Geospatial data in RDF – stSPARQL

Presenter: Kostis Kyzirakos

Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens

Outline

- Main idea
- Early works
- The data model stRDF
- Examples of publicly available linked geospatial data
- The query language stSPARQL

Main idea

How do we represent and query geospatial information in the Semantic Web?

Extend RDF to take into account the geospatial dimension.

Extend SPARQL to query the new kinds of data.

Early works

SPAUK

[Kolas and Self, 2007]

- Geometric attributes of a resource are represented by:
  - introducing a **blank node** for the geometry
  - specifying the geometry using **GML vocabulary**
  - associating the blank node with the resource using **GeoRSS vocabulary**
- Queries are expressed in SPARQL utilizing appropriate geometric vocabularies and ontologies (e.g., the topological relationships of RCC-8).
- Introduces a new **`PREMISE`** clause in SPARQL to specify spatial geometries to be used in a query
- Use some form of the **`DESCRIBE`** query form of SPARQL for asking queries about geometries

Early works

SPARQL-ST

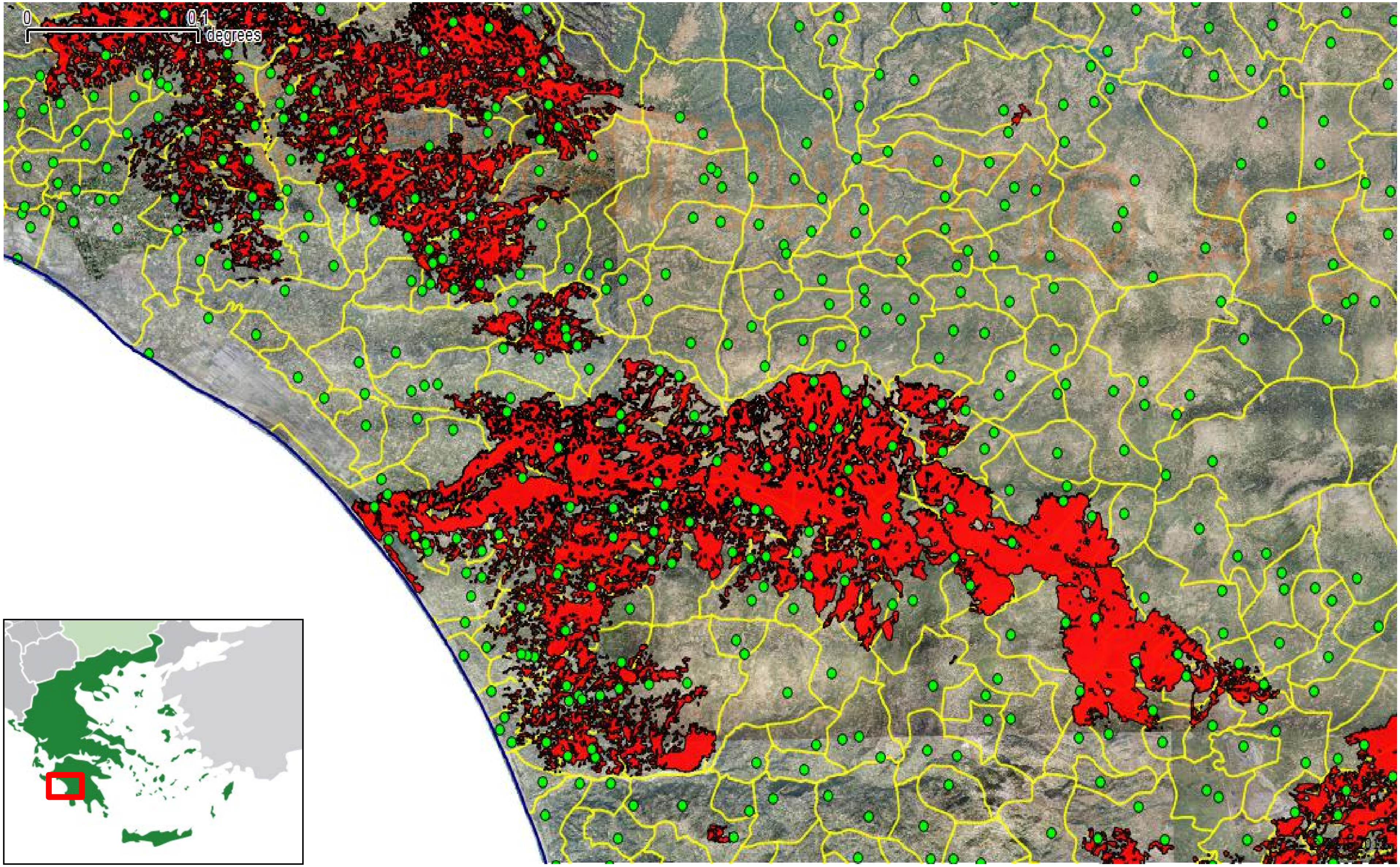
[Perry, 2008]

- Assumes a particular upper ontology expressed in RDFS for modeling **theme**, **space** and **valid time**.
- Spatial geometries in SPARQL-ST are specified by **sets of RDF triples** that give various details of the geometry.
- SPARQL-ST provides a set of built-in spatial conditions that can be used in **`SPATIAL FILTER`** clauses to constrain the geometries that are returned as answers to queries.

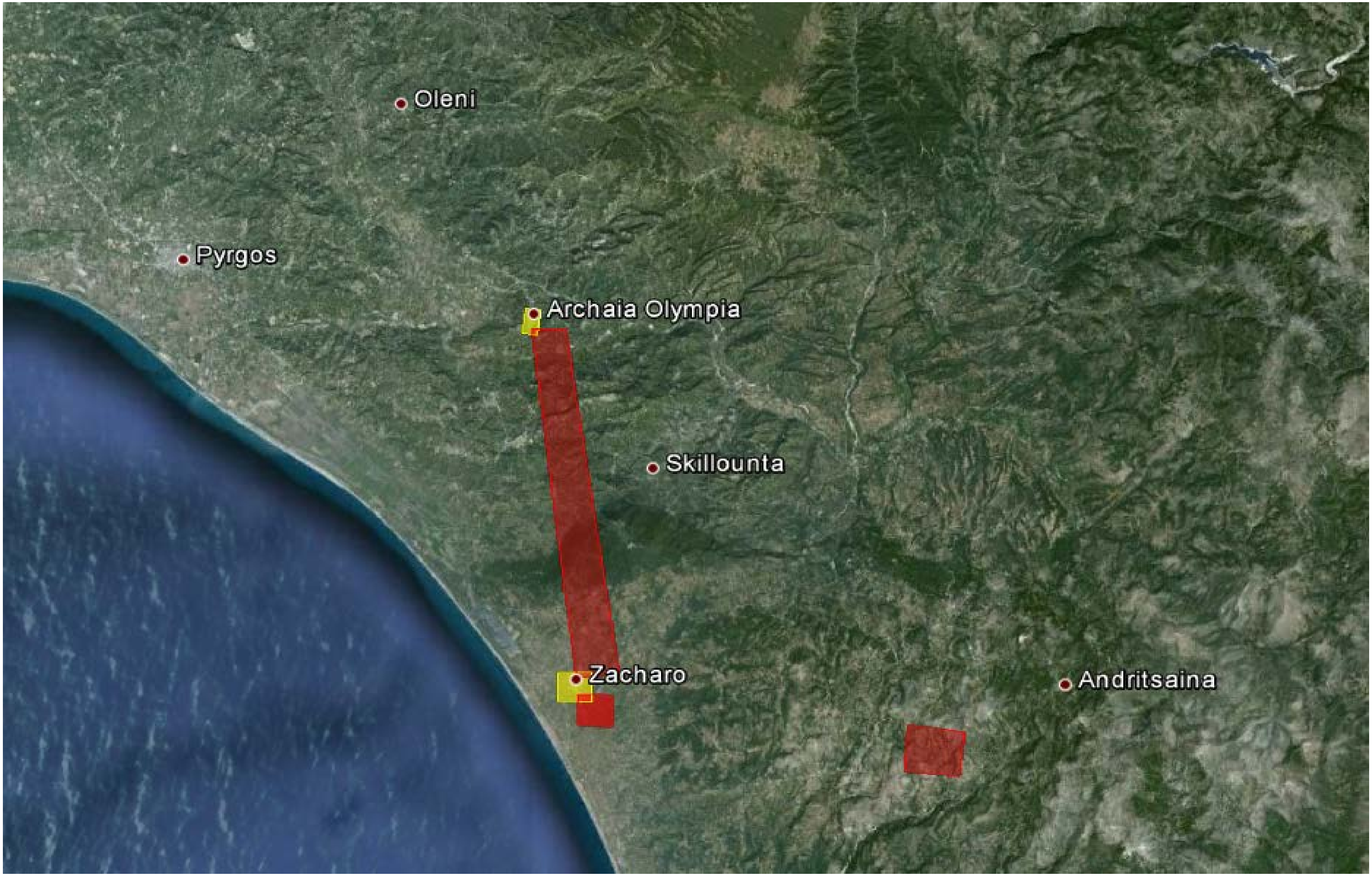
stRDF and stSPARQL

- Similar approach to SPARQL-ST (**theme**, **space** and **valid time** can be represented) [Koubarakis and Kyzirakos, 2010]
- **Linear constraints** are used to represent geometries
- Constraints are represented using literals of an appropriate datatype
- Formal approach
- New version to be presented today uses **OGC standards** to represent and query geometries

Example

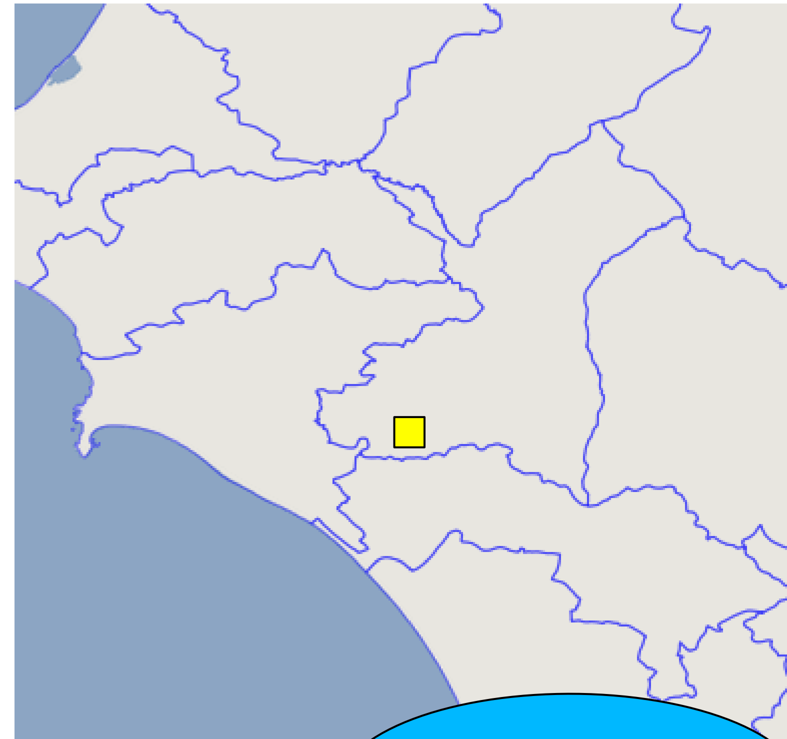


Example with simplified geometries



Example in stRDF

```
geonames:Olympia
  geonames:name "Ancient Olympia";
  owl:sameAs dbpedia:Olympia_Greece;
  rdf:type dbpedia:Community .
```



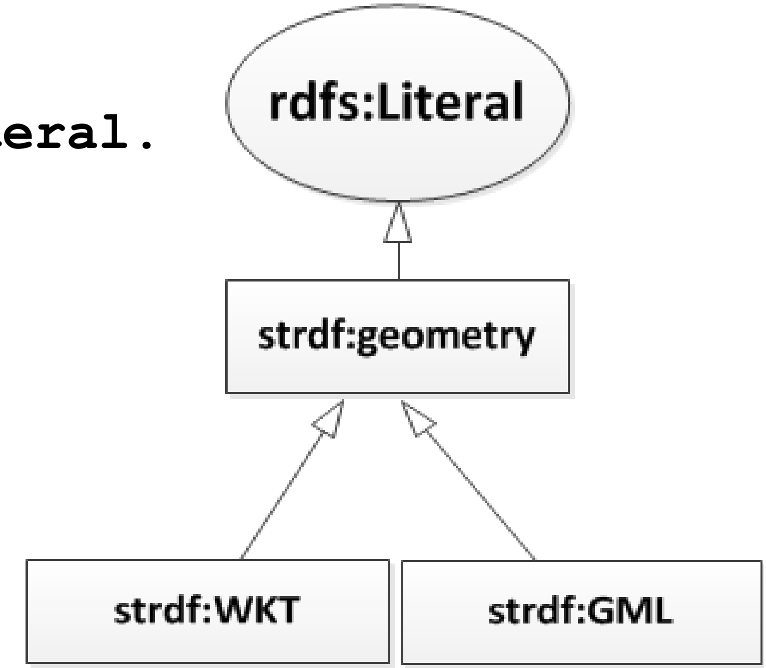
Spatial literal

```
geonames:Olympia strdf:hasGeometry
           "POLYGON((21.5 18.5, 23.5 18.5,
                      23.5 21, 21.5 21, 21.5 18.5));
  <http://www.opengis.net/def/crs/EPSG/0/4326>"^^
  strdf:WKT .
```

Spatial data type

The stRDF Data Model

```
strdf:geometry rdf:type rdfs:Datatype;
               rdfs:subClassOf rdfs:Literal.
```




```
strdf:WKT   rdf:type rdfs:Datatype;
            rdfs:subClassOf   rdfs:Literal;
            rdfs:subClassOf   strdf:geometry.

strdf:GML   rdf:type rdfs:Datatype;
            rdfs:subClassOf   rdfs:Literal;
            rdfs:subClassOf   strdf:geometry.
```

The stRDF Data Model

We define the datatypes **`strdf:WKT`** and **`strdf:GML`** that can be used to represent spatial objects using the WKT and GML serializations.

- **Lexical space:** the finite length sequences of characters that can be produced from the WKT and GML specifications.
  - Literals of type **`strdf:WKT`** consist of an optional URI identifying the coordinate reference system used.

e.g.,
```
"POINT(21 18);
 <http://www.opengis.net/def/crs/EPSG/0/4326>"
^^strdf:WKT
```

The stRDF Data Model

- **Value space**: the set of geometry values defined in the WKT and GML standard that is a subset of the powerset of $\mathbb{R}^2$ and $\mathbb{R}^3$.

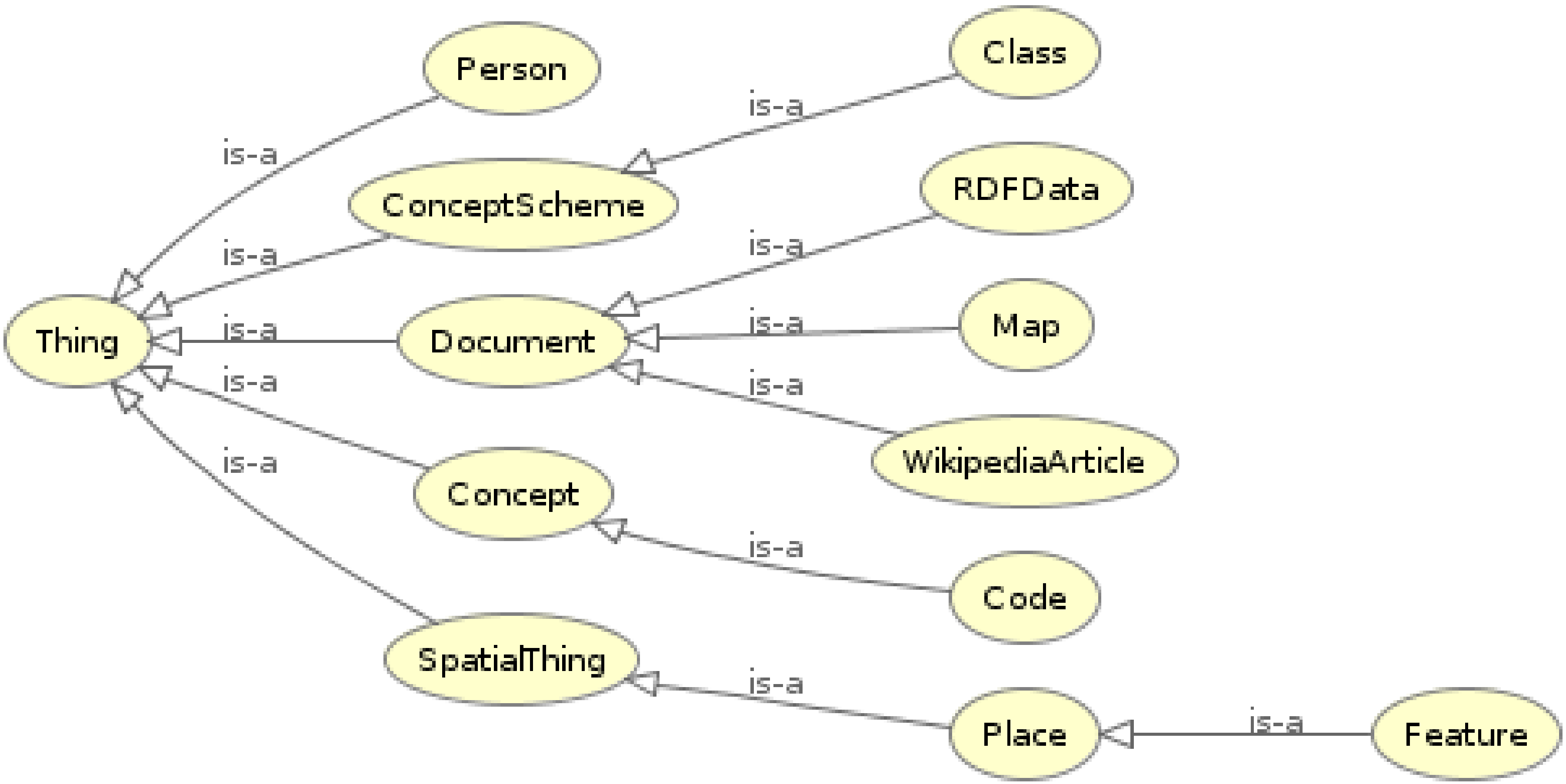
- **Lexical-to-value mapping:** takes into account that the vector-based model is used for representing geometries.

- The datatype **`strdf:geometry`** is the union of the datatypes **`strdf:WKT`** and **`strdf:GML`**.

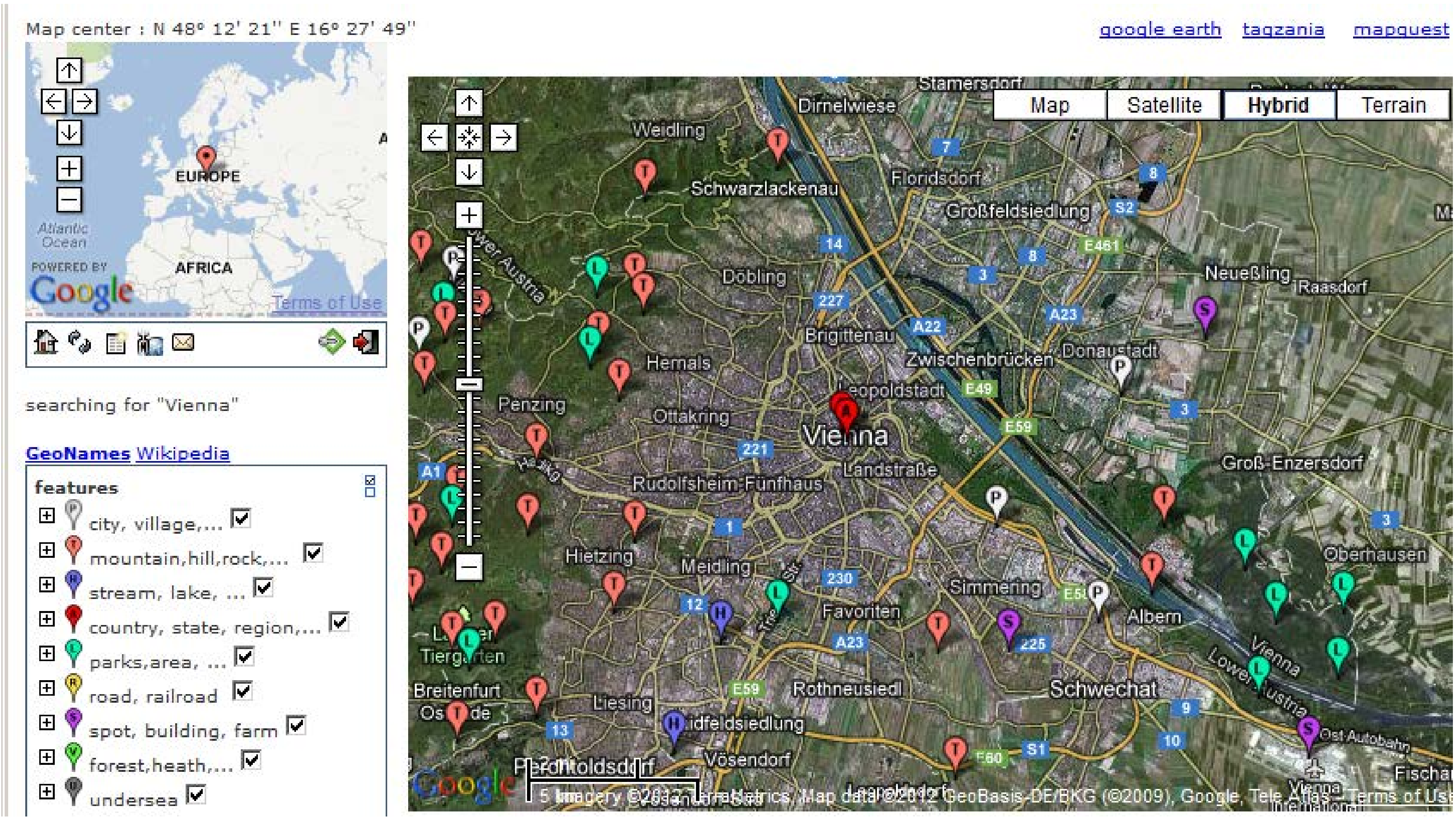
Examples of publicly available linked geospatial data

- Geonames
- Greek Administrative Geography
- Corine Land Use / Land Cover
- Burnt Area Products

Geonames


Person
Class
is-a
is-a
ConceptScheme
RDFData
is-a
is-a
Map
Thing
is-a
Document
is-a
is-a
is-a
WikipediaArticle
is-a
Concept
is-a
Code
SpatialThing
is-a
Place
is-a
Feature

Geonames



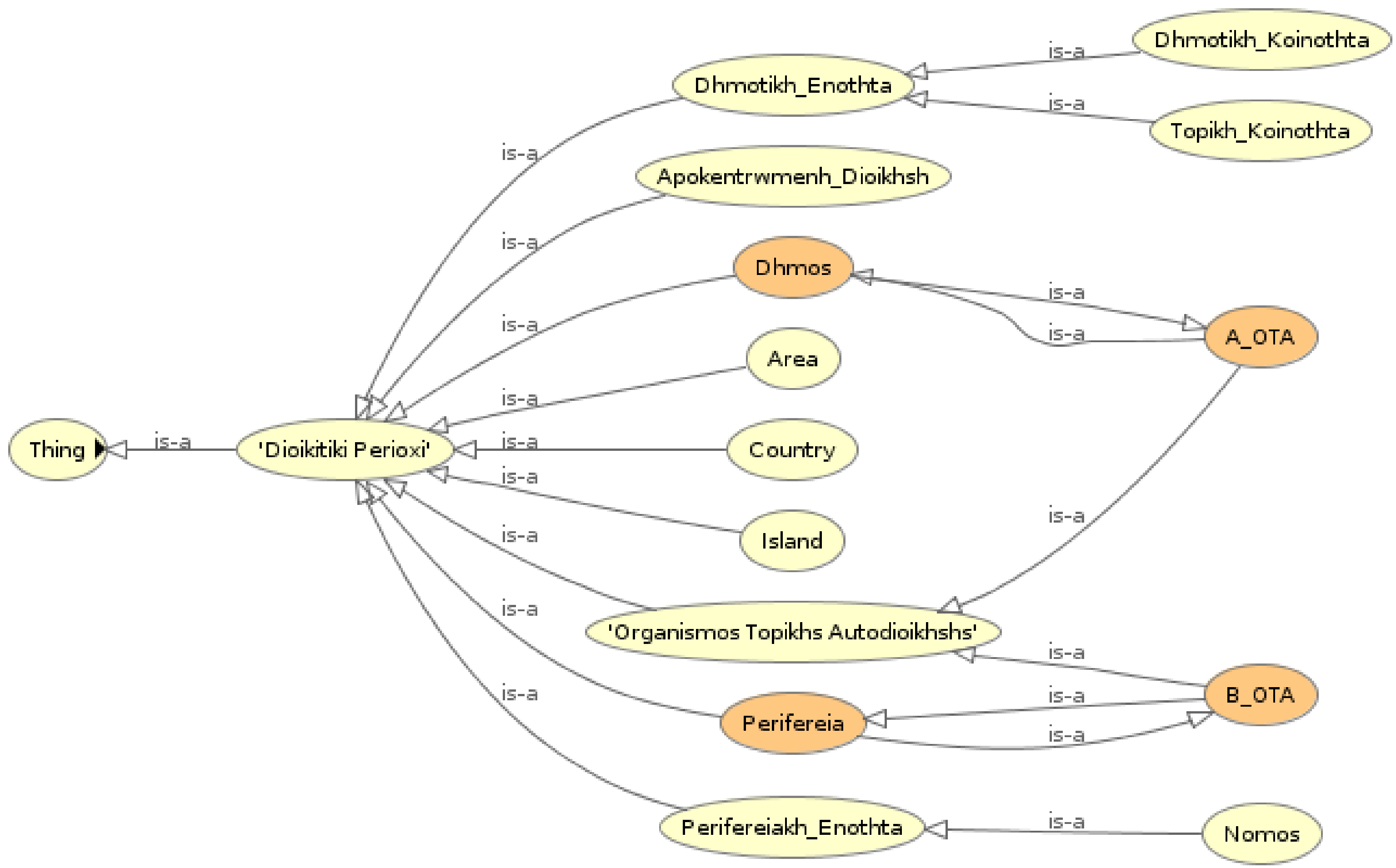
Map center : N 48° 12' 21" E 16° 27' 49"
google earth
tagzania
mapquest
EUROPE
AFRICA
Atlantic Ocean
POWERED BY Google
Terms of Use
searching for "Vienna"
GeoNames Wikipedia
features
city, village,...
mountain,hill,rock,...
stream, lake, ...
country, state, region,...
parks,area, ...
road, railroad
spot, building, farm
forest,heath,...
undersea
Map
Satellite
Hybrid
Terrain
Vienna
Dirnelwiese
Weidling
Schwarzlackenau
Floridsdorf
Großfeldsiedlung
Döbling
Brigittenau
Zwischenbrücken
Donaustadt
Neueßling
Raasdorf
Hernals
Penzing
Ottakring
Leopoldstadt
Landstraße
Rudolfsheim-Fünfhaus
Groß-Enzersdorf
Hietzing
Meidling
Simmering
Favoriten
Albern
Oberhausen
Breitenfurt
Liesing
Rothneusiedl
Schwechat
Vösendorf

Geonames

```
gn:2761333
   rdf:type geonames:Feature;
   geonames:officialName "Vienna"@en;
   geonames:name "Politischer Bezirk Wien (Stadt)";
   geonames:countryCode "AT";
   wgs84_pos:lat "48.2066";
   wgs84_pos:long "16.37341".
   geonames:parentCountry gn:2782113;

gn:2782113
   geonames:name "Austria";
   geonames:altName    "Republic of Austria"@en,
                       "Republik Osterreich"@de,
                       "Αυστρία"@el.
```

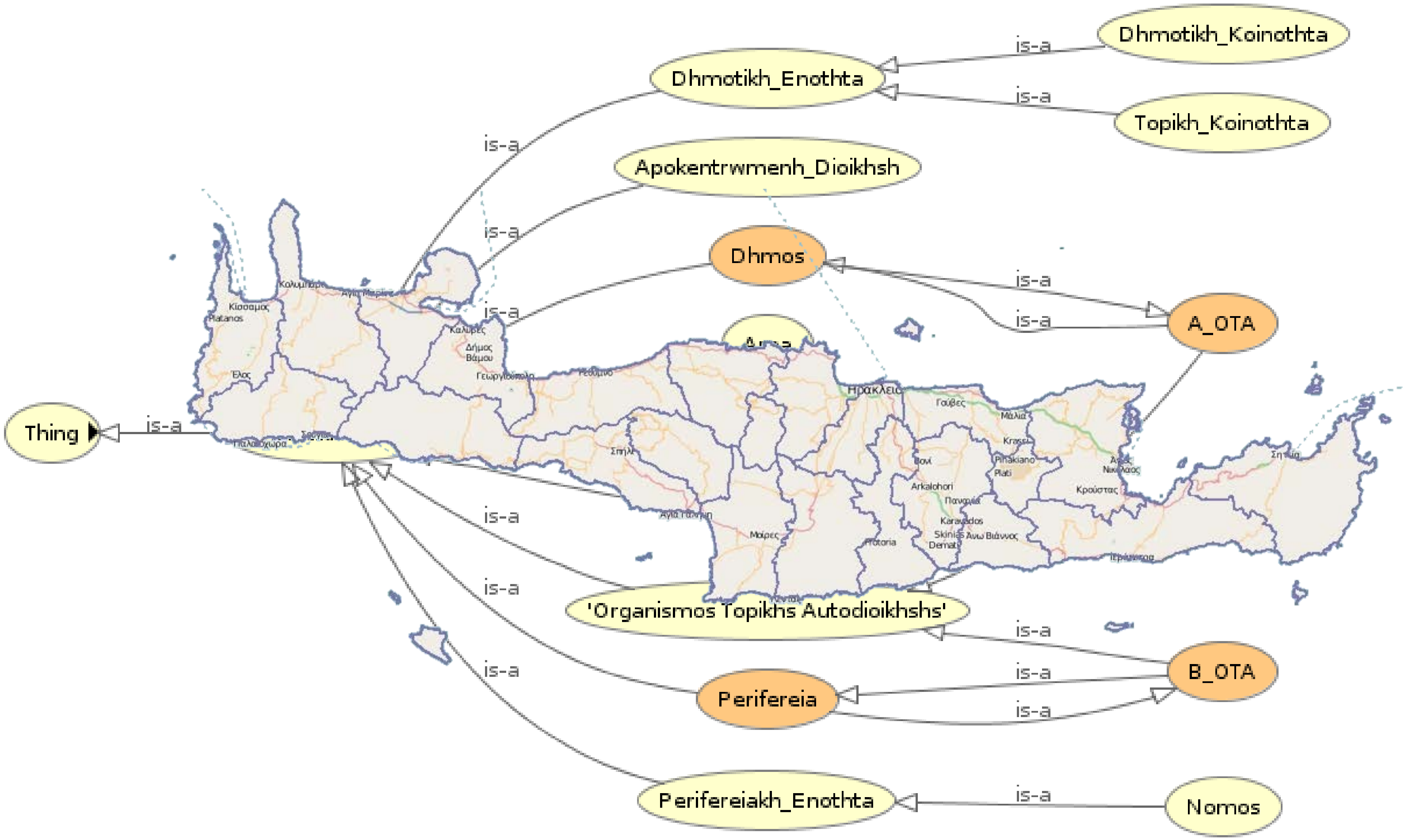
Greek Administrative Geography

Kallikrates ontology

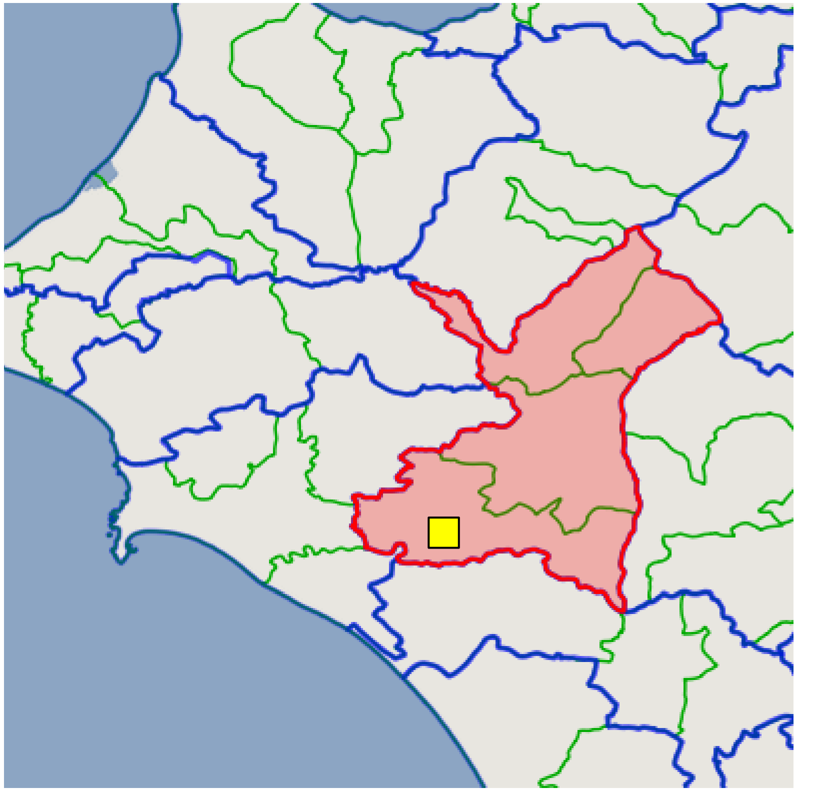


Greek Administrative Geography

Kallikrates ontology

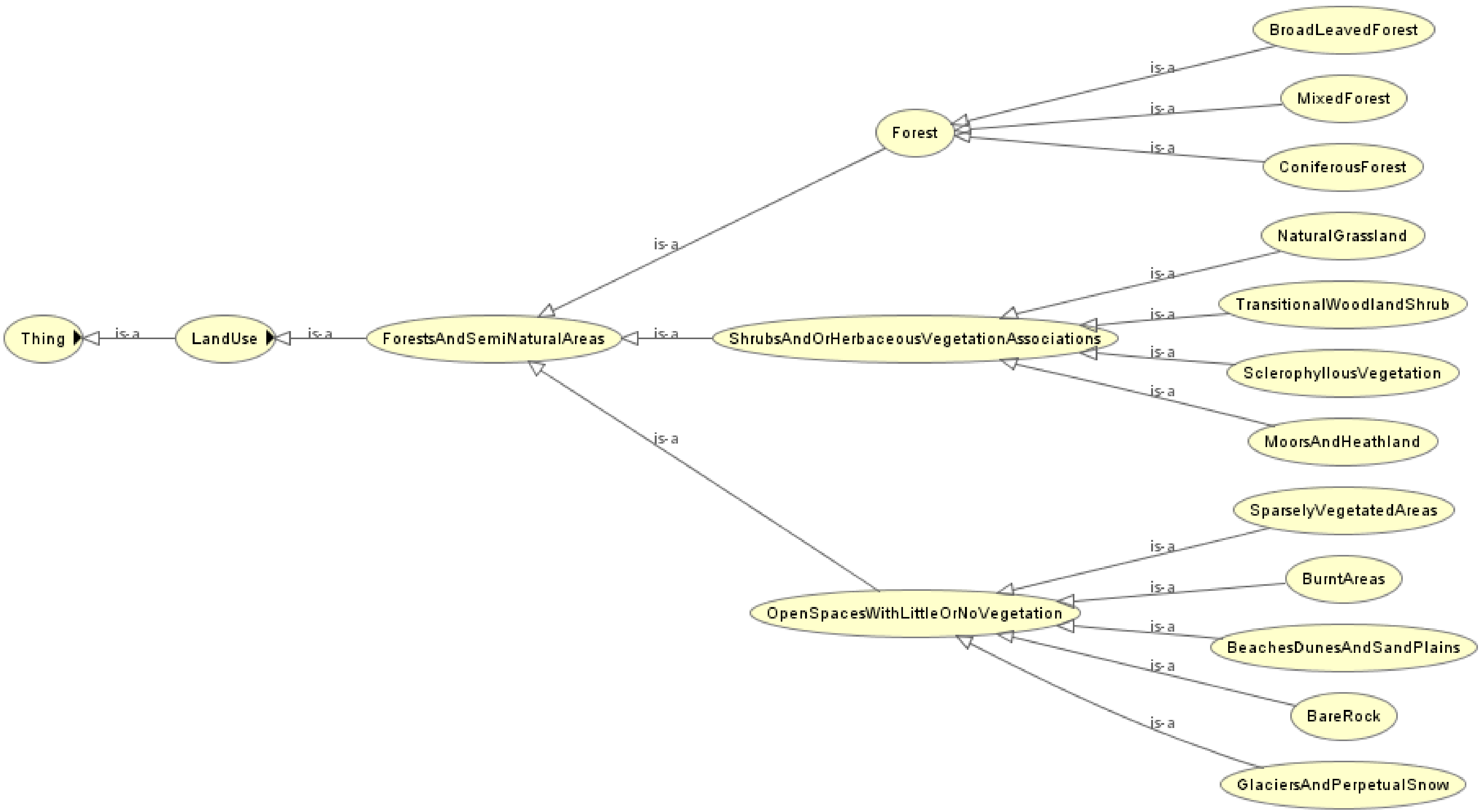


Greek Administrative Geography

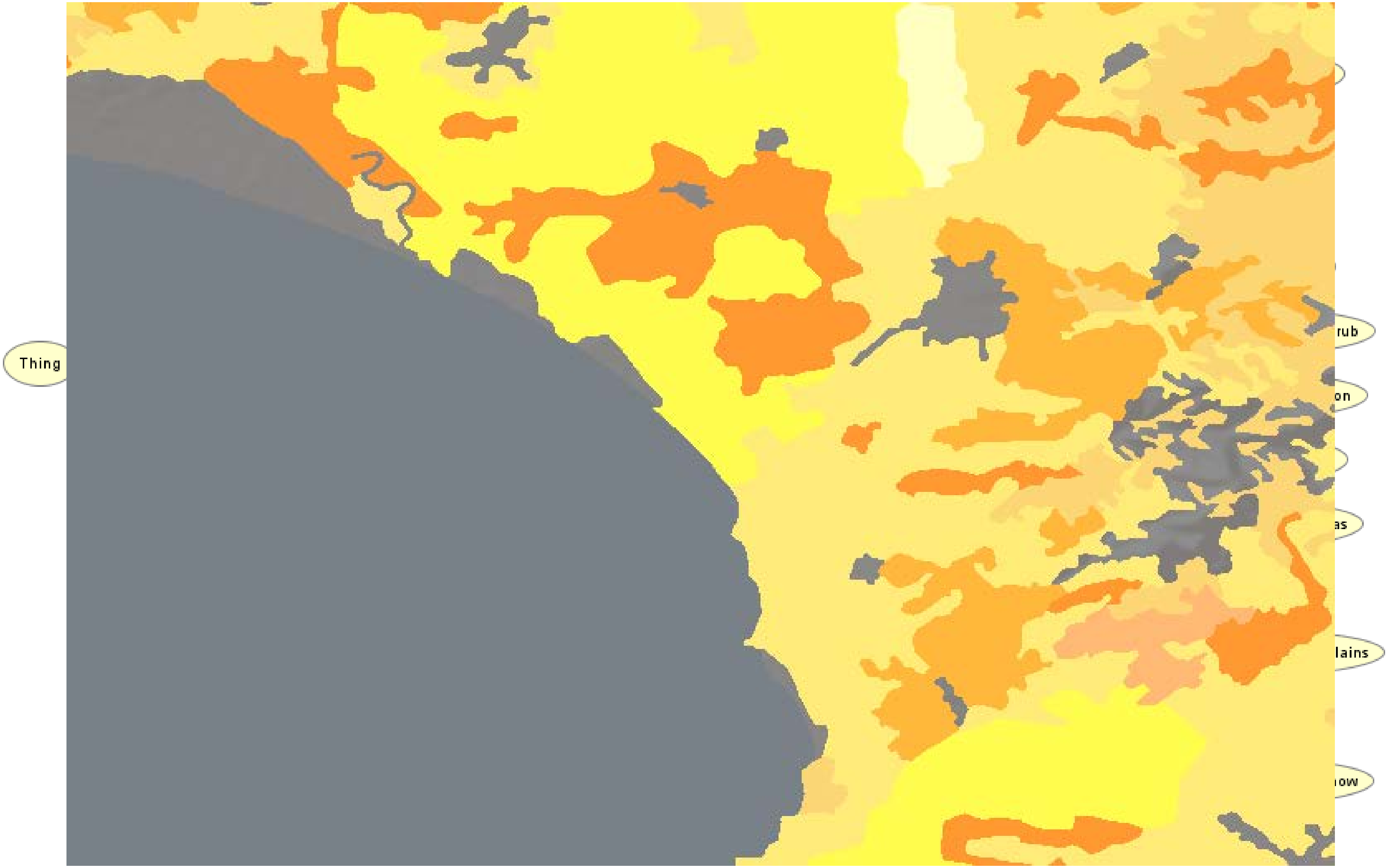


```
gag:Olympia
  rdf:type gag:Community;
  geonames:name "Ancient Olympia";
  gag:population "184"^^xsd:int;
  strdf:hasGeometry "POLYGON
  (((25.37 35.34,…)))"^^strdf:WKT.

gag:OlympiaBorough
  rdf:type gag:Borough;
  rdfs:label "Borough of
              Ancient Olympia".

gag:OlympiaMunicipality
  rdf:type gag:Municipality;
  rdfs:label "Municipality of
              Ancient Olympia".

gag:Olympia gag:isPartOf gag:OlympiaBorough .

gag:OlympiaBorough gag:isPartOf gag:OlympiaMunicipality.
```

Corine Land Use / Land Cover



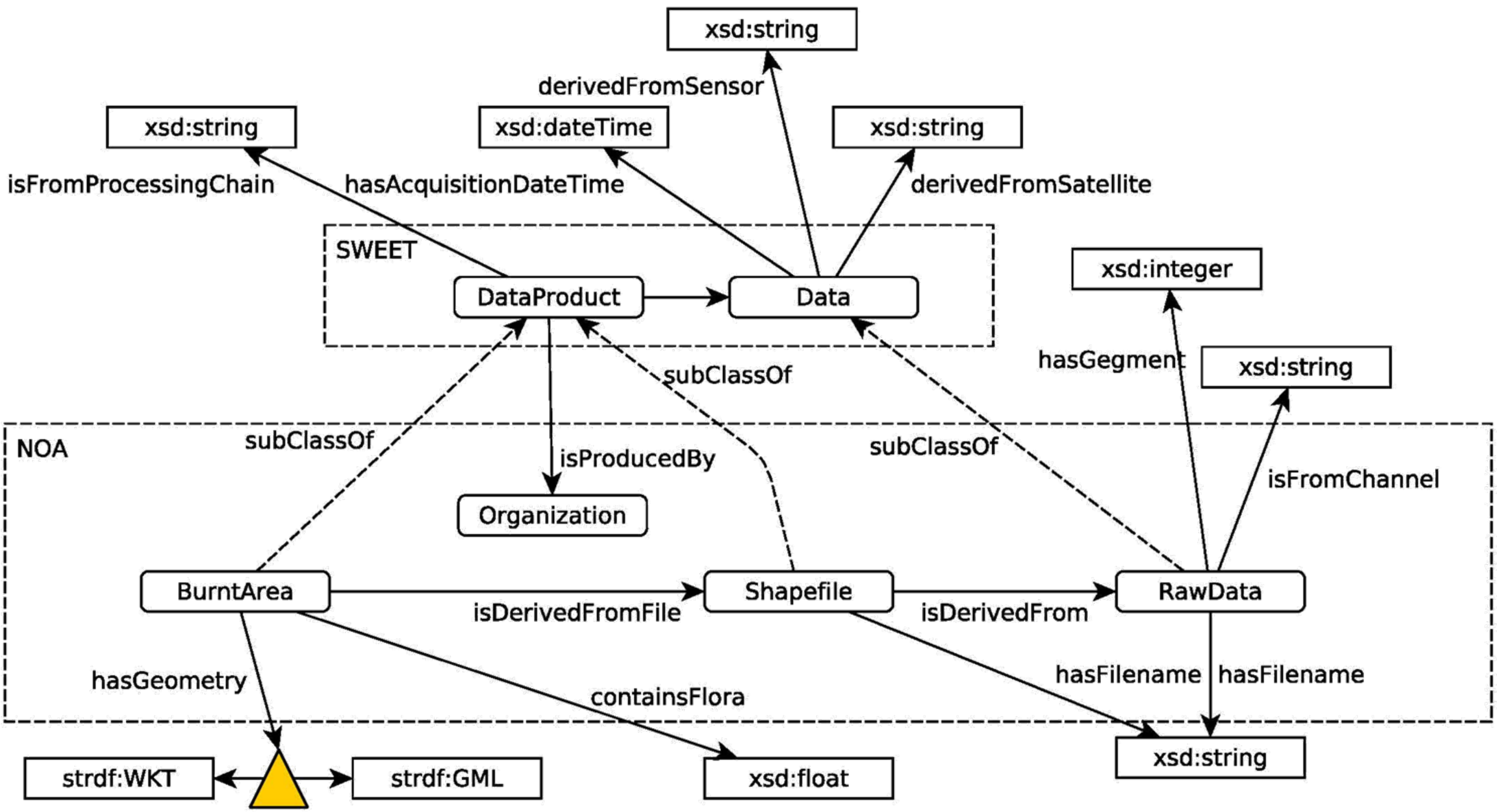
Corine Land Use / Land Cover



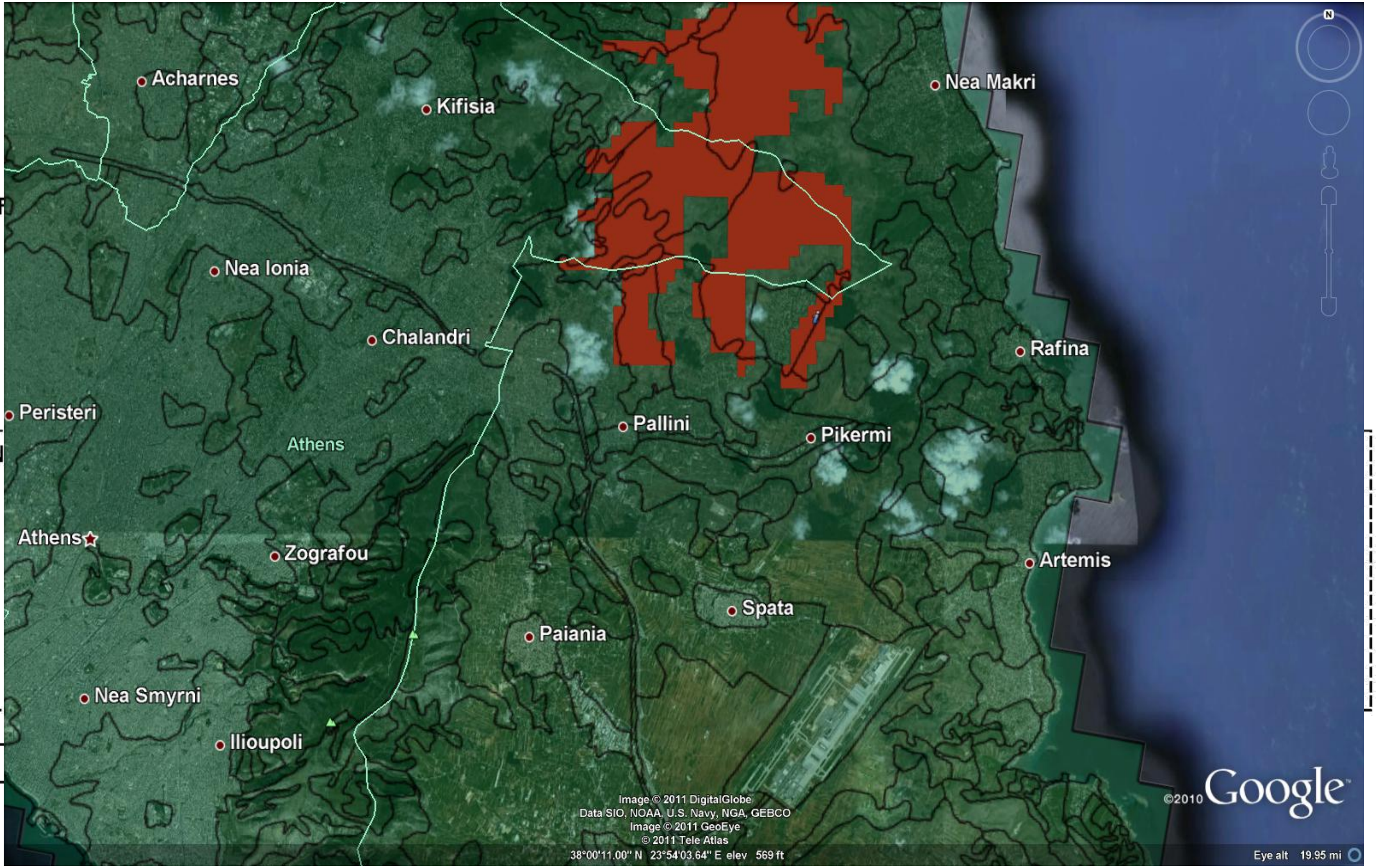
Corine Land Use / Land Cover

```
noa:Area_24015134
  rdf:type noa:Area ;
  noa:hasCode "312"^^xsd:decimal;
  noa:hasID "EU-203497"^^xsd:string;
  noa:hasArea_ha "255.5807904"^^xsd:double;
  strdf:hasGeometry "POLYGON((15.53 62.54,
                              …))"^^strdf:WKT;
  noa:hasLandUse noa:ConiferousForest
```

Burnt Area Products



Burnt Area Products



Burnt Area Products

```
noa:ba_15
  rdf:type noa:BurntArea;
  noa:isProducedByProcessingChain
         "static thresholds"^^xsd:string;
  noa:hasAcquisitionTime
         "2010-08-24T13:00:00"^^xsd:dateTime;

  strdf:hasGeometry "MULTIPOLYGON(((
  393801.42 4198827.92, ..., 393008 424131)));
  <http://www.opengis.net/def/crs/
                    EPSG/0/2100>"^^strdf:WKT.
```

stSPARQL: Geospatial SPARQL 1.1

We define a SPARQL extension function for each function defined in the OpenGIS Simple Features Access standard

Basic functions

- Get a property of a geometry
  ```
  xsd:int strdf:Dimension(strdf:geometry A)
  xsd:string strdf:GeometryType(strdf:geometry A)
  xsd:int strdf:SRID(strdf:geometry A)
  ```
- Get the desired representation of a geometry
  ```
  xsd:string strdf:AsText(strdf:geometry A)
  strdf:wkb strdf:AsBinary(strdf:geometry A)
  xsd:string strdf:AsGML(strdf:geometry A)
  ```
- Test whether a certain condition holds
  ```
  xsd:boolean strdf:IsEmpty(strdf:geometry A)
  xsd:boolean strdf:IsSimple(strdf:geometry A)
  ```

stSPARQL: Geospatial SPARQL 1.1

Functions for testing topological spatial relationships

- **OGC Simple Features Access**

```
xsd:boolean strdf:Equals(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:Disjoint(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:Intersects(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:Touches(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:Crosses(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:Within(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:Contains(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:Overlaps(strdf:geometry A, strdf:geometry B)

xsd:boolean strdf:Relate(strdf:geometry A, strdf:geometry B,
                         xsd:string intersectionPatternMatrix)
```

- **Egenhofer**
- **RCC-8**

stSPARQL: Geospatial SPARQL 1.1

Spatial analysis functions

- Construct new geometric objects from existing geometric objects

```
strdf:geometry strdf:Boundary(strdf:geometry A)
strdf:geometry strdf:Envelope(strdf:geometry A)
strdf:geometry strdf:Intersection(strdf:geometry A, strdf:geometry B)
strdf:geometry strdf:Union(strdf:geometry A, strdf:geometry B)
strdf:geometry strdf:Difference(strdf:geometry A, strdf:geometry B)
strdf:geometry strdf:SymDifference(strdf:geometry A, strdf:geometry B)
strdf:geometry strdf:Buffer(strdf:geometry A, xsd:double distance)
```

- Spatial metric functions

```
xsd:float strdf:distance(strdf:geometry A, strdf:geometry B)
xsd:float strdf:area(strdf:geometry A)
```

- Spatial aggregate functions

```
strdf:geometry strdf:Union(set of strdf:geometry A)
strdf:geometry strdf:Intersection(set of strdf:geometry A)
strdf:geometry strdf:Extent(set of strdf:geometry A)
```

stSPARQL: Geospatial SPARQL 1.1

Select clause

- Construction of new geometries (e.g., **`strdf:buffer(?geo, 0.1)`**)
- Spatial aggregate functions (e.g., **`strdf:union(?geo)`**)
- Metric functions (e.g., **`strdf:area(?geo)`**)

Filter clause

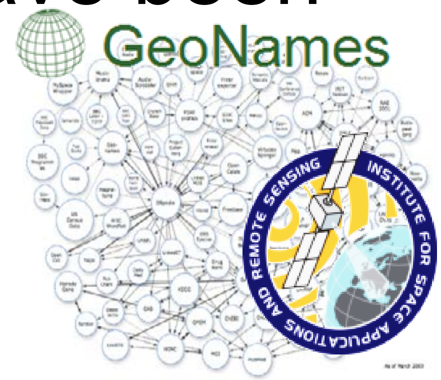
- Functions for testing topological spatial relationships between spatial terms (e.g., **`strdf:contains(?G1, strdf:union(?G2, ?G3))`**)
- Numeric expressions involving spatial metric functions (e.g., **`strdf:area(?G1) ≤ 2*strdf:area(?G2)+1`**)
- Boolean combinations

Having clause

- Boolean expressions involving spatial aggregate functions and spatial metric functions or functions testing for topological relationships between spatial terms (e.g., **`strdf:area(strdf:union(?geo))>1`**)

stSPARQL: An example (1/3)

Return the names of communities that have been affected by fires




```
SELECT   ?name
WHERE {
  ?community rdf:type dbpedia:Community;
             geonames:name ?name;
             strdf:hasGeometry ?comGeom.
  ?ba  rdf:type noa:BurntArea;
       strdf:hasGeometry ?baGeom.

  FILTER(strdf:overlap(?comGeom,?baGeom))
}
```

Spatial Function

stSPARQL: An example (2/3)

Find all burnt forests near communities

```
SELECT ?ba ?baGeom
WHERE {
   ?r rdf:type noa:Region;
      strdf:geometry ?rGeom;
      noa:hasCorineLandCoverUse ?f.
   ?f rdfs:subClassOf clc:Forests.
   ?c rdf:type dbpedia:Community;
         strdf:geometry ?cGeom.
   ?ba rdf:type noa:BurntArea;
       strdf:geometry ?baGeom.

   FILTER( strdf:intersects(?rGeom,?baGeom) &&
           strdf:distance(?baGeom,?cGeom) < 0.02)}
```

Spatial Functions

stSPARQL: An example 3/3)

Isolate the parts of the burnt areas that lie in coniferous forests.

Spatial Aggregate

```
SELECT ?burntArea
(strdf:intersection(?baGeom,
                    strdf:union(?fGeom))
 AS ?burntForest)
WHERE {
   ?burntArea   rdf:type noa:BurntArea;
                strdf:hasGeometry ?baGeom.
   ?forest rdf:type noa:Region;
           noa:hasLandCover noa:coniferousForest;
           strdf:hasGeometry ?fGeom.

   FILTER(strdf:intersects(?baGeom,?fGeom))
}
GROUP BY ?burntArea ?baGeom
```

Spatial Function

Conclusions

- **Geospatial data in the Semantic Web - stSPARQL**
  - Early works
  - The data model stRDF
  - Examples of publicly available linked geospatial data
  - The query language stSPARQL

- **Next topic:** Geospatial data in RDF - GeoSPARQL

Bibliography

[Kolas and Self, 2007]
Kolas, D., Self, T.: *Spatially Augmented Knowledgebase*. In: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007). Lecture Notes in Computer Science, vol. 4825, pp. 785-794. Springer Verlag (2007)

[Perry, 2008]
Perry, M.: *A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data*. Ph.D. thesis, Wright State University (2008)

[Koubarakis and Kyzirakos, 2010]
Koubarakis, M., Kyzirakos, K.: *Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL*. In: ESWC. pp. 425-439 (2010)

Geospatial data in RDF – GeoSPARQL

Presenter: Kostis Kyzirakos

Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens

GeoSPARQL

GeoSPARQL is a recently completed OGC standard [Perry and Herring, 2012]

Functionalities **similar to stSPARQL**:

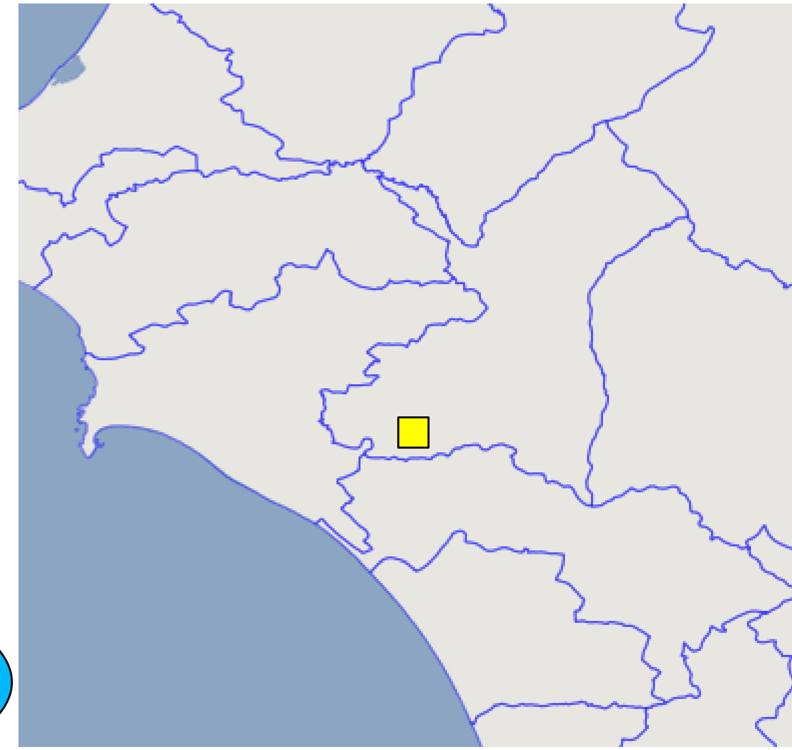
- Geometries are represented using **literals** similarly to stSPARQL.
- The same families of **functions** are offered for querying geometries.

Functionalities **beyond stSPARQL**:

- **Topological relations** can now be **asserted** as well so that reasoning and querying on them is possible.

Example in GeoSPARQL (1/2)

```
geonames:Olympia
  geonames:name "Ancient Olympia";
  rdf:type dbpedia:Community ;
  geo:hasGeometry ex:polygon1.
```



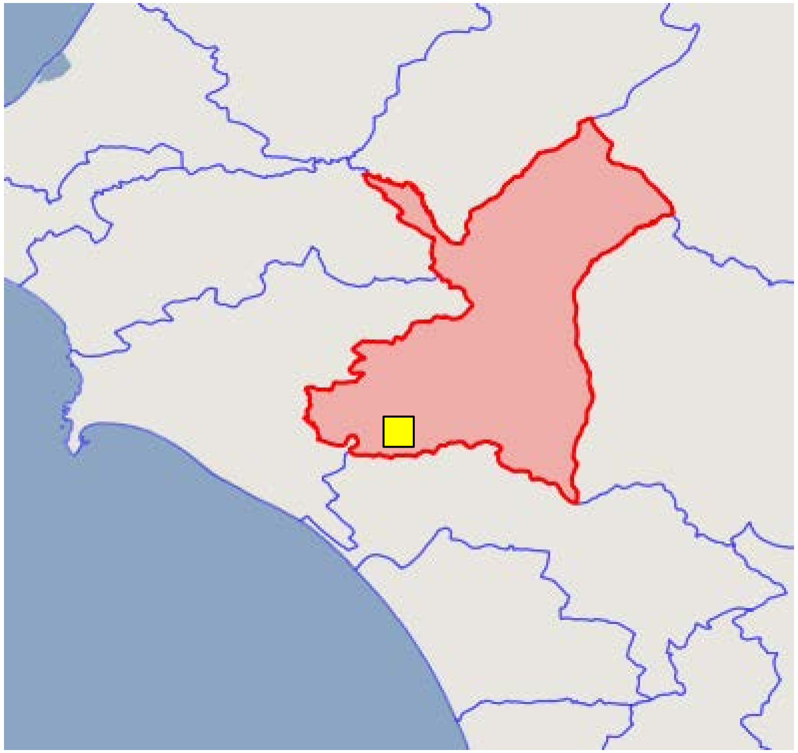
```
ex:polygon1
   rdf:type geo:Polygon;
   geo:asWKT "POLYGON((21.5 18.5,23.5 18.5,
                      23.5 21,21.5 21,21.5 18.5))
             "^^sf:wktLiteral.
```

Spatial literal

Spatial data type

Example in GeoSPARQL (2/2)

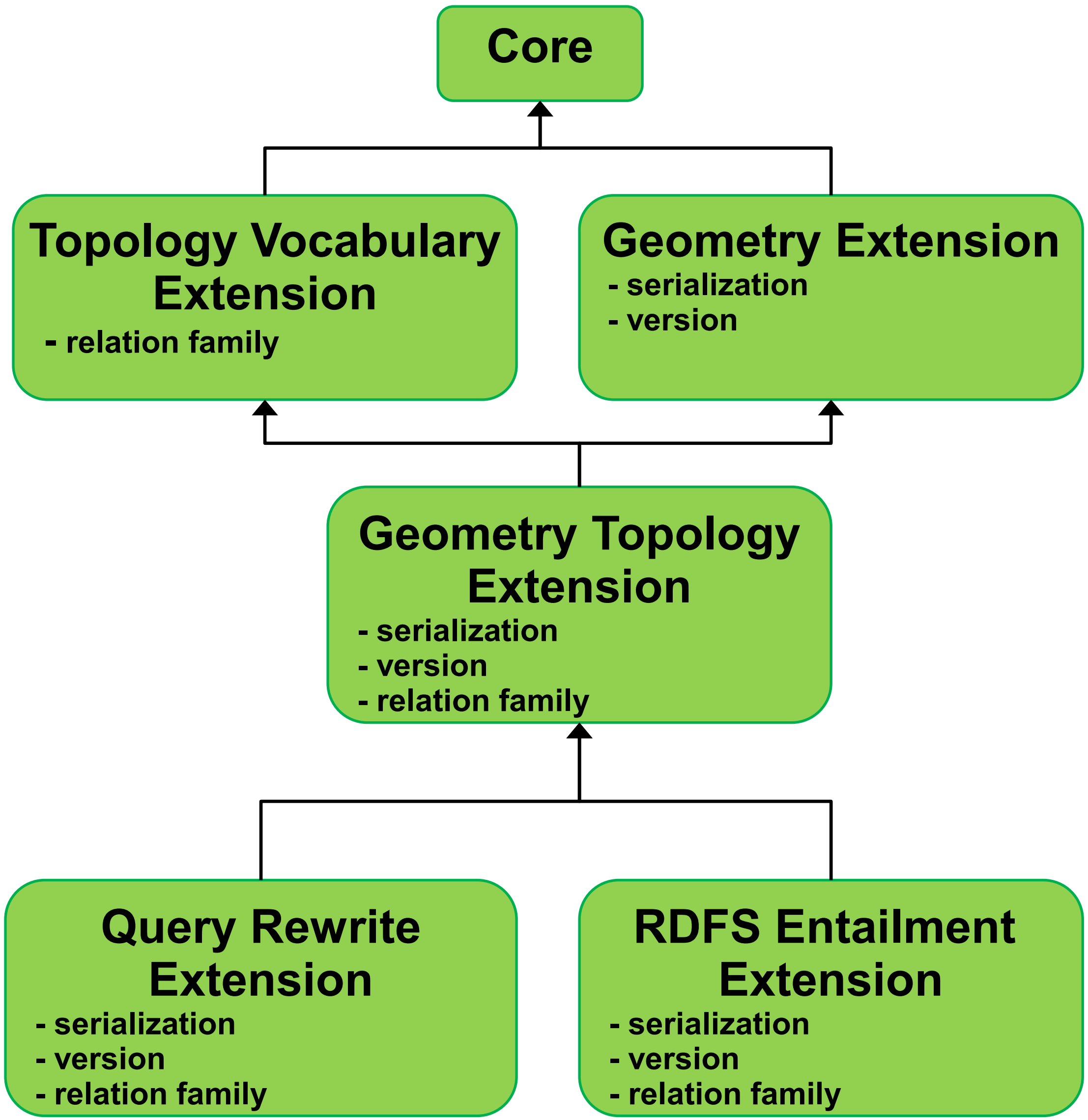
```
gag:OlympiaMunicipality
   rdf:type gag:Municipality;
   rdfs:label "ΔΗΜΟΣ ΑΡΧΑΙΑΣ
              ΟΛΥΜΠΙΑΣ"@el;
   rdfs:label "Municipality of
              Ancient Olympia".
```



```
gag:olympiaMunicipality geo:sfContains geonames:olympia .
```

Asserted topological relation

GeoSPARQL Components



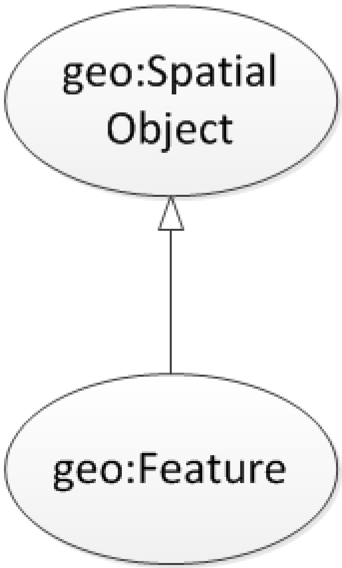
Parameters

- **Serialization**
  - WKT
  - GML
- **Relation Family**
  - Simple Features
  - RCC-8
  - Egenhofer

GeoSPARQL Core

Defines **top level classes** that provides users with vocabulary for modeling geospatial information.

- The class **`geo:SpatialObject`** is the top class and has as instances everything that can have a spatial representation.
- The class **`geo:Feature`** is a subclass of **`geo:SpatialObject`**. Feature is a domain entity that can have various **attributes** that describe **spatial and non-spatial** characteristics.



Example

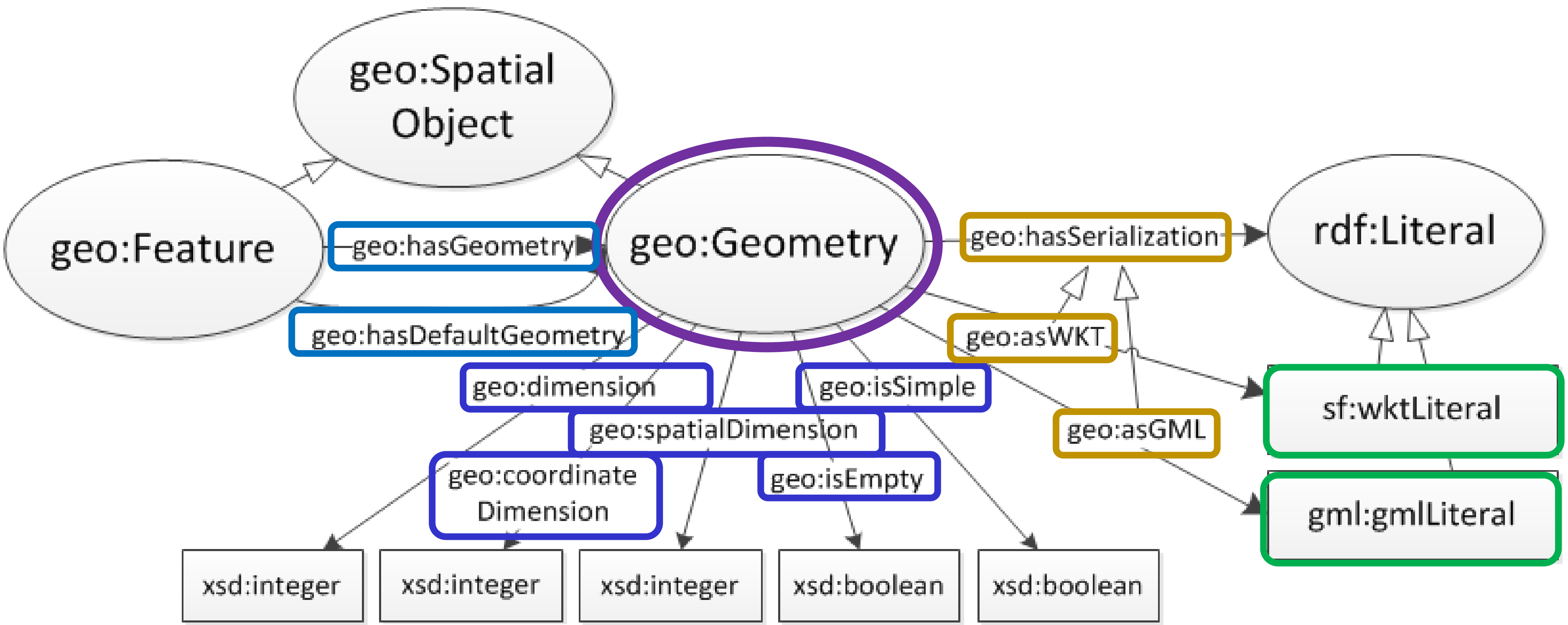
GeoSPARQL representation of the community of Ancient Olympia.

```
dbpedia:Community rdfs:subClassOf geo:Feature .
geonames:Olympia  geonames:name "Ancient Olympia";
                  rdf:type dbpedia:Community .
```

GeoSPARQL Geometry Extension

Provides vocabulary for asserting and querying information about geometries.

- The class **`geo:Geometry`** is a top class which is a superclass of all geometry classes.



Example

GeoSPARQL representation of the community of Ancient Olympia.

```
dbpedia:Community rdfs:subClassOf geo:Feature .
geonames:Olympia  geonames:name "Ancient Olympia";
                  rdf:type dbpedia:Community .


geonames:Olympia  geo:hasGeometry ex:polygon1.

ex:polygon1  rdf:type geo:Polygon;
             geo:isEmpty "false"^^xsd:boolean;
             geo:asWKT "POLYGON((21.5 18.5, 23.5
                        18.5, 23.5 21, 21.5 21,
                        21.5 18.5))"^^sf:wktLiteral.
```

Spatial data type

GeoSPARQL Geometry Extension

Spatial analysis functions

- **Construct new geometric objects from existing geometric objects**

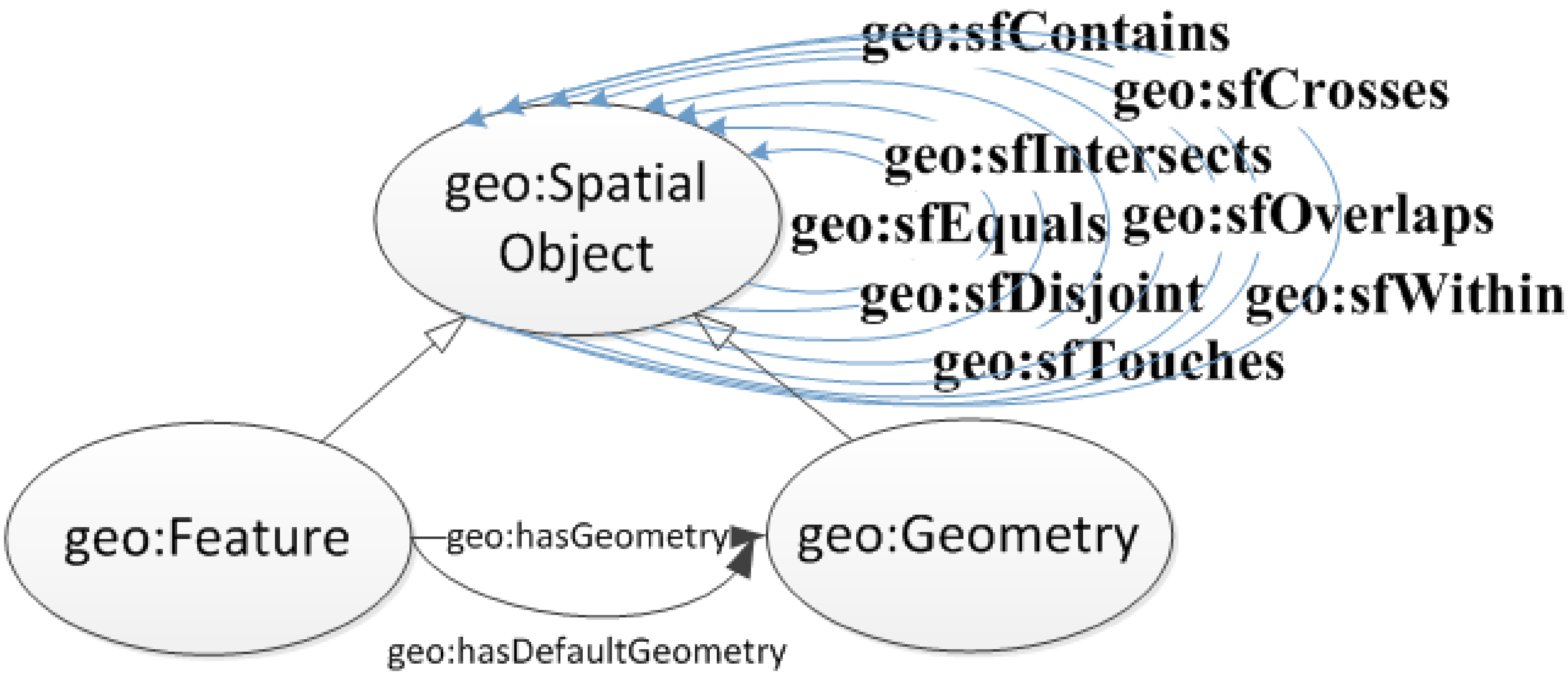
```
geof:boundary (geom1: ogc:geomLiteral): ogc:geomLiteral
geof:envelope (geom1: ogc:geomLiteral): ogc:geomLiteral
geof:intersection(  geom1: ogc:geomLiteral,
                    geom2: ogc:geomLiteral): ogc:geomLiteral
geof:union ( geom1: ogc:geomLiteral,
             geom2: ogc:geomLiteral): ogc:geomLiteral
geof:difference (   geom1: ogc:geomLiteral,
                    geom2: ogc:geomLiteral): ogc:geomLiteral
geof:symDifference (geom1: ogc:geomLiteral,
                    geom2:ogc:geomLiteral): ogc:geomLiteral
geof:buffer(geom: ogc:geomLiteral, radius: xsd:double,
            units: xsd:anyURI): ogc:geomLiteral
geof:convexHull(geom1: ogc:geomLiteral): ogc:geomLiteral
```

- **Spatial metric functions**

```
geof:distance(geom1: ogc:geomLiteral, geom2:
             ogc:geomLiteral, units: xsd:anyURI): xsd:double
```

GeoSPARQL Topology Vocabulary Extension

- The extension is parameterized by the family of topological relations supported.
  - Topological relations for simple features



  - The Egenhofer relations e.g., **`geo:ehMeet`**
  - The RCC-8 relations e.g., **`geo:rcc8ec`**

Example

```
gag:Olympia
  rdf:type gag:Community;
  geonames:name "Ancient Olympia".

gag:OlympiaBorough
  rdf:type gag:Borough;
  rdfs:label "Borough of
              Ancient Olympia".

gag:OlympiaMunicipality
  rdf:type gag:Municipality;
  rdfs:label "Municipality of
              Ancient Olympia".

gag:OlympiaBorough geo:sfContains geonames:Olympia .

gag:OlympiaMunicipality geo:sfContains
                              geonames:OlympiaBorough.
```

Asserted topological relation

GeoSPARQL: An example

Find the borough that contains the community of Ancient Olympia

```
SELECT  ?m
WHERE {
 ?m rdf:type gag:Borough.
 ?m geo:sfContains geonames:Olympia.
}
```

Topological Predicate

GeoSPARQL: An example

Find the municipality that contains the community of Ancient Olympia

```
SELECT   ?m
WHERE {
 ?m rdf:type gag:Municipality.
 ?m geo:sfContains geonames:Olympia.
}
```

What is the answer to this query?

Example (cont'd)

The answer to the previous query is

`?m = gag:OlympiaMunicipality`

GeoSPARQL does not tell you how to compute this answer which needs **reasoning about the transitivity** of relation `geo:sfContains`.

Options:

- Use rules
- Use constraint-based techniques

GeoSPARQL Geometry Topology Extension

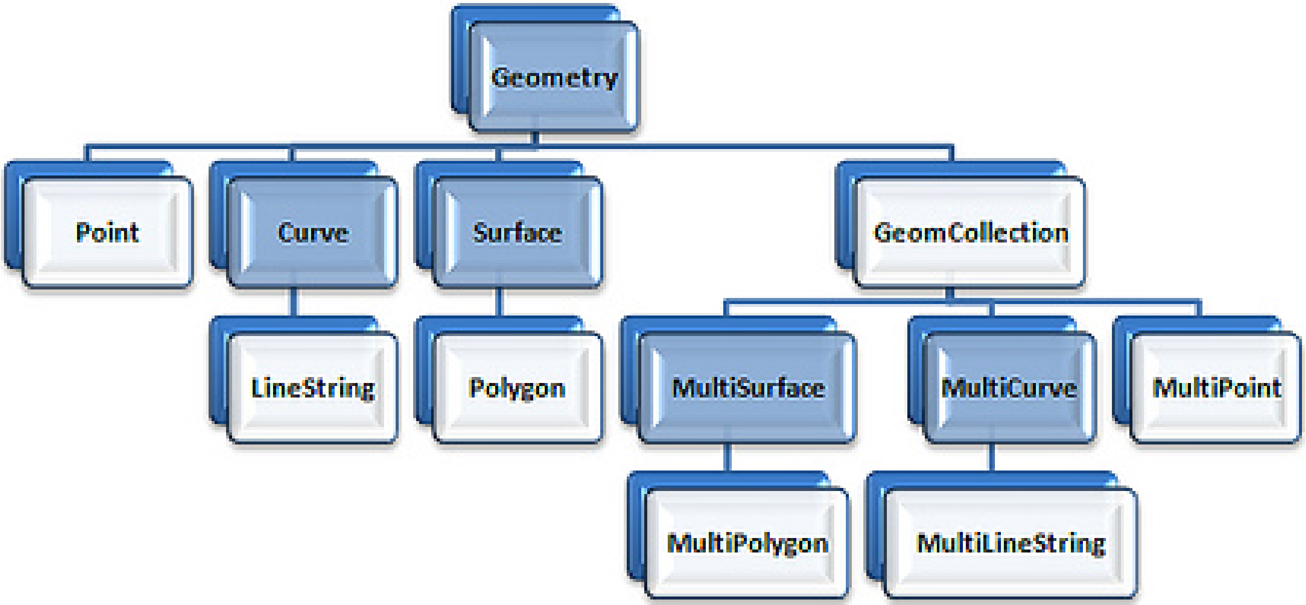
- Defines Boolean functions that correspond to each of the topological relations of the topology vocabulary extension:
  - OGC Simple Features Access

```
geof:sfEquals(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfDisjoint(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfIntersects(geom1: ogc:geomLiteral,geom2: ogc:geomLiteral): xsd:boolean
geof:sfTouches(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfCrosses(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfWithin(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfContains(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfOverlaps(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
```

  - Egenhofer
  - RCC-8

GeoSPARQL RDFS Entailment Extension

- Provides a mechanism for realizing the RDFS entailments that follow from the geometry class hierarchies defined by the WKT and GML standards.



- Systems should use an implementation of RDFS entailment to allow the derivation of new triples from those already in a graph.

Example

Given the triples

```
ex:f1 geo:hasGeometry ex:g1 .
geo:hasGeometry rdfs:domain geo:Feature.
```

we can infer the following triples:

```
ex:f1 rdf:type geo:Feature .
ex:f1 rdf:type geo:SpatialObject .
```

GeoSPARQL Query Rewrite Extension

- Provides a collection of **RIF rules** that use topological extension functions to establish the existence of topological predicates.

- Example: given the RIF rule named **`geor:sfWithin`**, the serializations of the geometries of **`dbpedia:Athens`** and **`dbpedia:Greece`** named **`AthensWKT`** and **`GreeceWKT`** and the fact that

`geof:sfWithin(AthensWKT, GreeceWKT)`

returns true from the computation of the two geometries, we can derive the triple

`dbpedia:Athens geo:sfWithin dbpedia:Greece`

- One possible implementation is to re-write a given SPARQL query.

RIF Rule

```
    Forall ?f1 ?f2 ?g1 ?g2 ?g1Serial ?g2Serial
      (?f1[geo:sfWithin->?f2] :-
        Or(
          And (?f1[geo:defaultGeometry->?g1]
               ?f2[geo:defaultGeometry->?g2]
               ?g1[ogc:asGeomLiteral->?g1Serial]
               ?g2[ogc:asGeomLiteral->?g2Serial]
               External(geo:sfWithin (?g1Serial,?g2Serial)))

          And (?f1[geo:defaultGeometry->?g1]
               ?g1[ogc:asGeomLiteral->?g1Serial]
               ?f2[ogc:asGeomLiteral->?g2Serial]
               External(geo:sfWithin (?g1Serial,?g2Serial)))

          And (?f2[geo:defaultGeometry->?g2]
               ?f1[ogc:asGeomLiteral->?g1Serial]
               ?g2[ogc:asGeomLiteral->?g2Serial]
               External(geo:sfWithin (?g1Serial,?g2Serial)))

          And (?f1[ogc:asGeomLiteral->?g1Serial]
               ?f2[ogc:asGeomLiteral->?g2Serial]
               External(geo:sfWithin (?g1Serial,?g2Serial)))
    ))
```

Feature - Feature

Feature - Geometry

Geometry - Feature

Geometry - Geometry

GeoSPARQL: An example

Discover the features that are inside the municipality of Ancient Olympia

```
SELECT ?feature
WHERE {
?feature geo:sfWithin
              geonames:OlympiaMunicipality.
}
```

GeoSPARQL: An example

```
SELECT ?feature
WHERE { {?feature geo:sfWithin geonames:Olympia }
        UNION
        { ?feature geo:defaultGeometry ?featureGeom .
          ?featureGeom geo:asWKT ?featureSerial .
          geonames:Olympia geo:defaultGeometry ?olGeom .
          ?olGeom geo:asWKT ?olSerial .
          FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
        UNION { ?feature geo:defaultGeometry ?featureGeom .
          ?featureGeom geo:asWKT ?featureSerial .
          geonames:Olympia geo:asWKT ?olSerial .
          FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
        UNION { ?feature geo:asWKT ?featureSerial .
          geonames:Olympia geo:defaultGeometry ?olGeom .
          ?olGeom geo:asWKT ?olSerial .
          FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
        UNION {
          ?feature geo:asWKT ?featureSerial .
          geonames:Olympia geo:asWKT ?olSerial .
          FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
```

Conclusions

- **Geospatial data in the Semantic Web**
  - The query language GeoSPARQL
    - Core
    - Topology vocabulary extension
    - Geometry extension
    - Geometry topology extension
    - Query rewrite extension
    - RDFS entailment extension

- **Next topic:** Implemented RDF Stores with Geospatial Support

Bibliography

[Perry and Herring, 2012]
Open Geospatial Consortium. *OGC GeoSPARQL - A geographic query language for RDF data*. OGC Candidate Implementation Standard (2012)

Implemented RDF Stores with Geospatial Support

Presenter: Kostis Kyzirakos

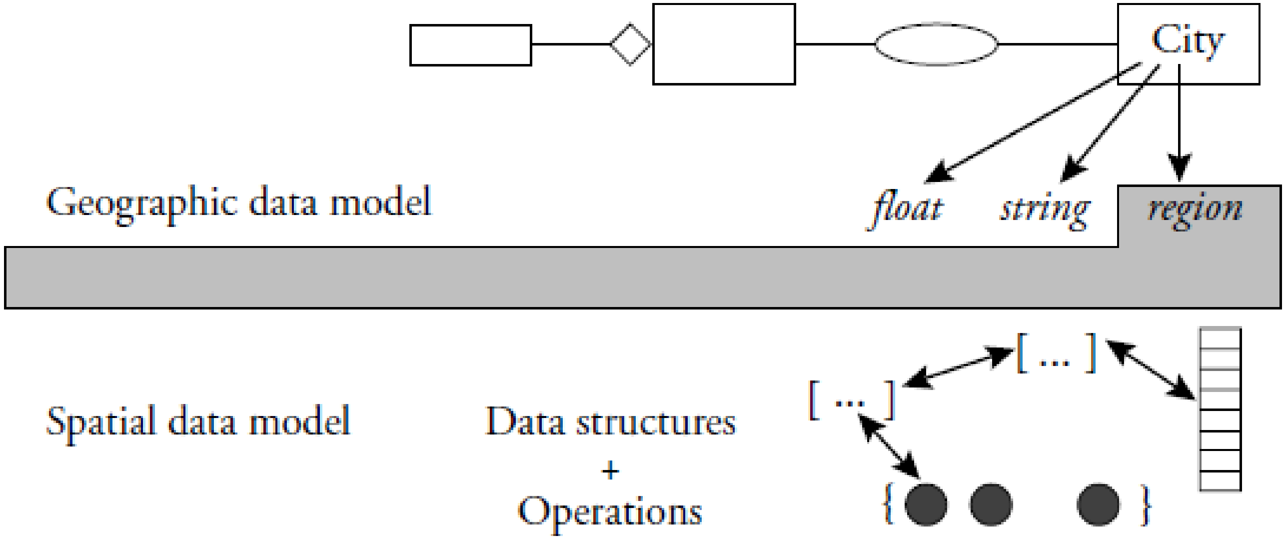
Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens

Outline

- Relational DBMS with a geospatial extension
- RDF stores with a geospatial component:
  - Research prototypes
  - Commercial systems

How does an RDBMS handle geometries? (1/2)

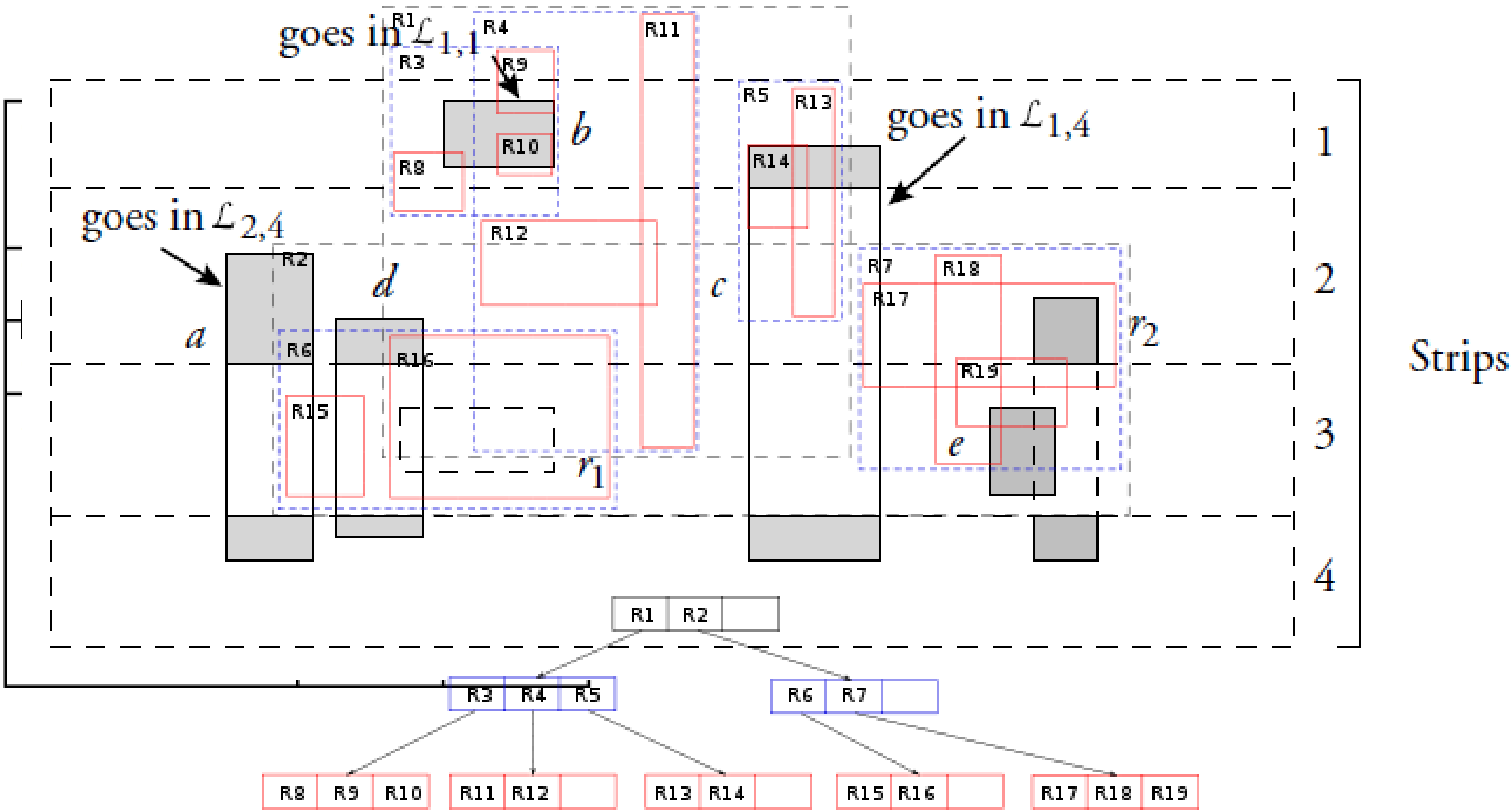
- Geometries are not explicitly handled by query language (SQL)
- Define datatypes that extend the SQL type system
  - Model geometries using Abstract Data Type (ADT)
  - Hide the structure of the data type to the user
    - The interface to an ADT is a list of operations
      - For spatial ADTs: Operations defined according to OGC Simple Features for SQL
    - Vendor-specific implementation irrelevant - extend SQL with geometric functionality independently of a specific representation/implementation



How does an RDBMS handle geometries? (2/2)

Special indices needed for geometry data types

Specialised query processing methods



Implemented Systems

Will examine following aspects:

- Data model
- Query language
- Functionality exposed
- Coordinate Reference System support
- Indexing Mechanisms

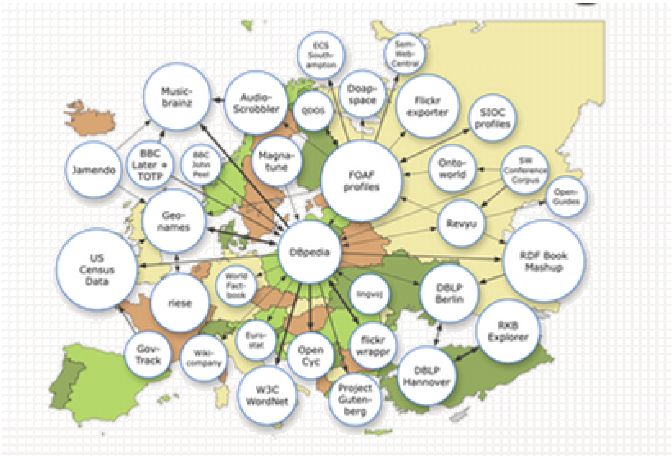
Research Prototypes

- Strabon
- Parliament
- Brodt et al.
- Perry

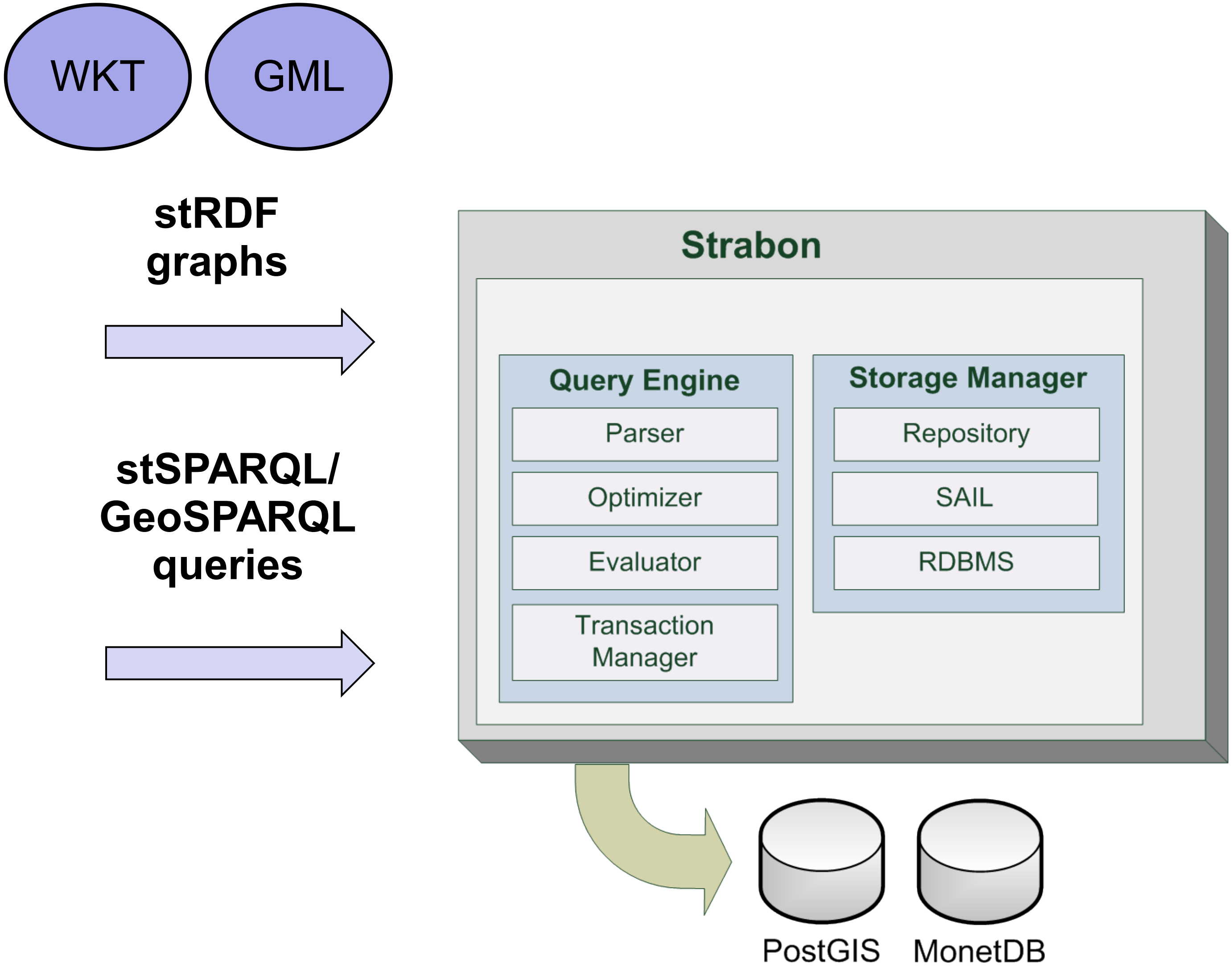
Strabon

[Kyzirakos et al.,'10,'12]

- Storage and query evaluation module for stSPARQL
- Geometries represented using typed literals
  - WKT & GML serializations supported
- Spatial predicates represented as SPARQL functions
  - OGC-SFA, Egenhofer, RCC-8 families exposed
  - Spatial aggregate functions
- Support for multiple coordinate reference systems
- GeoSPARQL support
  - Core
  - Geometry Extension
  - Geometry Topology Extension



Strabon - Implementation



Open Source, available from http://www.strabon.di.uoa.gr/

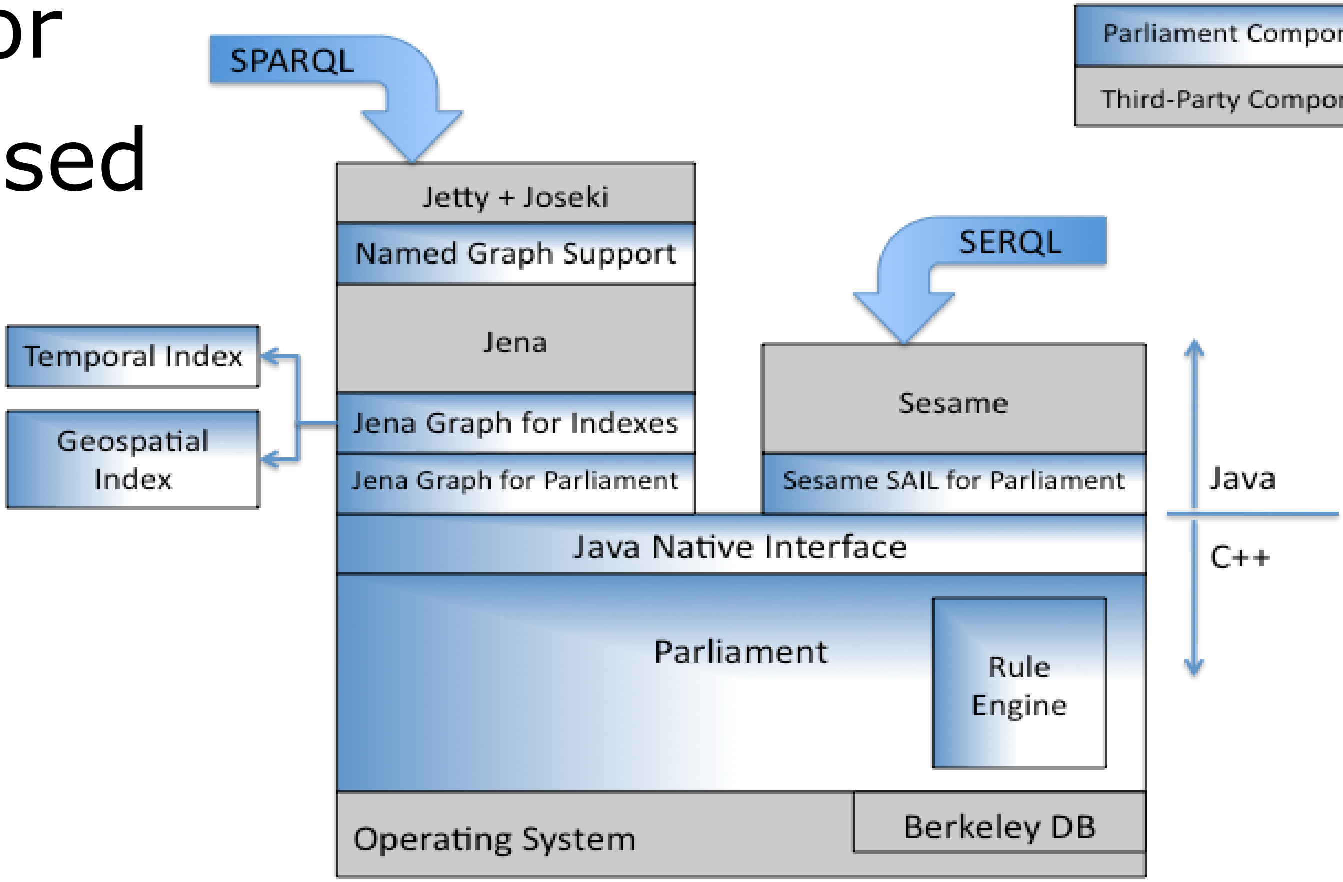
Parliament

- Storage Engine
- Developed by Raytheon BBN Technologies
- Implementation of GeoSPARQL
  - Geometries represented using typed literals
    - WKT & GML serializations supported
  - Three families of topological functions exposed
    - OGC-SFA
    - Egenhofer
    - RCC-8
  - Multiple CRS support

[Battle and Kolas, 2011]

Parliament - Implementation

- Rule engine included
- Paired with query processor
- R-tree used



Open Source, available from
http://www.parliament.semwebcentral.org

Brodt et al.

[Brodt et al., 2010]

- Built on top of RDF-3X
- Implemented at University of Stuttgart
- No formal definitions of data model and query language given
- Geometries expressed according to OGC-SFA
  - Typed Literals
  - WKT serialization supported
  - Expressed in WGS84
- Spatial predicates represented as SPARQL filter functions
  - OGC-SFA functionality exposed

Brodt et al. - Implementation

Focus on spatial query processing and spatial indexing techniques for spatial selections

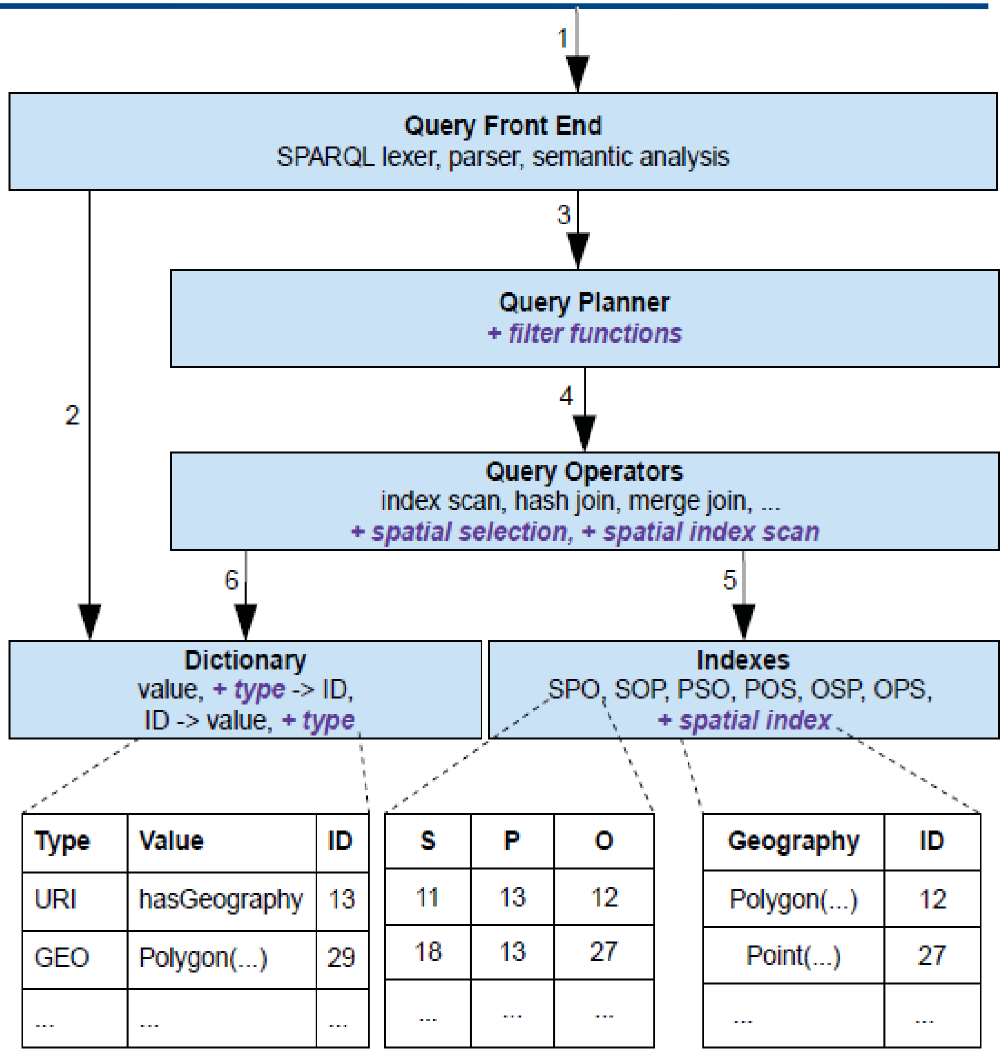
- e.g. "Retrieve features located inside a given polygon"

Naive spatial selection operator

- Placed in front of the execution plan which the planner returns

Spatial index (R-Tree) implemented

- Only utilized in spatial selections



Available upon request

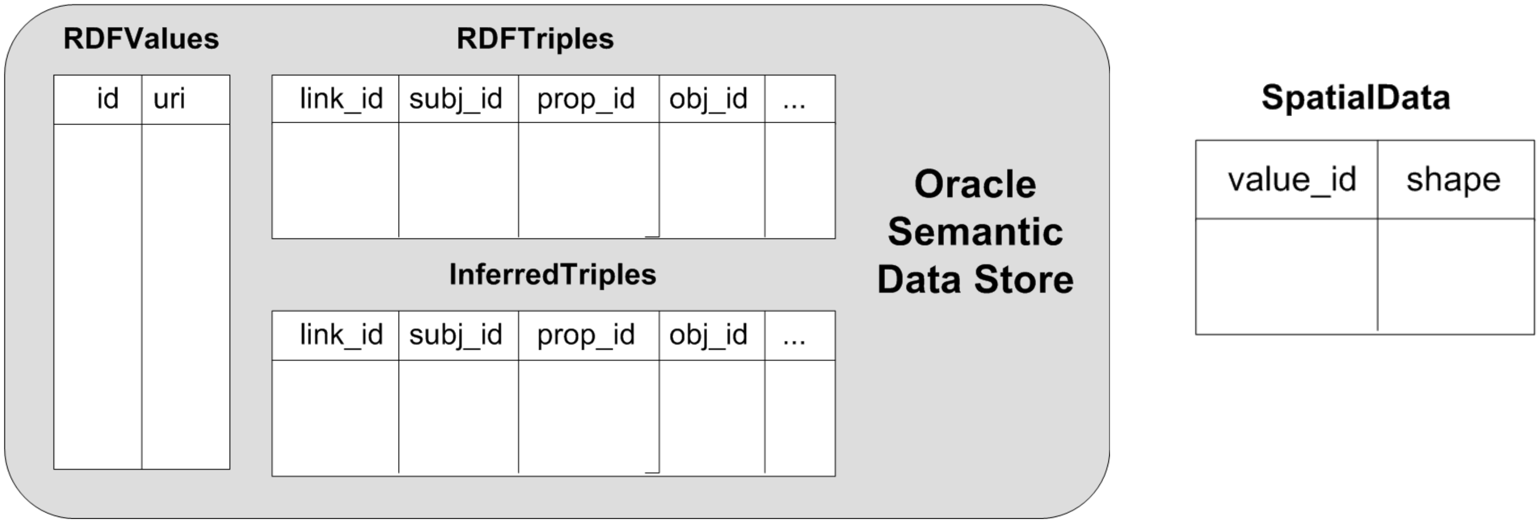
Perry

[Perry, 2008]

- Built on top of Oracle 10g
- Implemented at Wright State University
- Implementation of SPARQL-ST
  - Upper-level ontology imposed
- Geometries expressed according to GeoRSS GML
- Spatial and temporal variables introduced
- Spatial and temporal filters used to filter results with spatiotemporal constraints
  - RCC-8 calculus
  - Allen’s interval calculus

Perry - Implementation

- Spatiotemporal operators implemented using Oracle's extensibility framework
  - Three spatial operators defined
- Strictly RDF concepts implemented using Oracle's RDF storage and inferencing capabilities
- R-Tree used for indexing spatial objects



Available upon request

Commercial RDF Stores

- AllegroGraph
- OWLIM
- Virtuoso
- uSeekM

AllegroGraph

- Well-known RDF store, developed by Franz Inc.
- Two-dimensional point geometries
  - Cartesian / spherical coordinate systems supported
- GEO operator introduced for querying
  - Syntax similar to SPARQL's GRAPH operator
  - Available operations:
    - Radius / Haversine (Buffer)
    - Bounding Box
    - Distance
- Linear Representation of data
  - X and Y ordinates of a point are combined into a single datum
- Distribution sweeping technique used for indexing
  - Strip-based index
- Closed source, available from http://www.franz.com/agraph/allegrograph/

OWLIM

- Semantic Repository, developed by Ontotext
- Two-dimensional point geometries supported
  - Expressed using W3C Geo Vocabulary
    - Point Geometries
    - WGS84
- Spatial predicates represented as property functions
  - Available operations:
    - Point-in-polygon
    - Buffer
    - Distance
- Implemented as a Storage and Inference Layer for Sesame
- Custom spatial index used
- Closed Source
  - Free version available for evaluation purposes
  - http://www.ontotext.com/owlim

Virtuoso

- Multi-model data server, developed by OpenLink
- Two-dimensional point geometries
  - Typed literals
  - WKT serialization supported
  - Multiple CRS support
- Spatial predicates represented as functions
  - Subset of SQL/MM supported
- R-Tree used for indexing
- Spatial capabilities firstly included in Virtuoso 6.1
- Closed Source
  - Open Source Edition available from http://virtuoso.openlinksw.com/
    - Does not include the spatial capabilities extension

uSeekM

- Add-on library for Sesame-enabled semantic repositories, developed by OpenSahara
- Geometries expressed according to OGC-SFA
  - WKT serialization
  - Only WGS84 supported
- Spatial predicates represented as functions
  - OGC-SFA functionality exposed
  - Additional functions
    - e.g. `shortestline(geometry,geometry)`
- Implemented as a Storage and Inference Layer (SAIL) for Sesame
  - May be used with RDF stores that have a Sesame Repository/SAIL layer
- R-tree-over-GiST index used (provided by PostGIS)
- Open Source, Apache v2 License
  - Available from https://dev.opensahara.com/projects/useekm

System	Language	Index	Geometries	CRS support	Comments on Functionality
Strabon	stSPARQL/ GeoSPARQL*	R-tree-over-GiST	WKT / GML support	Yes	• OGC-SFA • Egenhofer • RCC-8
Parliament	GeoSPARQL	R-Tree	WKT / GML support	Yes	•OGC-SFA •Egenhofer •RCC-8
Brodt et al. (RDF-3X)	SPARQL	R-Tree	WKT support	No	OGC-SFA
Perry	SPARQL-ST	R-Tree	GeoRSS GML	Yes	RCC-8
AllegroGraph	Extended SPARQL	Distribution sweeping technique	2D point geometries	Partial	•Buffer •Bounding Box •Distance
OWLIM	Extended SPARQL	Custom	2D point geometries (W3C Basic Geo Vocabulary)	No	•Point-in-polygon •Buffer •Distance
Virtuoso	SPARQL	R-Tree	2D point geometries (in WKT)	Yes	SQL/MM (subset)
uSeekM	SPARQL	R-tree-over GiST	WKT support	No	OGC-SFA

Conclusions

- **Semantic Geospatial Systems**:
  - Research Prototypes
  - Commercial Systems

- **Next topic:** Geospatial information with description logics, OWL and rules

Bibliography

[Kyzirakos et al, 2010]

K. Kyzirakos , M. Karpathiotakis, M. Koubarakis: Developing Registries for the Semantic Sensor Web using stRDF and stSPARQL (short paper). In: Proceedings of the 3rd International Workshop on Semantic Sensor Networks (SSN10) (2010)

[Kyzirakos et al, 2012]

K. Kyzirakos , M. Karpathiotakis, M. Koubarakis: *Strabon: A Semantic Geospatial DBMS*. In: Proceedings of the 11th International Semantic Web Conference (2012)

[Battle and Kolas, 2011]

Battle, R., Kolas, D.: *Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL* (2011)

Bibliography

[Brodt et al, 2010]

A. Brodt, D. Nicklas, and B. Mitschang. *Deep integration of spatial query processing into native rdf triple stores*. In ACM SIGSPATIAL, 2010.

[Perry, 2007]

Matthew Perry. *A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data*. PhD thesis, Wright State University, 2008