ogcf:sfOverlaps(geom1: ogc:wktLiteral,
                  geom2: ogc:wktLiteral): xsd:boolean
```

Assumes Simple Features Relation Family

OGC®

Example Query

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 .
        ?parcel       :exactGeometry    ?pGeo .
        ?pGeo         ogc:asWKT         ?pWKT .
        :District1    :exactGeometry    ?dGeo .
        ?dGeo         ogc:asWKT         ?dWKT .
        :City1        :extent           ?cGeo .
        ?cGeo         ogc:asWKT         ?cWKT .
        FILTER(ogcf:sfWithin(?pWKT,
                 ogcf:intersection(?dWKT,?cWKT)))}
```

OGC®

GEOSPARQL ENTAILMENT COMPONENTS

OGC®

GeoSPARQL RDFS Entailment Extension

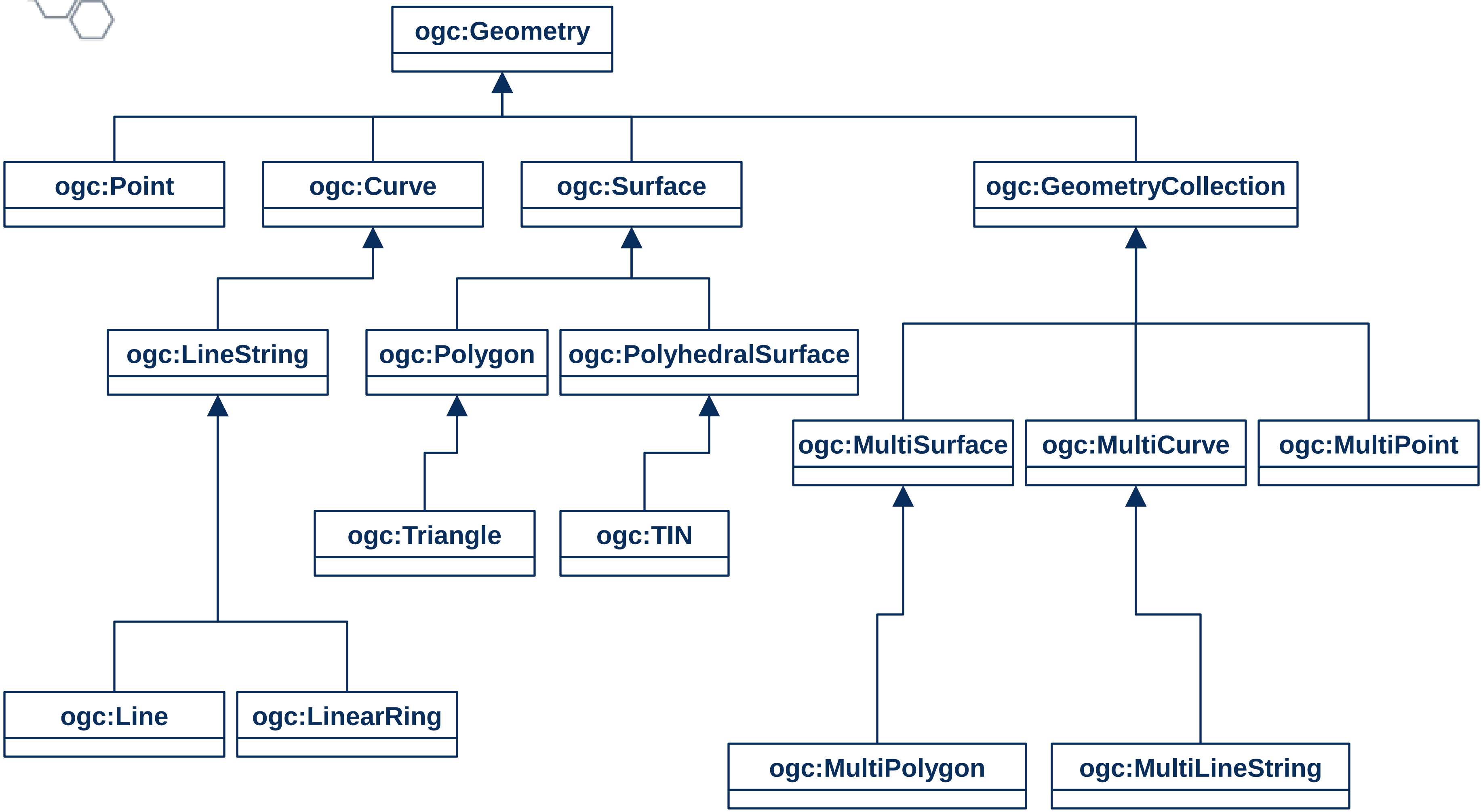
Main Requirements:

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

OGC®

Simple Features Geometry Types



OGC®

GeoSPARQL Query Rewrite Extension

Find all water bodies within New Hampshire

```
SELECT ?water
WHERE { ?water  rdf:type        :WaterBody .
        ?water  ogc:rcc8Within    :NH }
```

Same Query Specification

Qualitative → RCC8 Backward Chaining

Quantitative ↓

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

Query Rewrite

Specified with a RIF rule

GeoSPARQL Query Rewrite Extension

Main Requirement:

Basic graph pattern matching shall use the semantics defined by the **RIF Core Entailment Regime** [W3C SPARQL Entailment] for the RIF rules [W3C RIF Core] `geor:sfEquals`, `geor:sfDisjoint`, `geor:sfIntersects`, `geor:sfTouches`, `geor:sfCrosses`, `geor:sfWithin`, `geor:sfContains`, `geor:sfOverlaps`.

OGC®

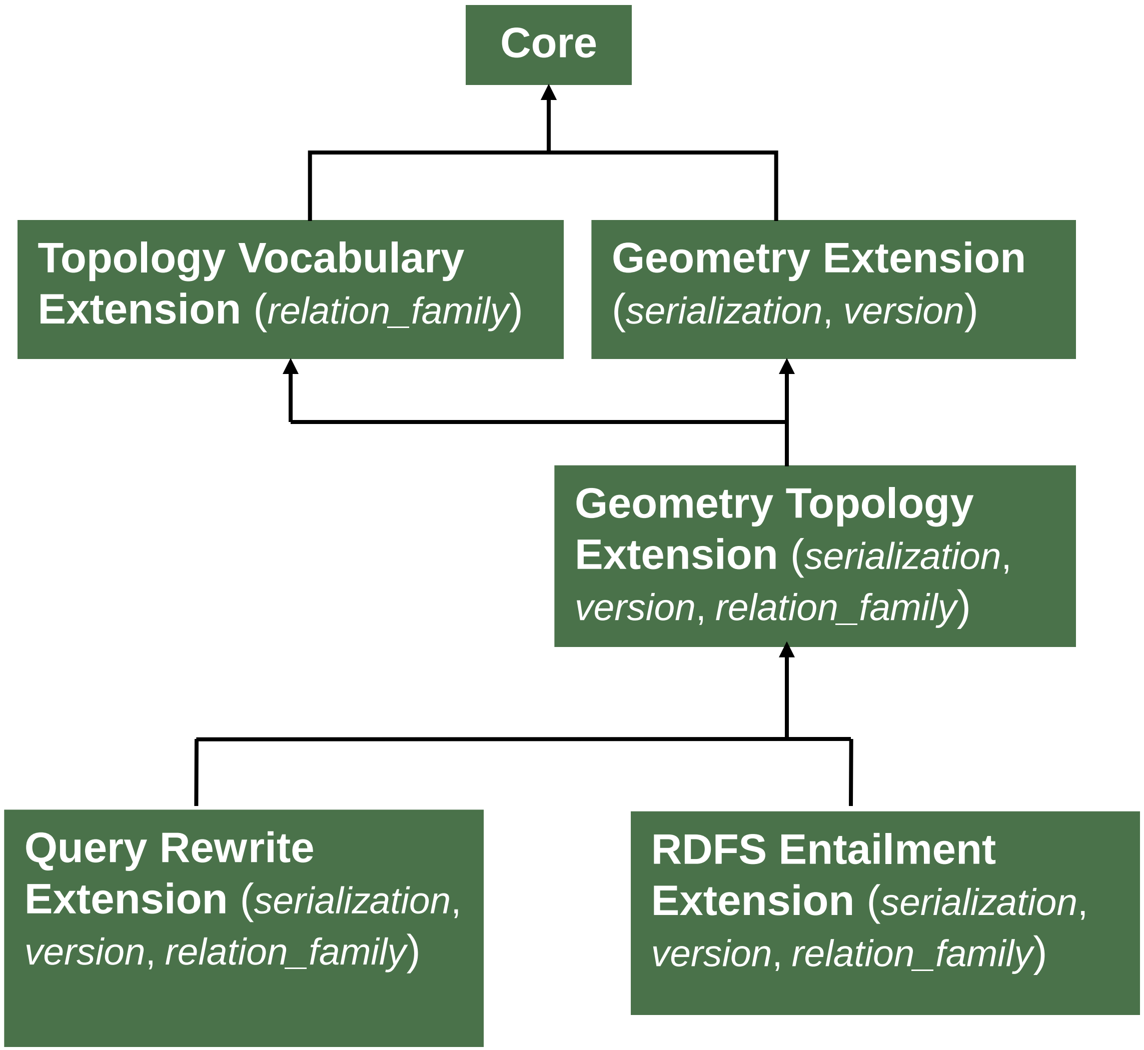
Query Rewrite Rules

- 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)))
  )
)
```

OGC®

Summary of Conformance Classes



Parameters

- Serialization
  - *WKT*
  - *GML*

  Determines geometry classes and geometry literal datatype

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

  Determines topology properties and topology functions

IMPLEMENTATION CONSIDERATIONS

OGC®

Implementing Spatial Operations

- These are standard OGC operators that have been around for some time
- Lots of infrastructure available
  - Open Source
  - Commercial

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

OGC®

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)

OGC®

Thanks to all members of the GeoSPARQL SWG !

QUESTIONS?

OGC®