Data models, Query Languages, Implemented Systems and Applications of Linked Geospatial Data

Manolis Koubarakis, Kostis Kyzirakos and Manos Karpathiotakis

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

Tutorial Organization

- **14:00 – 14:15** Introduction
- **14:15 – 15:00** Background in geospatial data modeling
- **15:00 – 15:30** Geospatial data in the Semantic Web – stSPARQL
- **15:30 – 16:00** Coffee break
- **16:00- 16:30** Geospatial data in the Semantic Web – GeoSPARQL
- **16:30 – 17:00** Implemented systems and applications
- **17:00 – 17:15** Conclusions, questions, discussion
- **17:15 – 17:30** Demo of Strabon

Introduction

Presenter: Manolis Koubarakis

Outline

- Why geospatial information?
- Geographical Information Science and Systems
- Why this tutorial?

Why Geospatial Information?

- **Geospatial,** and in general **geographical,** information is very important in reality: everything that happens, happens somewhere (**location**).

- **Decision making can be substantially improved** if we know where things take place.

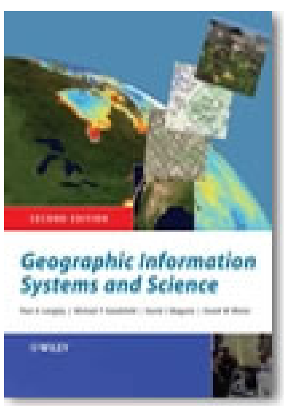
Geography

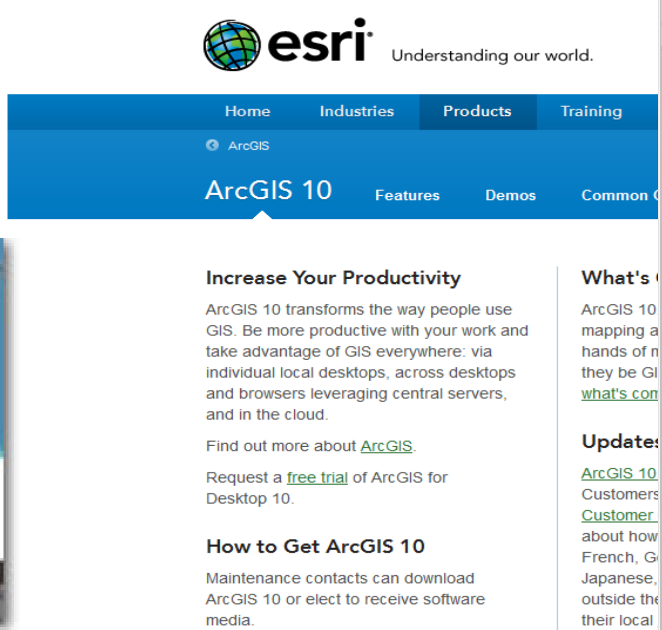
- From http://en.wikipedia.org/wiki/Geography
  - **Geography** is the science that studies the lands, the features, the inhabitants and the phenomena of the Earth.
  - From the Greek word **γεωγραφία (geographia)** which means “describing the Earth”.

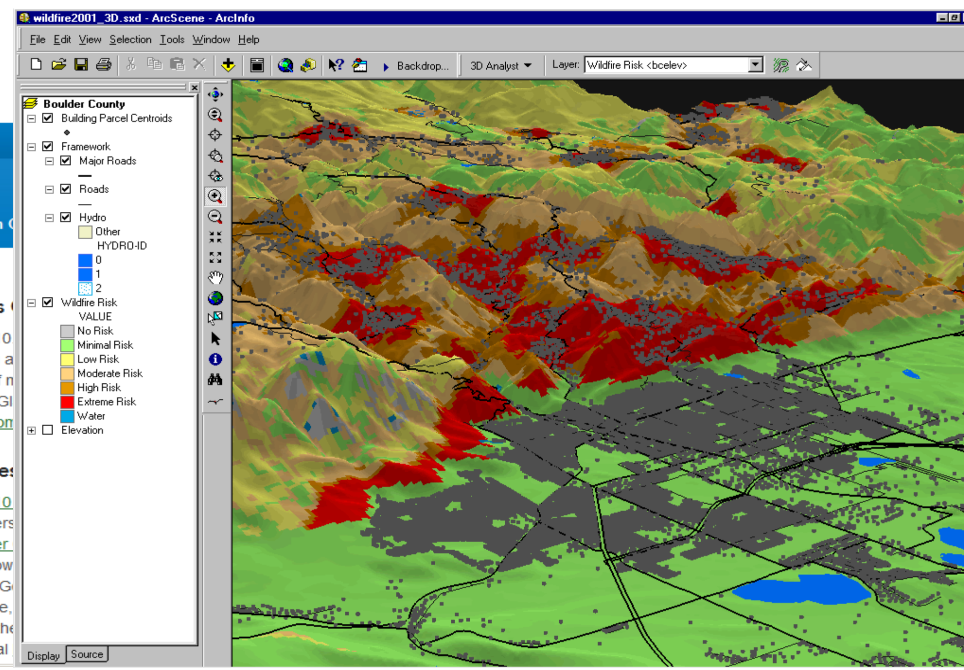
Geographical Information Systems and Science

- A **geographical information system (GIS)** is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data.

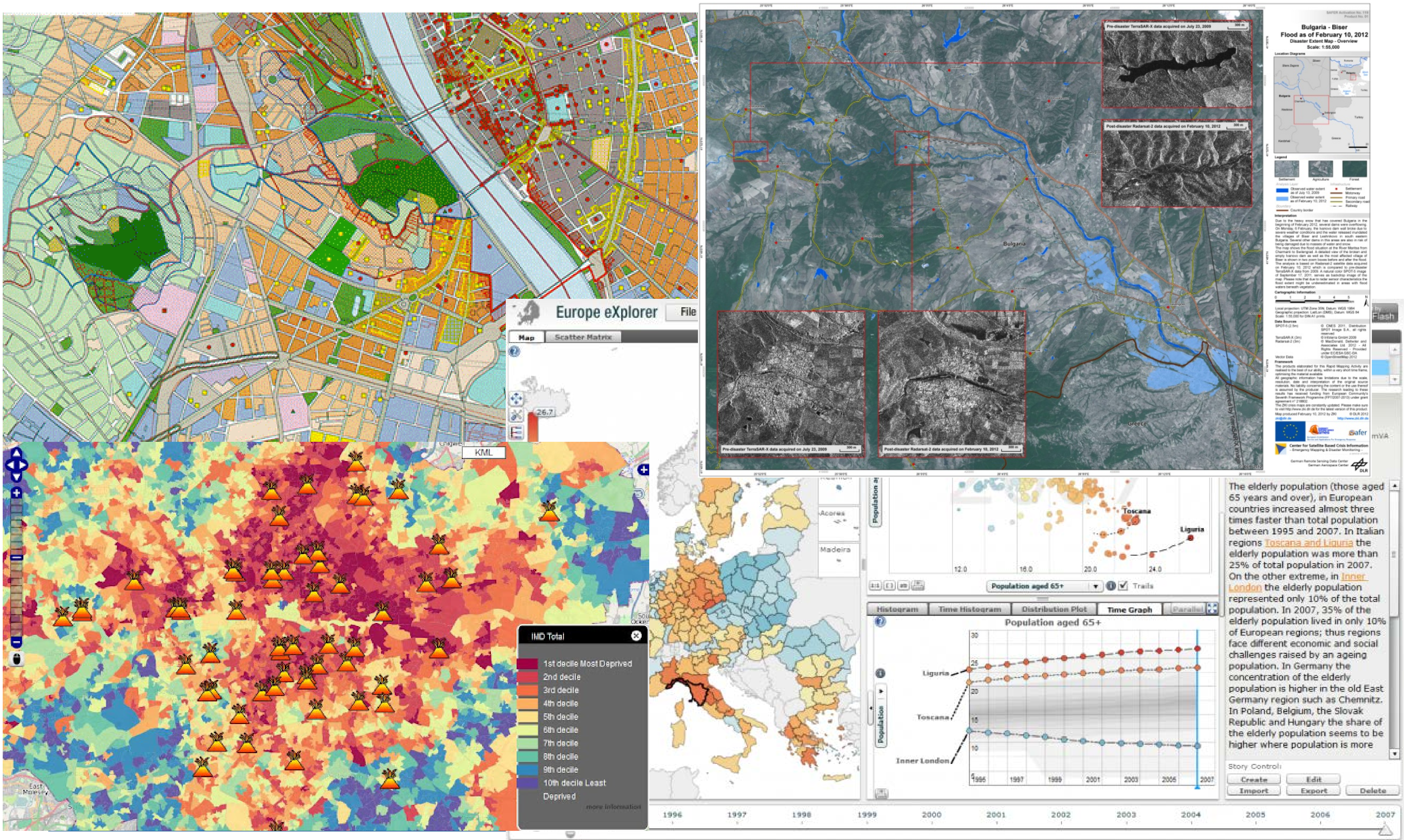
- **GIS science** is the field of study for developing and using GIS.







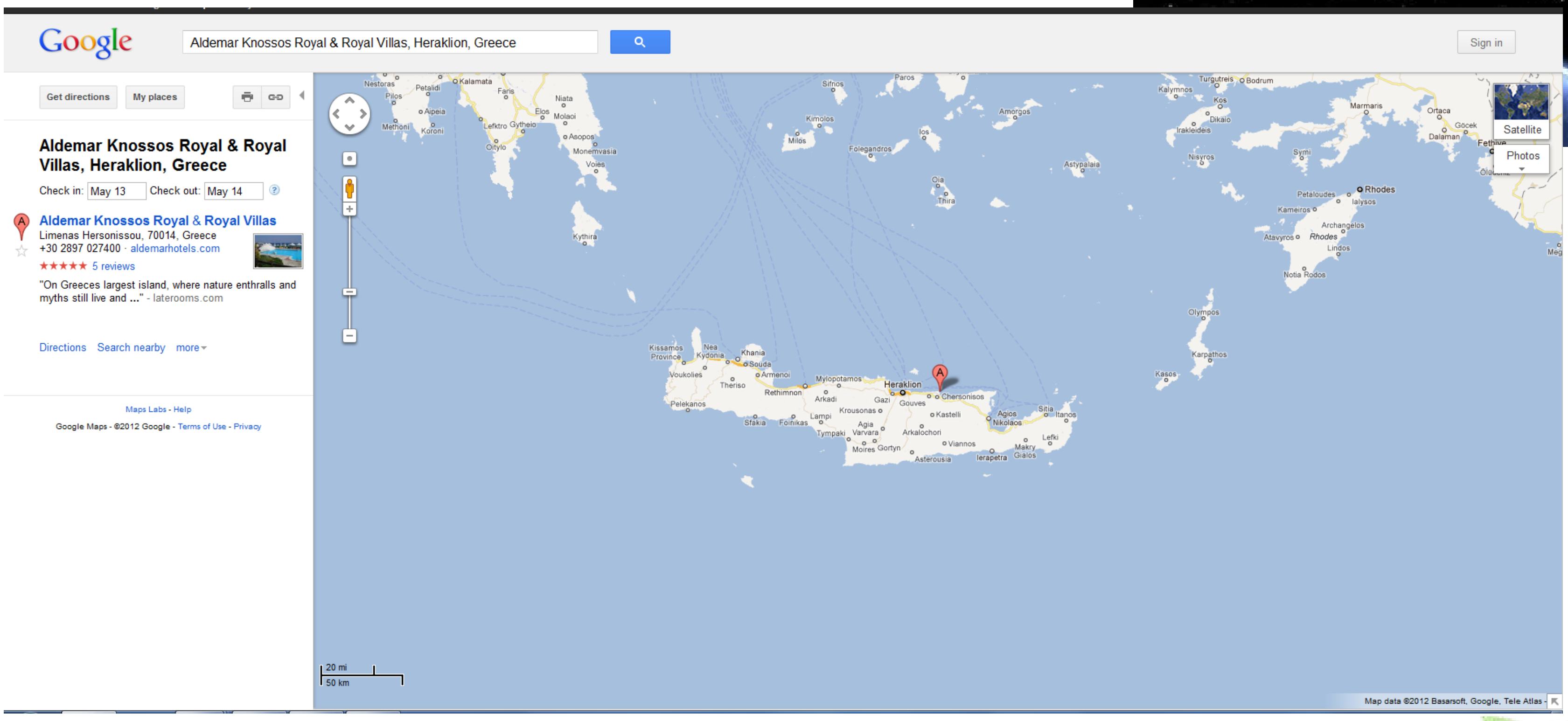
Combining GIS Data for Decision Making



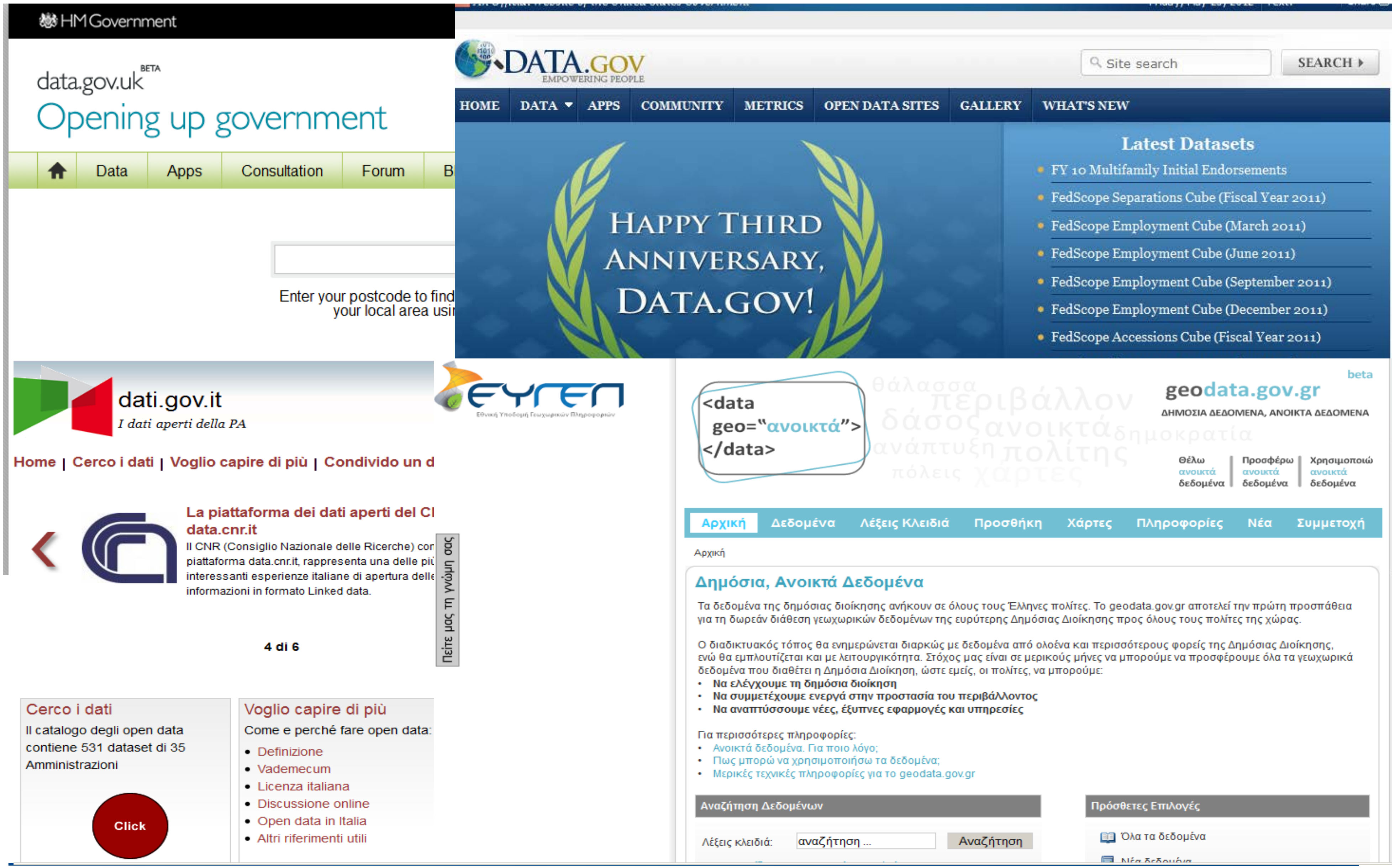
Why this tutorial?

- Lots of **geospatial data is available on the Web** today.
- Lots of **public data** coming out of open government initiatives is **geospatial**.
- Lots of this public data is quickly being **transformed into linked data!**
- People have started building **applications** utilizing linked data.

Geospatial data on the Web

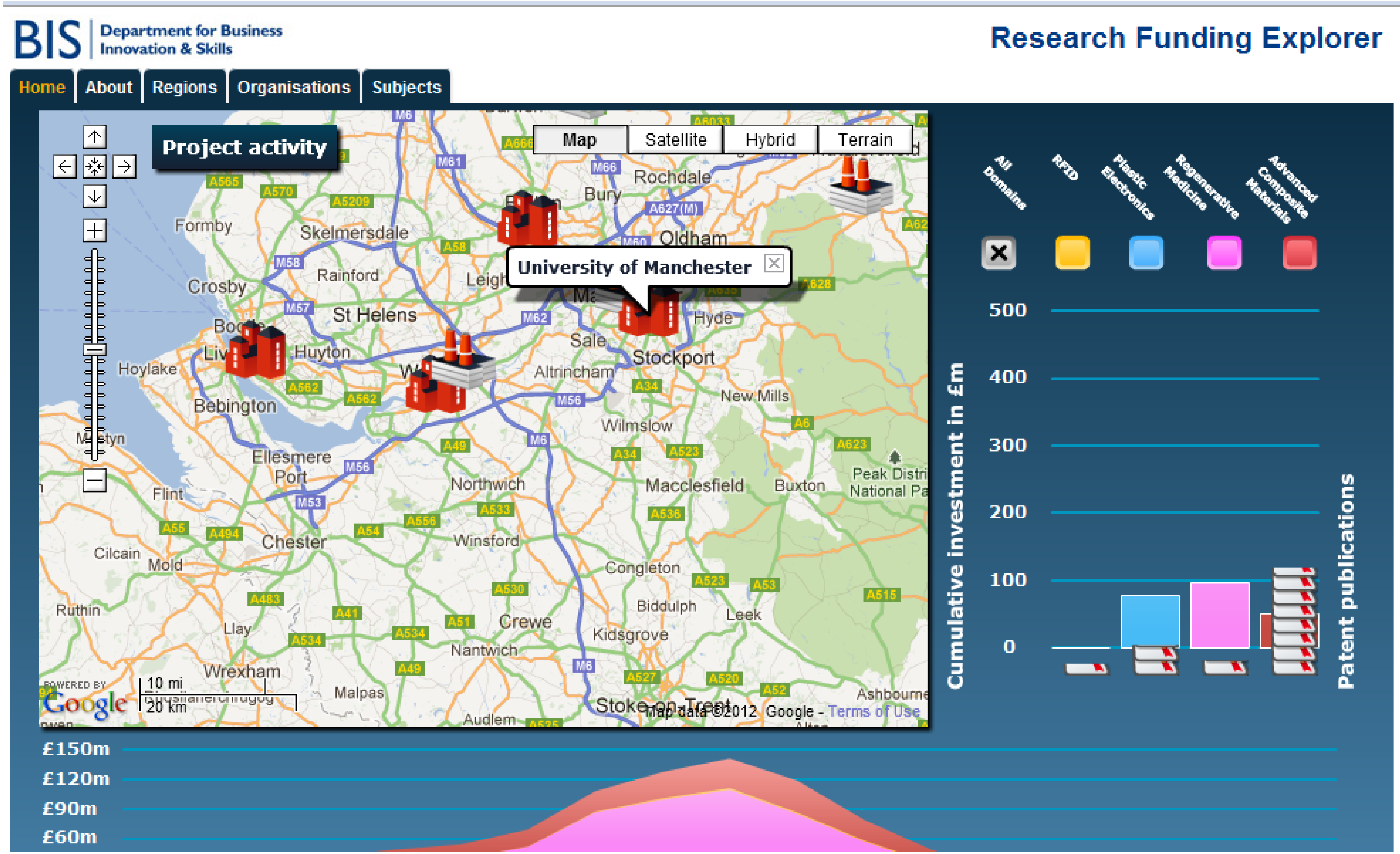


Open Government Data



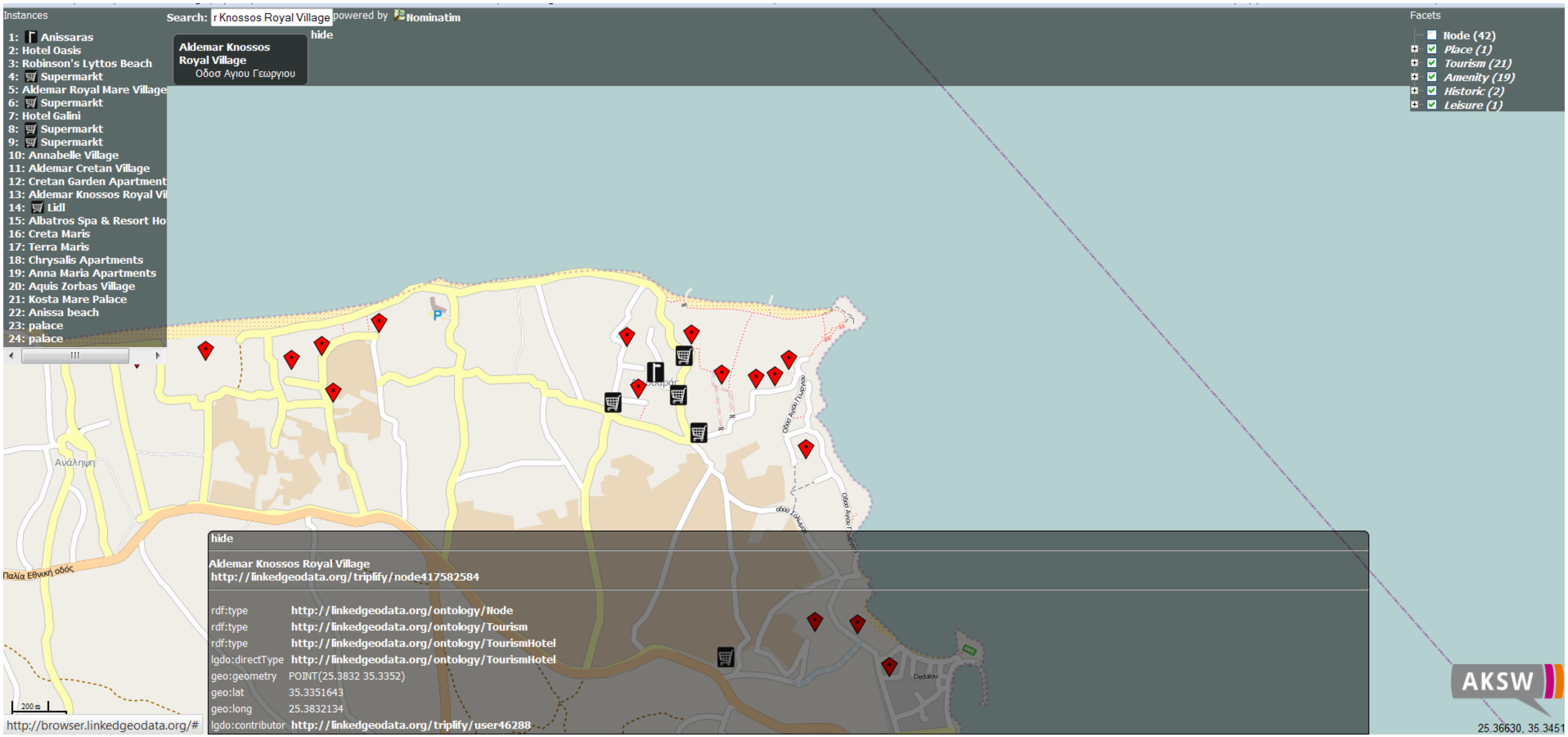
Linked geospatial data – Ordnance Survey

Linked geospatial data – Research Funding Explorer



Linked geospatial data – Spain

Linked geospatial data – Open Street Map



Conclusions

- **Introduction**
  - Why geospatial information?
  - Geographical Information Science and Systems
  - Why this tutorial?

- **Next topic:** Background in geospatial data modeling

Background in geospatial data modeling

Presenter: Manolis Koubarakis

Outline

- Basic GIS concepts and terminology
- Abstract geographic space modeling paradigms
- Concrete representations for the abstract modeling paradigms
- Geospatial data standards

Basic GIS Concepts and Terminology

- **Theme:** the information corresponding to a particular domain that we want to model. A theme is a set of **geographic features.**

- **Example:** the countries of Europe

Basic GIS Concepts (cont'd)

- **Geographic feature or geographic object:** a domain entity that can have various **attributes** that describe **spatial and non-spatial** characteristics.

- **Example:** the country Greece with attributes
  - Population
  - Flag
  - Capital
  - Geographical area
  - Coastline
  - Bordering countries

Basic GIS Concepts (cont'd)

- Geographic features can be **atomic** or **complex.**

- **Example:** According to the Kallikratis administrative reform of 2010, Greece consists of:
  - 13 **regions** (e.g., Crete)
  - Each region consists of **perfectures** (e.g., Heraklion)
  - Each perfecture consists of **municipalities** (e.g., Dimos Chersonisou)

Basic GIS Concepts (cont'd)

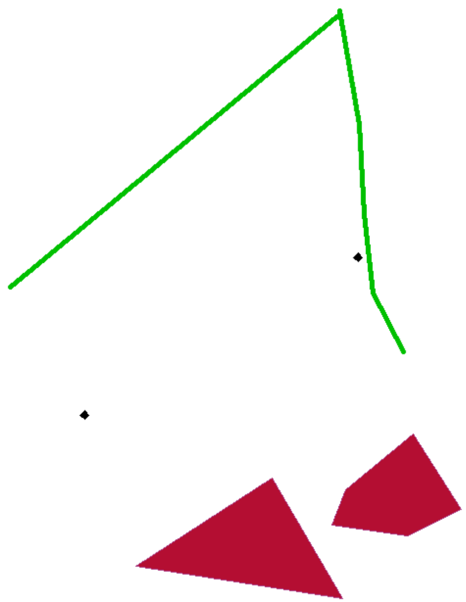
- The spatial characteristics of a feature can involve:
  - **Geometric information** (location in the underlying geographic space, shape etc.)
  - **Topological information** (containment, adjacency etc.).

Municipalities of the perfecture of Heraklion:

1. Dimos Irakliou
2. Dimos Archanon-Asterousion
3. Dimos Viannou
4. Dimos Gortynas
5. Dimos Maleviziou
6. Dimos Minoa Pediadas
7. Dimos Festou
8. Dimos Chersonisou

Geometric Information

- Geometric information can be captured by using geometric primitives (**points**, **lines**, **polygons,** etc.) to approximate the spatial attributes of the real world feature that we want to model.



- Geometries are associated with a **coordinate reference system** (or **spatial reference system**) which describes the coordinate space in which the geometry is defined.

Topological Information

- Topological information is **inherently qualitative** and it is expressed in terms of **topological relations** (e.g., containment, adjacency, overlap etc.).

- Topological information can be **derived from geometric information** or it might be captured by **asserting explicitly the topological relations** between features.

Topological Relations

- The study of topological relations has produced a lot of interesting results by researchers in:
  - GIS
  - Spatial databases
  - Artificial Intelligence (qualitative reasoning and knowledge representation)

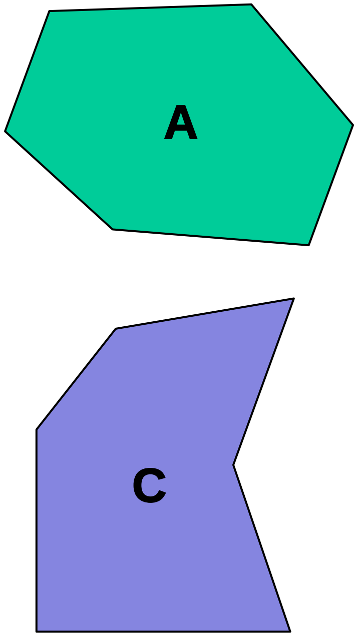
DE-9IM

- The **dimensionally extended 9-intersection model** has been defined by Clementini, Di Felice and van Oosterom in 1993 based on previous work by these authors, Egenhofer, Franzosa and others.

- It captures topological relationships between geometries in $\mathbb{R}^2$ by considering the **dimension of the intersections involving the interior, boundary and exterior of the two geometries:**

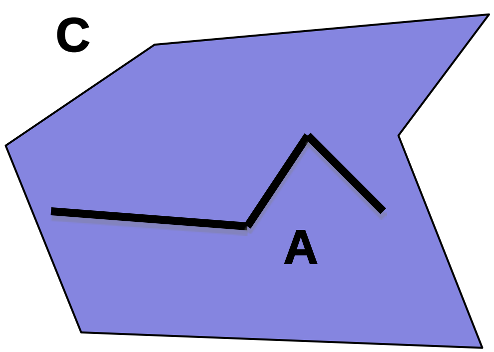
$$\text{DE-9IM}(a,b) = \begin{bmatrix} \dim(I(a)\cap I(b)) & \dim(I(a)\cap B(b)) & \dim(I(a)\cap E(b)) \\ \dim(B(a)\cap I(b)) & \dim(B(a)\cap B(b)) & \dim(B(a)\cap E(b)) \\ \dim(E(a)\cap I(b)) & \dim(E(a)\cap B(b)) & \dim(E(a)\cap E(b)) \end{bmatrix}.$$

Example: A DISJOINT C



	I(C)	**B**(C)	**E**(C)
I(A)	F	F	*
B(A)	F	F	*
E(A)	*	*	*

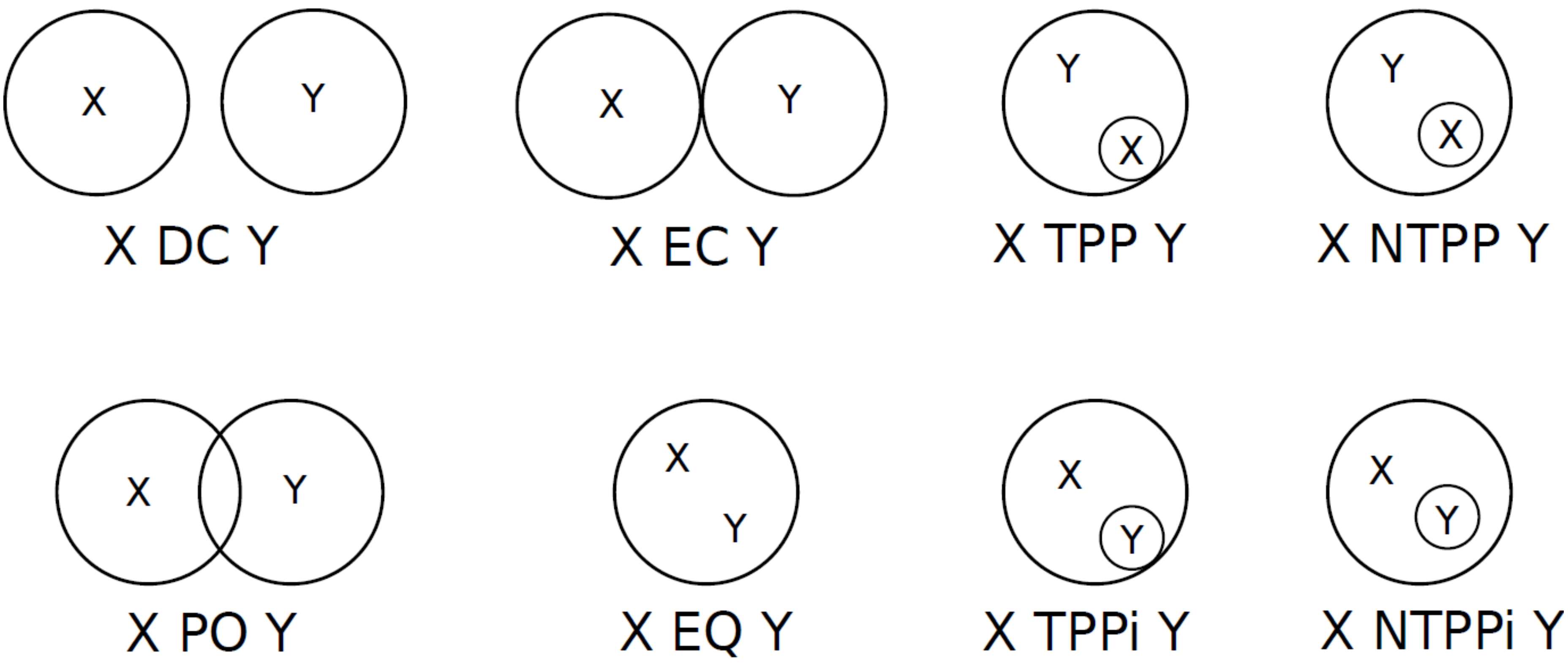
Example: A WITHIN C



	I(C)	**B**(C)	**E**(C)
I(A)	T	*	F
B(A)	*	*	F
E(A)	*	*	*

The RCC-8 Calculus (Randell et al., 1991)

- **Eight JEPD binary relations** that can be all defined in terms of a single primitive (**connection** of two regions).



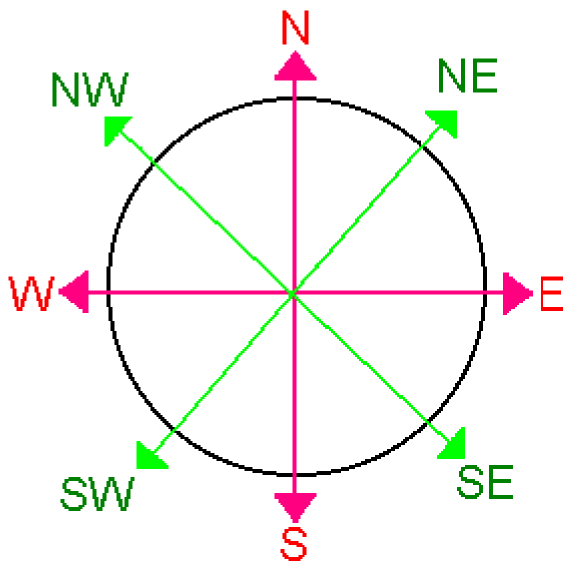
RCC-8 (cont'd)

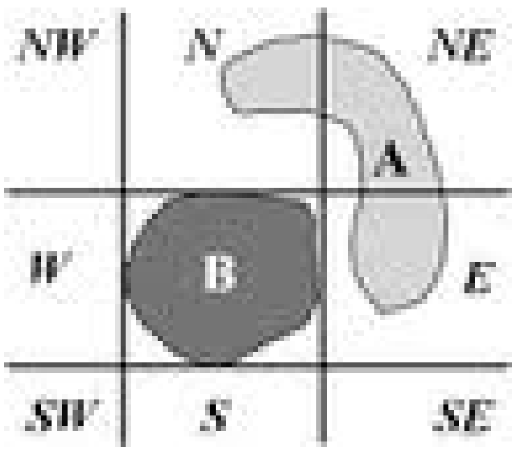
- Variations exist depending on:
  - **What kind of regions of a topological space** are considered (non-empty, regular, closed, connected, holes allowed, dimensionality).
  - How is **contact** of two regions defined (the regions have a point in common or their closures have a point in common).

- The **RCC-5 subset** has also been studied (no distinction among TPP and NTPP, called just PP).

More Qualitative Spatial Relations

- **Orientation/Cardinal directions** (left of, right of, north of, south of, northeast of etc.)



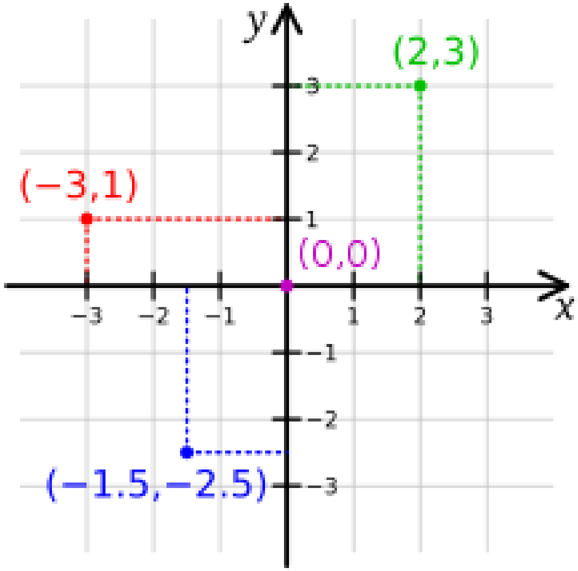


- **Distance** (close to, far from etc.). This information can also be **quantitative**.

Coordinate Systems

- **Coordinate:** one of *n* scalar values that determines the position of a point in an *n*-dimensional space.
- **Coordinate system:** a set of mathematical rules for specifying how coordinates are to be assigned to points.

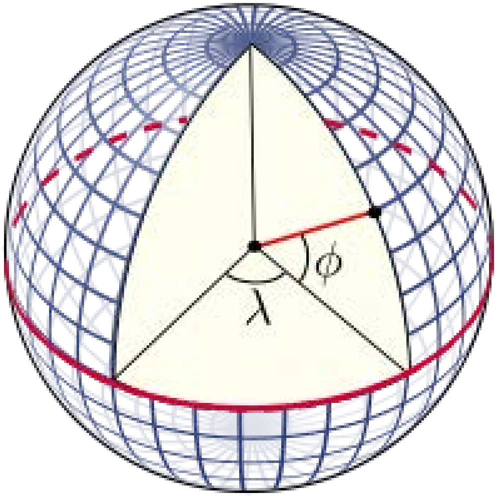
- **Example**: the Cartesian coordinate system



Coordinate Reference Systems

- **Coordinate reference system**: a coordinate system that is related to an **object** (e.g., the Earth, a planar projection of the Earth, a three dimensional mathematical space such as R^3) through a **datum** which species its origin, scale, and orientation.

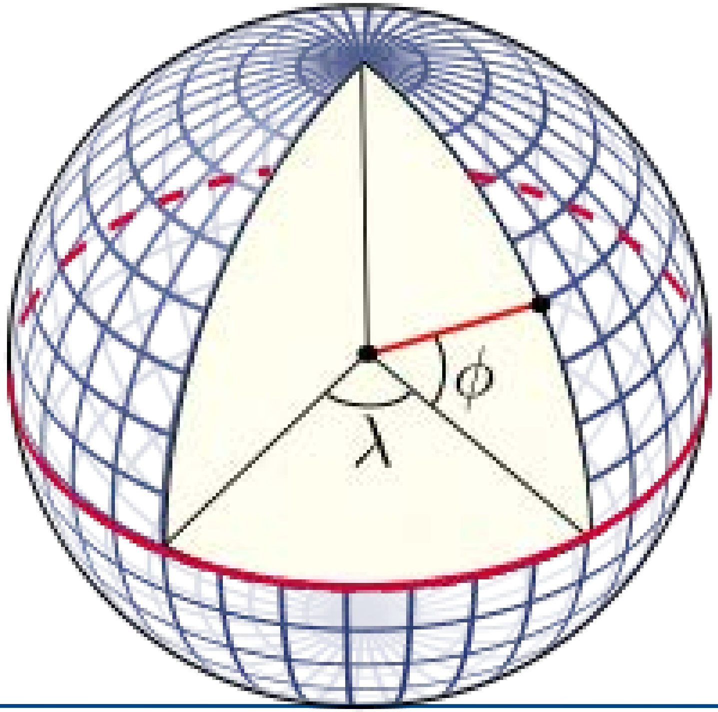
- **Geographic coordinate reference system**: a 3-dimensional coordinate system that utilizes **latitude (φ)**, **longitude (λ)** , and optionally **geodetic height (i.e., elevation)**, to capture geographic locations on Earth.



The World Geodetic System

- The **World Geodetic System (WGS)** is the most well-known geographic coordinate reference system and its latest revision is **WGS84.**

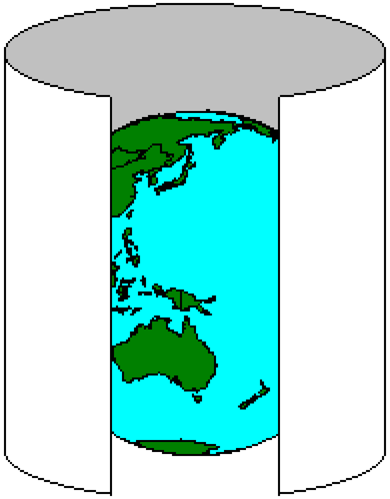
- **Applications:** cartography, geodesy, navigation (GPS), etc.



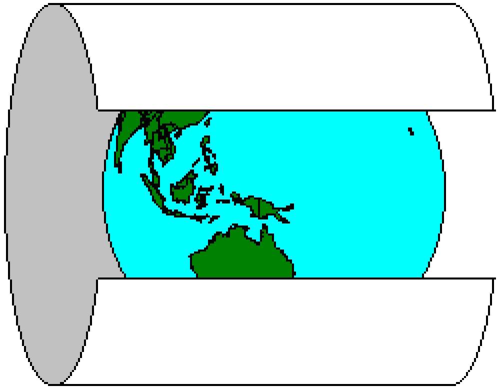
Projected Coordinate Reference Systems

- **Projected coordinate reference system:** they transform the 3-dimensional ellipsoid approximation of the Earth into a 2-dimensional surface (distortions!)

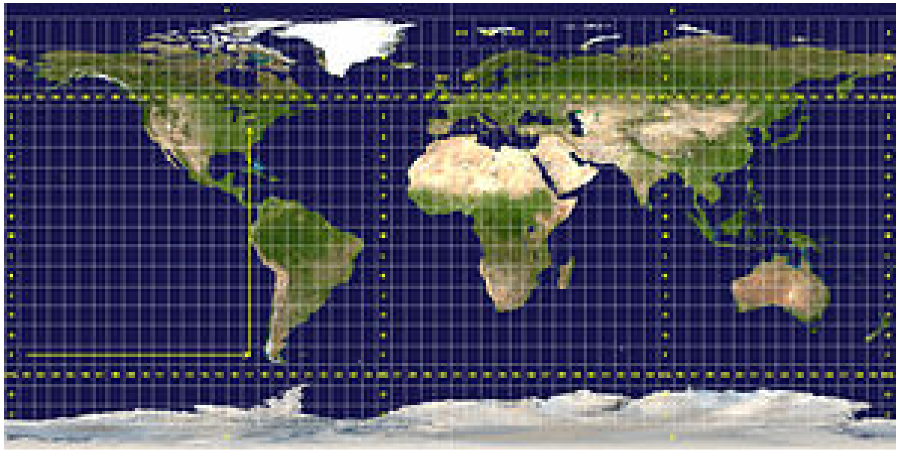
- **Example:** the **Universal Transverse Mercator (UTM)** system



Mercator projection



Transverse Mercator projection



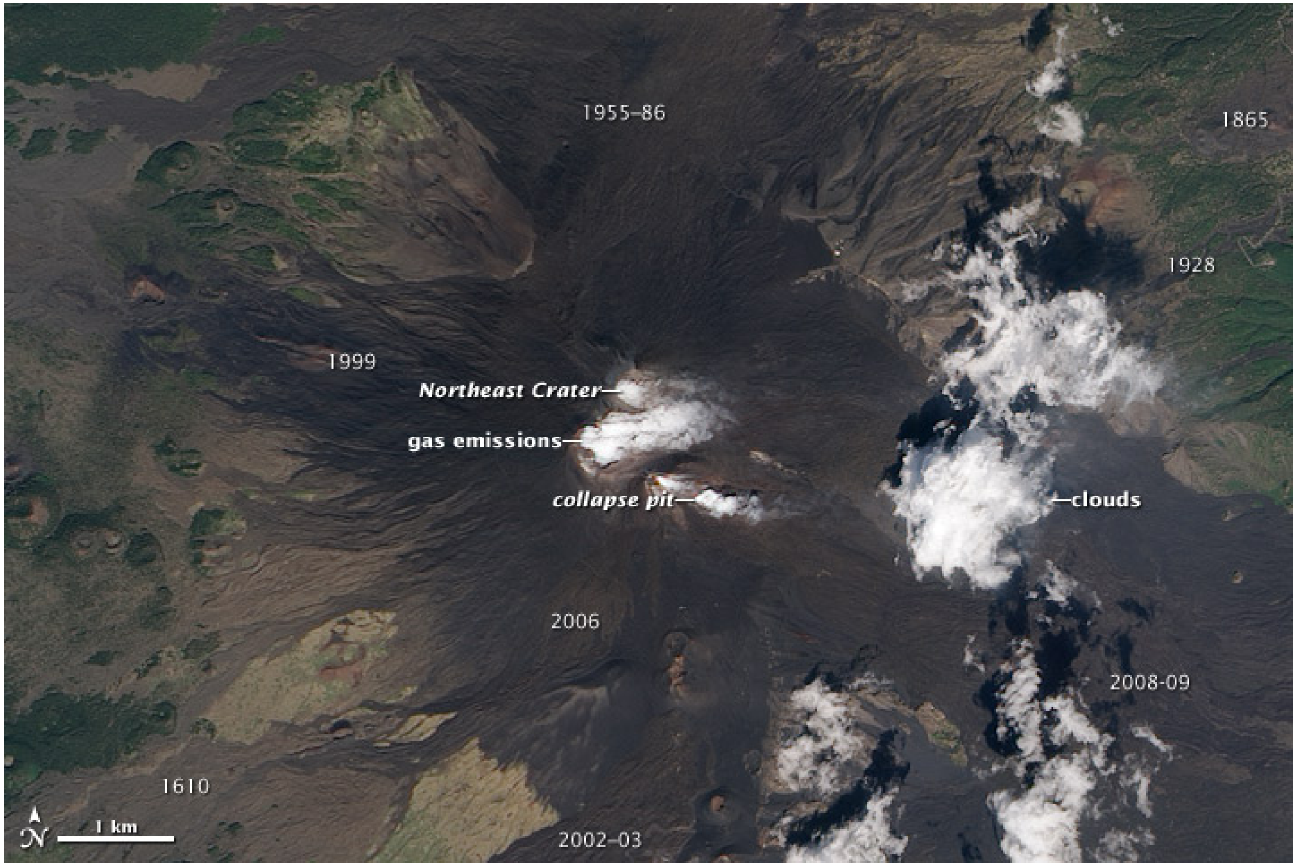
Coordinate Reference Systems (cont'd)

- There are well-known ways to **translate** between co-ordinate reference systems.

- Various authorities maintain lists of coordinate reference systems. See for example:
  - **OGC** http://www.opengis.net/def/crs/
  - **European Petroleum Survey Group** http://www.epsg-registry.org/

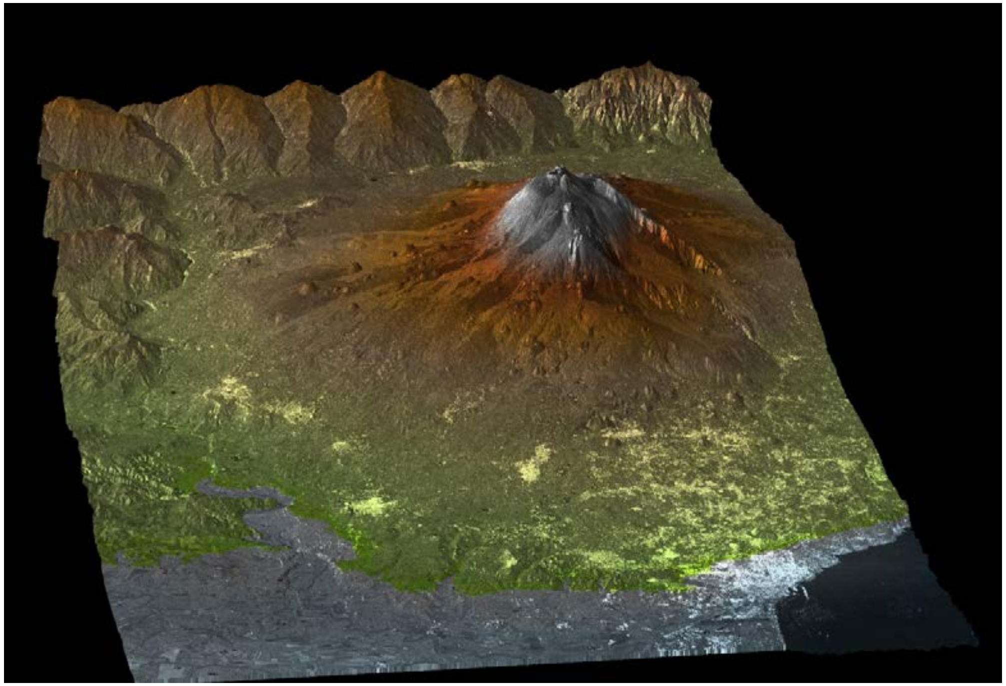
Abstract Modeling Paradigms: Feature-based

- **Feature-based** (or **entity-based** or **object-based**). This kind of modeling is based on the concepts we presented already.



Abstract Modeling Paradigms: Field-based

- Each point *(x,y)* in geographic space is associated with one or several attribute values defined as **continuous functions** in *x* and *y*.
- **Examples:** elevation, precipitation, humidity, temperature for each point *(x,y)* in the Euclidean plane.



From Abstract Modeling to Concrete Representations

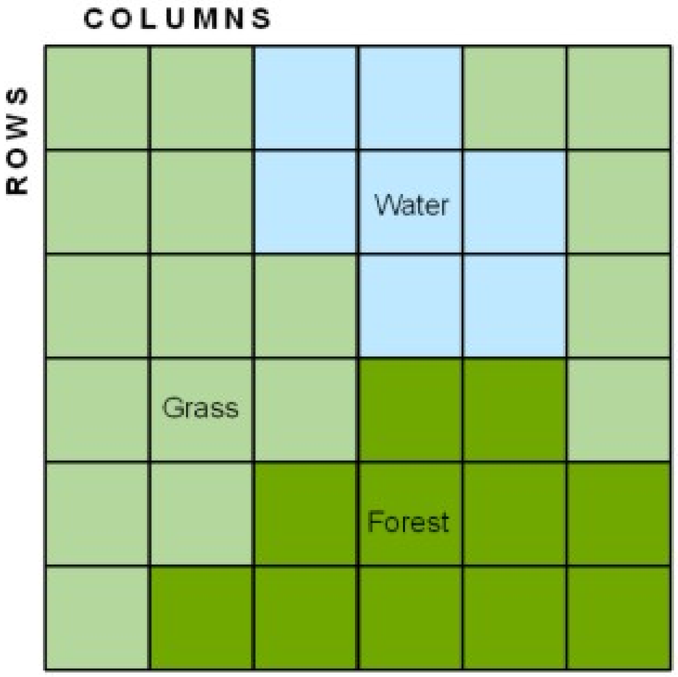
- **Question**: How do we represent the **infinite objects** of the abstract representations (points, lines, fields etc.) **by finite means** (in a computer)?

- **Answers**:
  - **Approximate** the continuous space (e.g., $\mathbb{R}^2$) by a discrete one ($\mathbb{Z}^2$).
  - Use **special encodings**

Concrete Representations - Tessellation

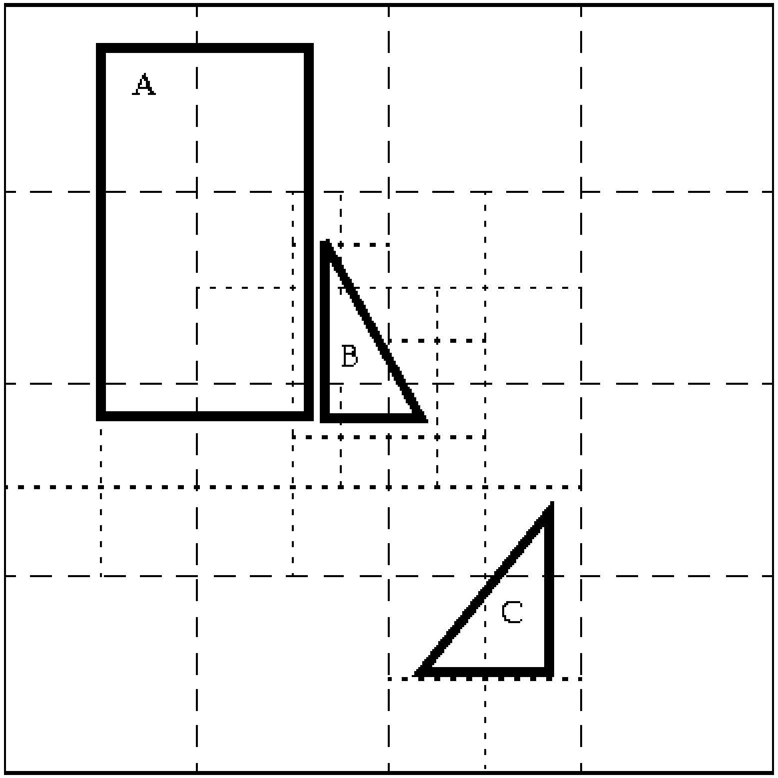
- In this case a **cellular decomposition of the plane** (usually, a grid) serves as a basis for representing the geometry.

- **Example:** raster representation



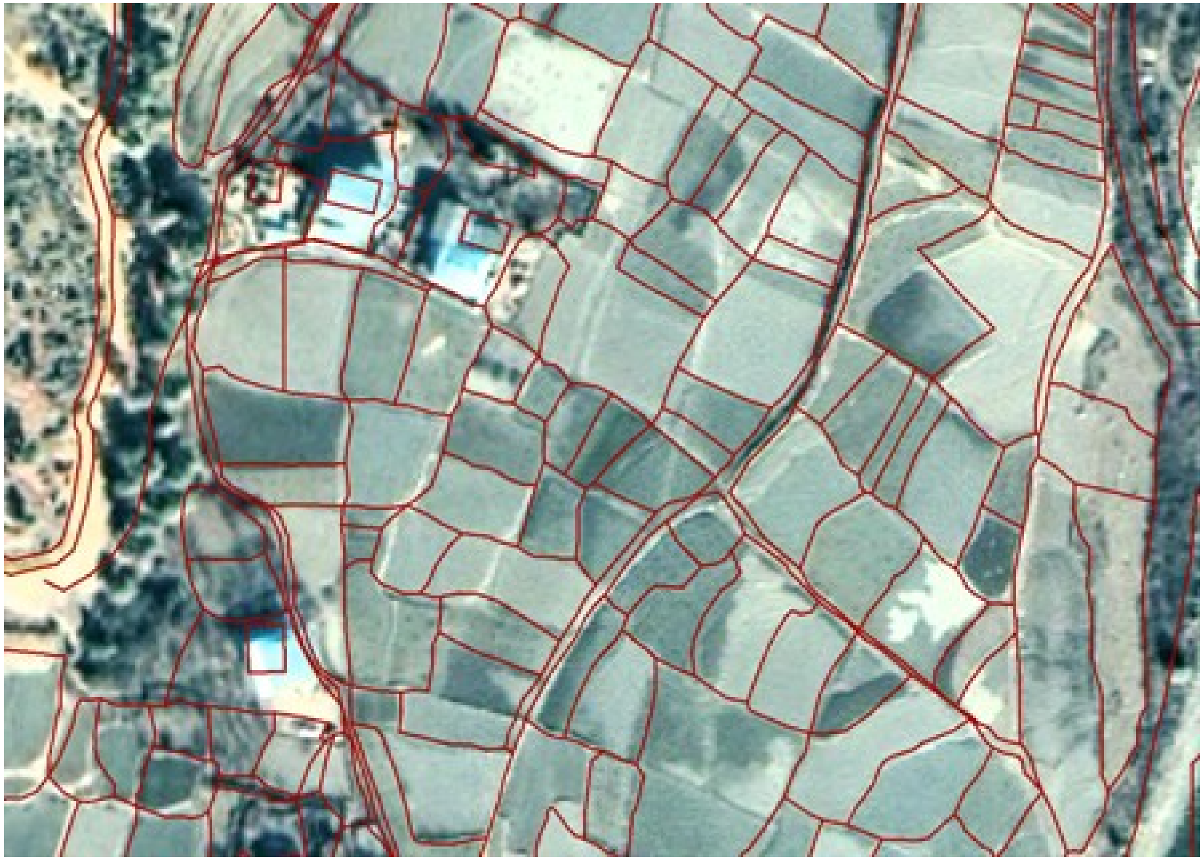
Tessellation (cont'd)

- Tiling (variable sized)



Tessellation (cont'd)

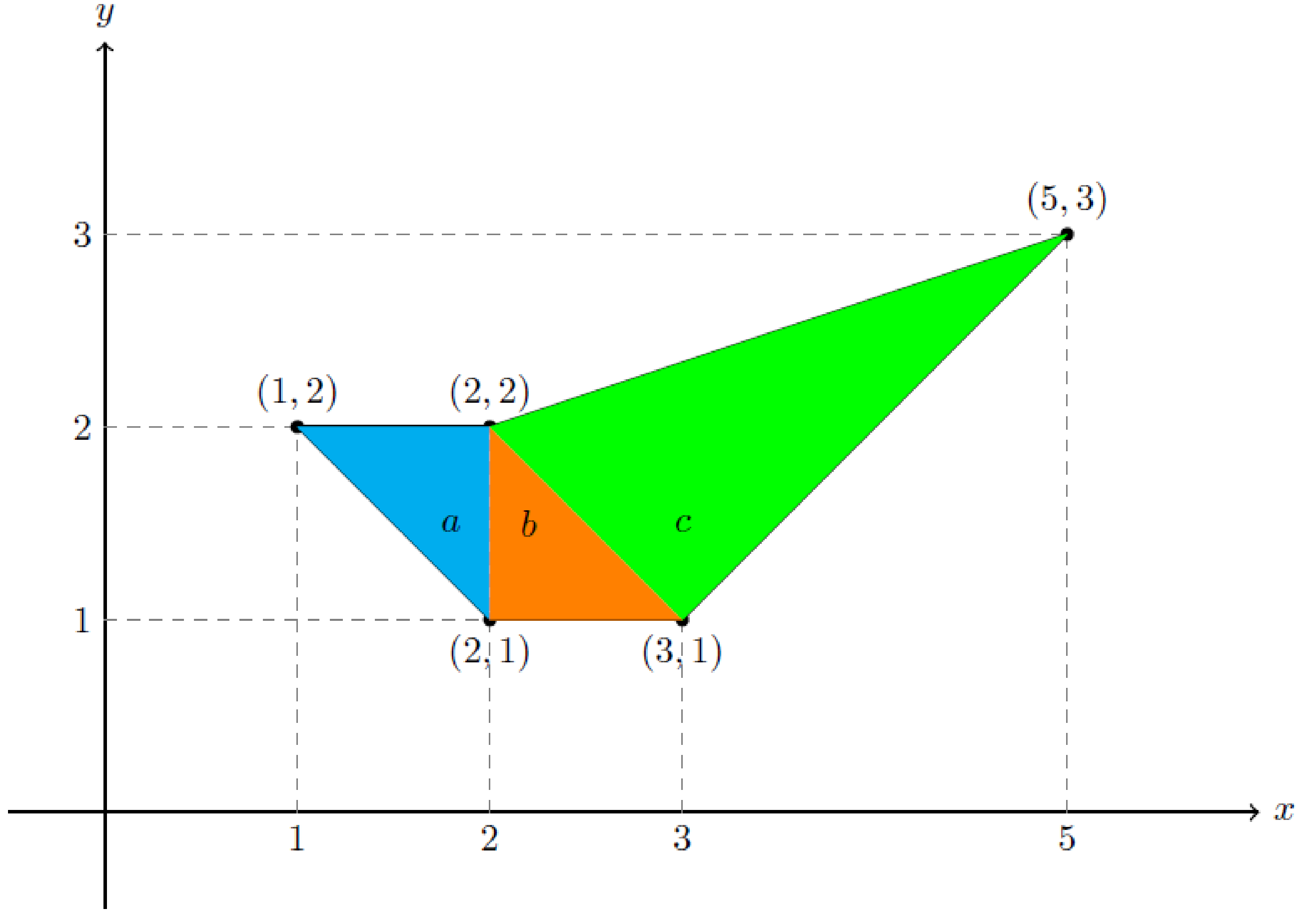
- **Cadastral map (irregular tessellation)** overlayed on a satellite image.



Concrete Representations: Vectors

- In this case objects are constructed from **points** and **line segments** as primitives as follows:
  - A **point** is represented by a tuple of coordinates.
  - A **line segment** is represented by a pair with its beginning and ending point.
  - **More complex objects** such as arbitrary lines, curves, surfaces etc. are built recursively by the basic primitives using constructs such as lists, sets etc.

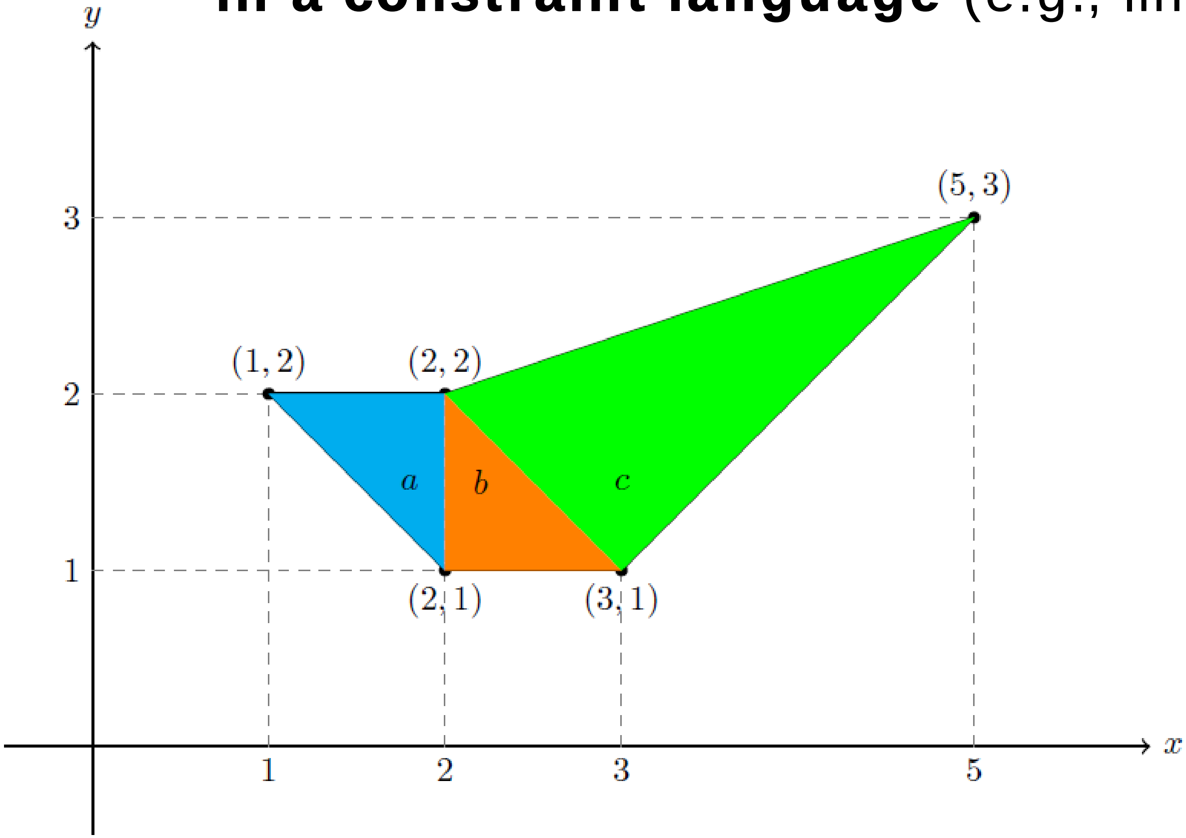
Concrete Representations: Vectors



[(1,2) (2,2) (5,3) (3,1) (2,1) (1 2)]

Concrete Representations: Constraints

- In this case objects are represented by **quantifier free formulas in a constraint language** (e.g., linear constraints).



$$(y + x \geq 3 \wedge x \leq 2 \wedge y \leq 2) \vee (y + x \leq 4 \wedge x \geq 2 \wedge y \geq 1) \vee (y \geq 3 \wedge x \leq 5 \wedge y - \frac{x}{3} \leq \frac{4}{3})$$

Geospatial Data Standards

- The **Open Geospatial Consortium (OGC)** and the **International Organization for Standardization (ISO)** have developed many geospatial data standards that are in wide use today. In this tutorial we will cover:
  - **Well-Known Text**
  - **Geography Markup Language**
  - **OpenGIS Simple Feature Access**

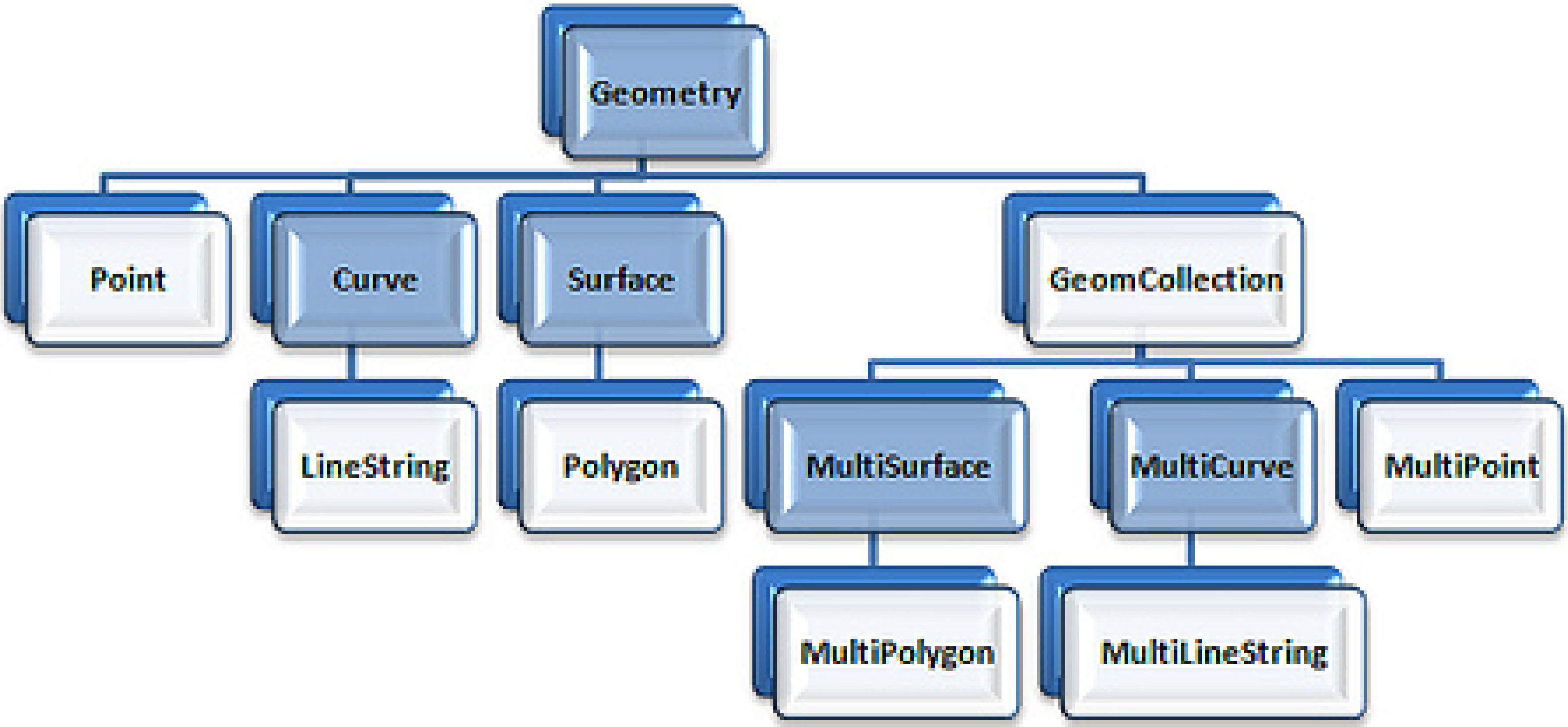
Well-Known Text (WKT)

- WKT is an OGC and ISO standard for representing **geometries**, **coordinate reference systems**, and **transformations** between coordinate reference systems.

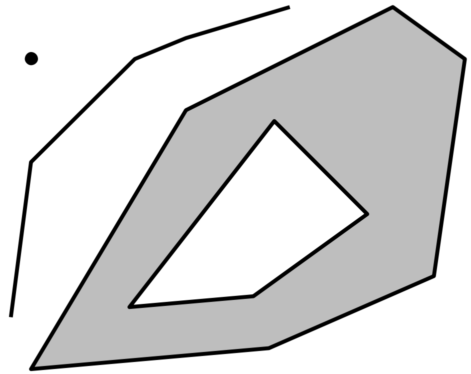
- WKT is specified in **OpenGIS Simple Feature Access - Part 1: Common Architecture** standard which is the same as the **ISO 19125-1** standard. Download from http://portal.opengeospatial.org/files/?artifact_id=25355 .

- This standard concentrates on **simple features:** features with all spatial attributes described piecewise by a straight line or a planar interpolation between sets of points.

WKT Class Hierarchy



Example



WKT representation:

```
GeometryCollection(
  Point(5 35),
  LineString(3 10,5 25,15 35,20 37,30 40),
  Polygon((5 5,28 7,44 14,47 35,40 40,20 30,5 5),
          (28 29,14.5 11,26.5 12,37.5 20,28 29))
  )
```

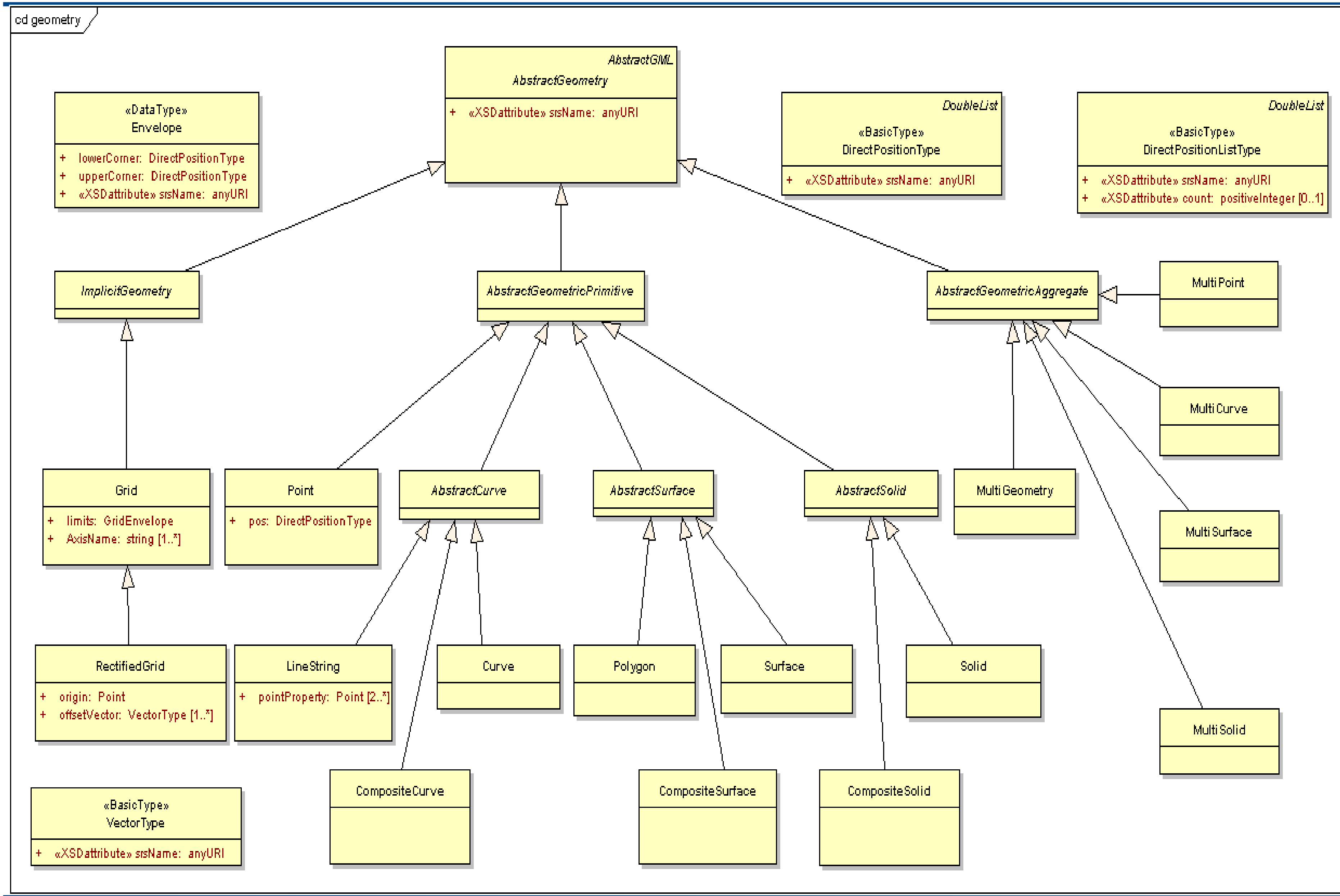
Geography Markup Language (GML)

- **GML** is an **XML-based encoding standard** for the representation of geospatial data.

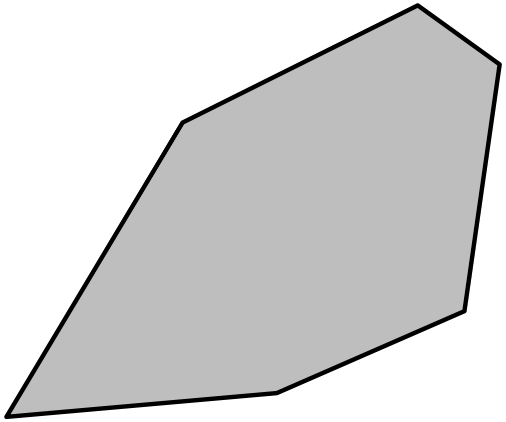
- GML provides XML schemas for defining a variety of concepts: **geographic features, geometry, coordinate reference systems, topology, time** and **units of measurement**.

- **GML profiles** are subsets of GML that target particular applications.
  - **Examples**: Point Profile, GML Simple Features Profile etc.

GML Simple Features: Class Hierarchy



Example



GML representation:

```
<gml:Polygon gml:id="p3" srsName="urn:ogc:def:crs:EPSG:6.6:4326”>
    <gml:exterior>
        <gml:LinearRing>
            <gml:coordinates>
                5,5 28,7 44,14 47,35 40,40 20,30 5,5
            </gml:coordinates>
        </gml:LinearRing>
    </gml:exterior>
</gml:Polygon>
```

OpenGIS Simple Features Access (cont'd)

- OGC has also specified a standard for the **storage, retrieval, query and update of sets of simple features using relational DBMS and SQL.**

- This standard is **"OpenGIS Simple Feature Access - Part 2: SQL Option"** and it is the same as the **ISO 19125-2** standard. Download from http://portal.opengeospatial.org/files/?artifact_id=25354.

- **Related standard**: ISO 13249 SQL/MM - Part 3.

OpenGIS Simple Features Access (cont'd)

- The standard covers two implementations options: (i) using **only the SQL predefined data types** and (ii) using **SQL with geometry types**.

- **SQL with geometry types:**
  - We use the WKT geometry class hierarchy presented earlier to define **new geometric data types for SQL**
  - We define new **SQL functions on those types**.

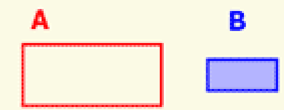
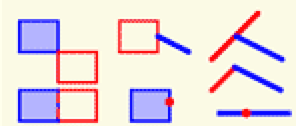
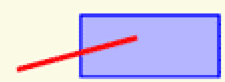
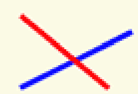
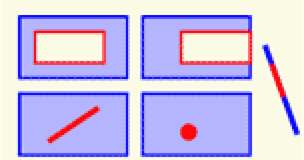
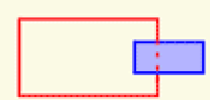
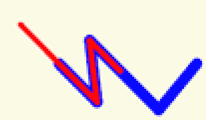
SQL with Geometry Types - Functions

- Functions that **request or check properties** of a geometry:
  - ST Dimension(A:Geometry, B:Geometry):Integer
  - ST_GeometryType(A:Geometry):Character Varying
  - ST_AsText(A:Geometry): Character Large Object
  - ST_AsBinary(A:Geometry): Binary Large Object
  - ST_SRID(A:Geometry): Integer
  - ST_IsEmpty(A:Geometry): Boolean
  - ST_IsSimple(A:Geometry): Boolean

SQL with Geometry Types – Functions (cont'd)

- Functions that test **topological spatial relationships** between two geometries using the **DE-9IM**:
  - `ST_Equals(A:Geometry, B:Geometry):Boolean`
  - `ST_Disjoint(A:Geometry, B:Geometry):Boolean`
  - `ST_Intersects(A:Geometry, B:Geometry):Boolean`
  - `ST_Touches(A:Geometry, B:Geometry):Boolean`
  - `ST_Crosses(A:Geometry, B:Geometry):Boolean`
  - `ST_Within(A:Geometry, B:Geometry):Boolean`
  - `ST_Contains(A:Geometry, B:Geometry):Boolean`
  - `ST_Overlaps(A:Geometry, B:Geometry):Boolean`
  - `ST_Relate(A:Geometry, B:Geometry, Matrix: Char(9)):Boolean`

DE-9IM Relation Definitions

Beziehung	Definition	Beispiele
A disjoint B	$\begin{bmatrix} F & F & * \\ F & F & * \\ * & * & * \end{bmatrix}$	
A touches B (d(A) > 0 ∨ d(B) > 0)	$\begin{bmatrix} F & T & * \\ * & * & * \\ * & * & * \end{bmatrix} \vee \begin{bmatrix} F & * & * \\ T & * & * \\ * & * & * \end{bmatrix} \vee \begin{bmatrix} F & * & * \\ * & T & * \\ * & * & * \end{bmatrix}$	
A crosses B (d(A) < d(B))	$\begin{bmatrix} T & * & T \\ * & * & * \\ * & * & * \end{bmatrix}$	
A crosses B (d(A) = d(B) = 1)	$\begin{bmatrix} 0 & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$	
A within B	$\begin{bmatrix} T & * & F \\ * & * & F \\ * & * & * \end{bmatrix}$	
A overlaps B (d(A) = d(B), d(A) ≠ 1, d(B) ≠ 1)	$\begin{bmatrix} T & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$	
A overlaps B (d(A) = d(B) = 1)	$\begin{bmatrix} 1 & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$	

SQL with Geometry Types – Functions (cont'd)

- Functions for **constructing new geometries** out of existing ones:
  - `ST Boundary(A:Geometry):Geometry`
  - `ST_Envelope(A:Geometry):Geometry`
  - `ST_Intersection(A:Geometry, B:Geometry):Geometry`
  - `ST_Union(A:Geometry, B:Geometry):Geometry`
  - `ST_Difference(A:Geometry, B:Geometry):Geometry`
  - `ST_SymDifference(A:Geometry, B:Geometry):Geometry`
  - `ST_Buffer(A:Geometry, distance:Double):Geometry`

Geospatial Relational DBMS

- The OpenGIS Simple Features Access Standard is today been used in all **relational DBMS with a geospatial extension.**

  - The **abstract data type mechanism** of the DBMS allows the representation of all kinds of geospatial data types supported by the standard.
  - The query language (SQL) offers the **functions** of the standard for querying data of these types.

PostGIS
Spatial PostgreSQL

ORACLE DATABASE 10g

Conclusions

- **Background in geospatial data modeling**:
  - Why geographical information?
  - Geographical Information Science and Systems
  - Geospatial data on the Web and linked geospatial data
  - Abstract geographic space modeling paradigms: discrete objects vs. continuous fields
  - Concrete representations: tessellation vs. vectors vs. constraints
  - Geospatial data standards

- **Next topic:** Geospatial data in the Semantic Web

Geospatial data in the Semantic Web

stSPARQL

Presenter: Kostis Kyzirakos

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, 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 the SPARQL query language utilizing appropriate geometric vocabularies and ontologies (e.g., the topological relationships of RCC8).
- 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.

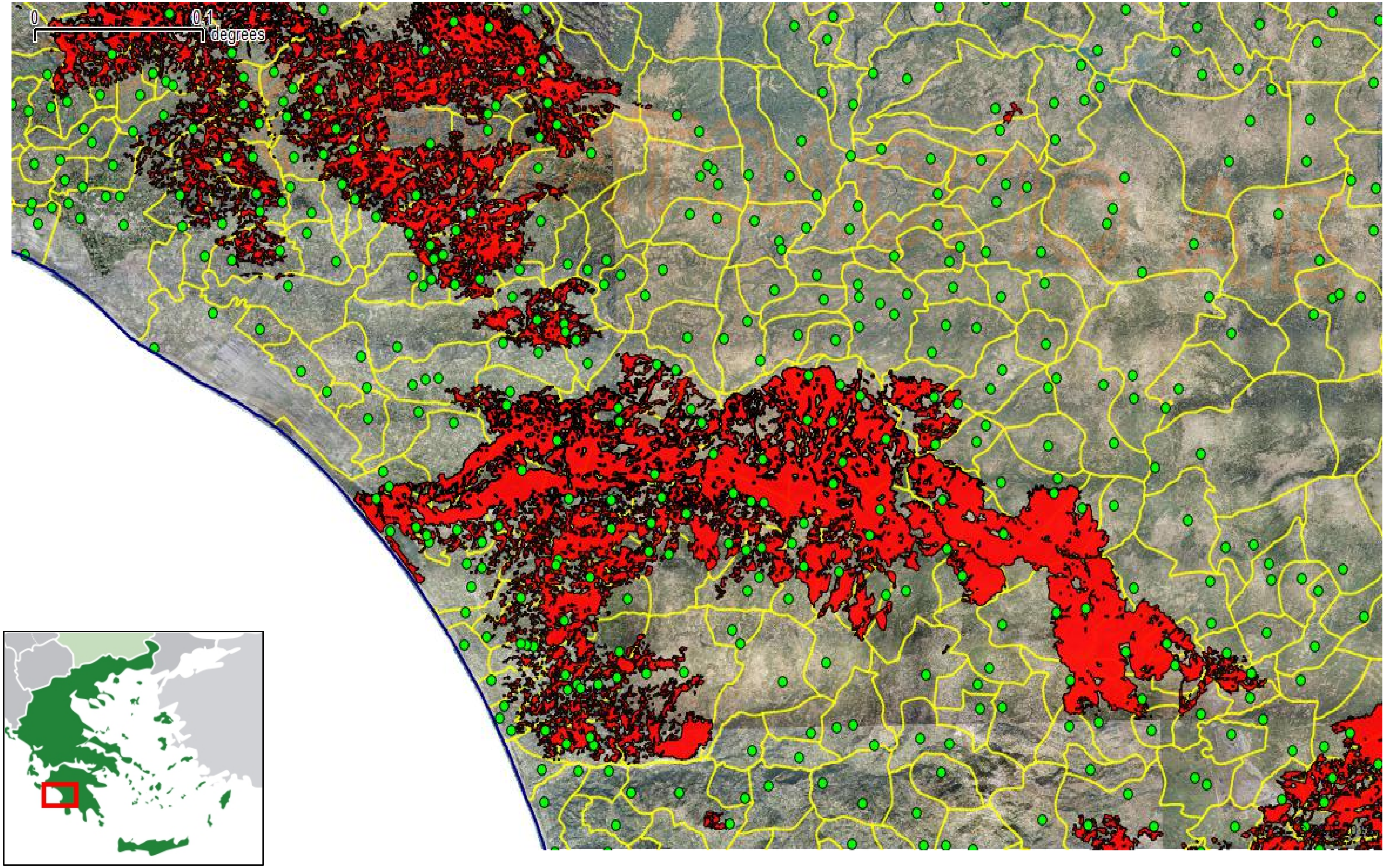
- SPARQL-ST provides a set of built-in spatial conditions that can be used in **SPATIAL FILTER** clauses to constrain the geometries that are returned as answers to queries.

stRDF and stSPARQL

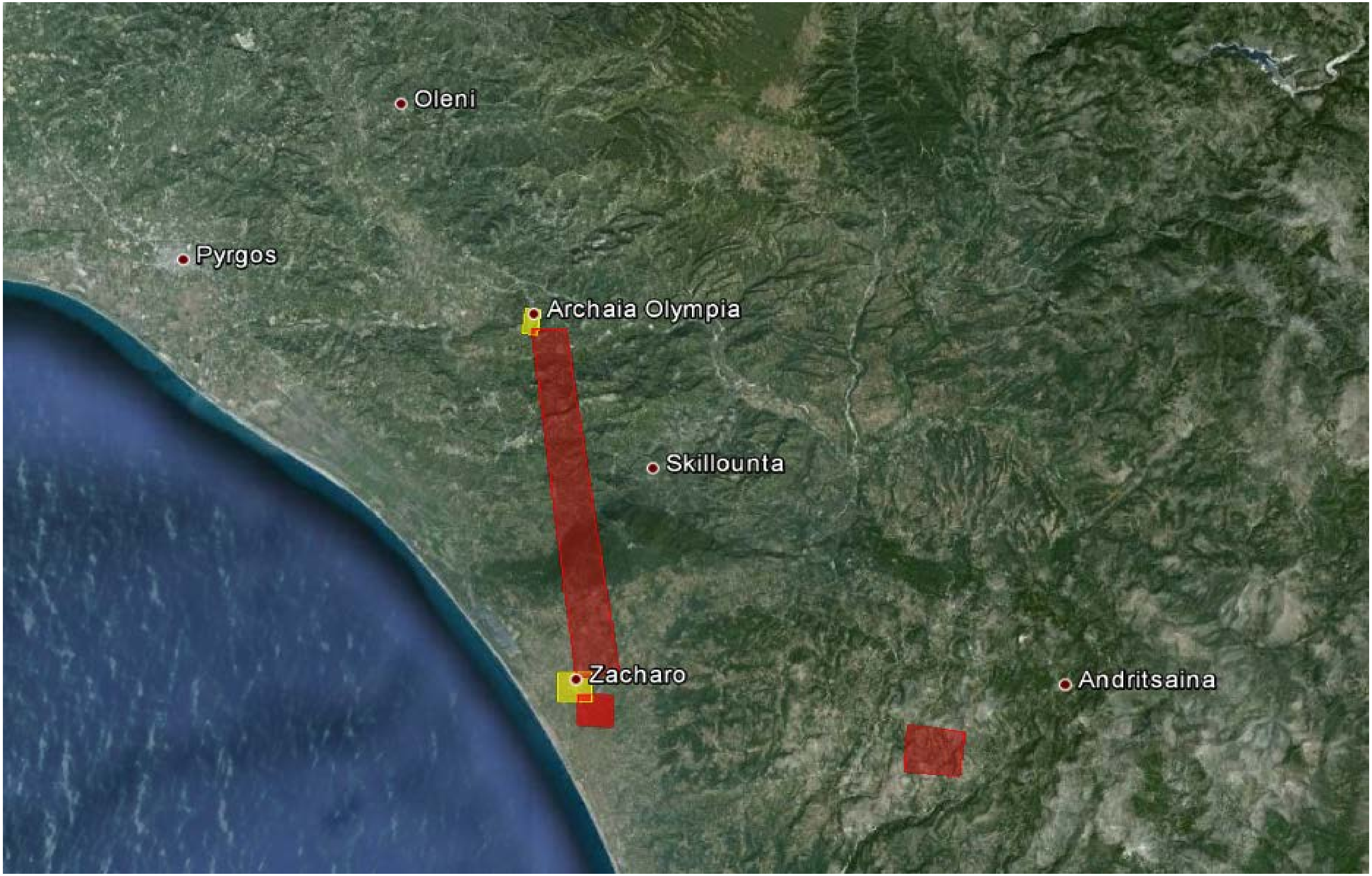
(Koubarakis and Kyzirakos, 2010)

- Similar approach to SPARQL-ST (**theme**, **space** and **valid time** can be represented)
- **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 .
```

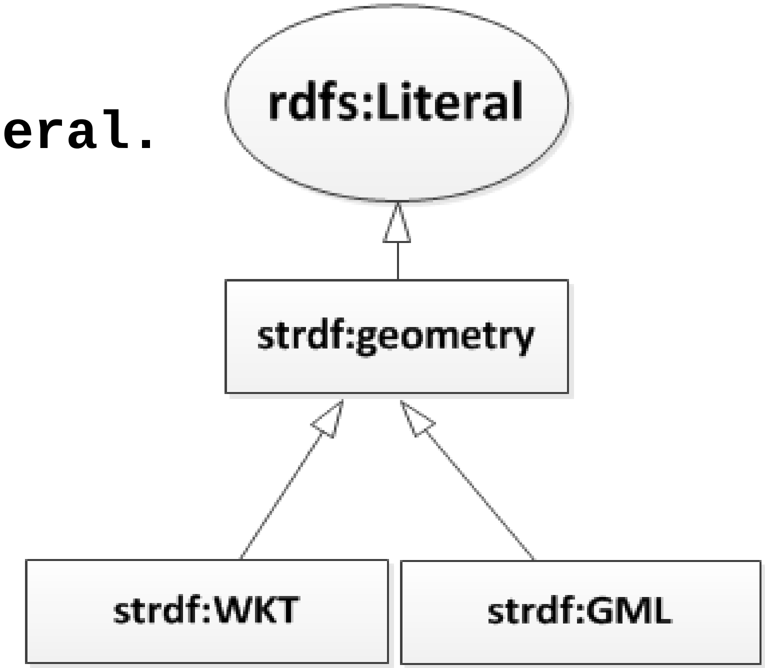
Spatial literal

```
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

Analipsi
Gouves Kriti
Goyrnes
Εθνική Οδός Ηρακλείου
Gouves
Chersonisos
Sisi
Stalida
Malia
Kalo Chorio
Mochos
E75
90
2 mi

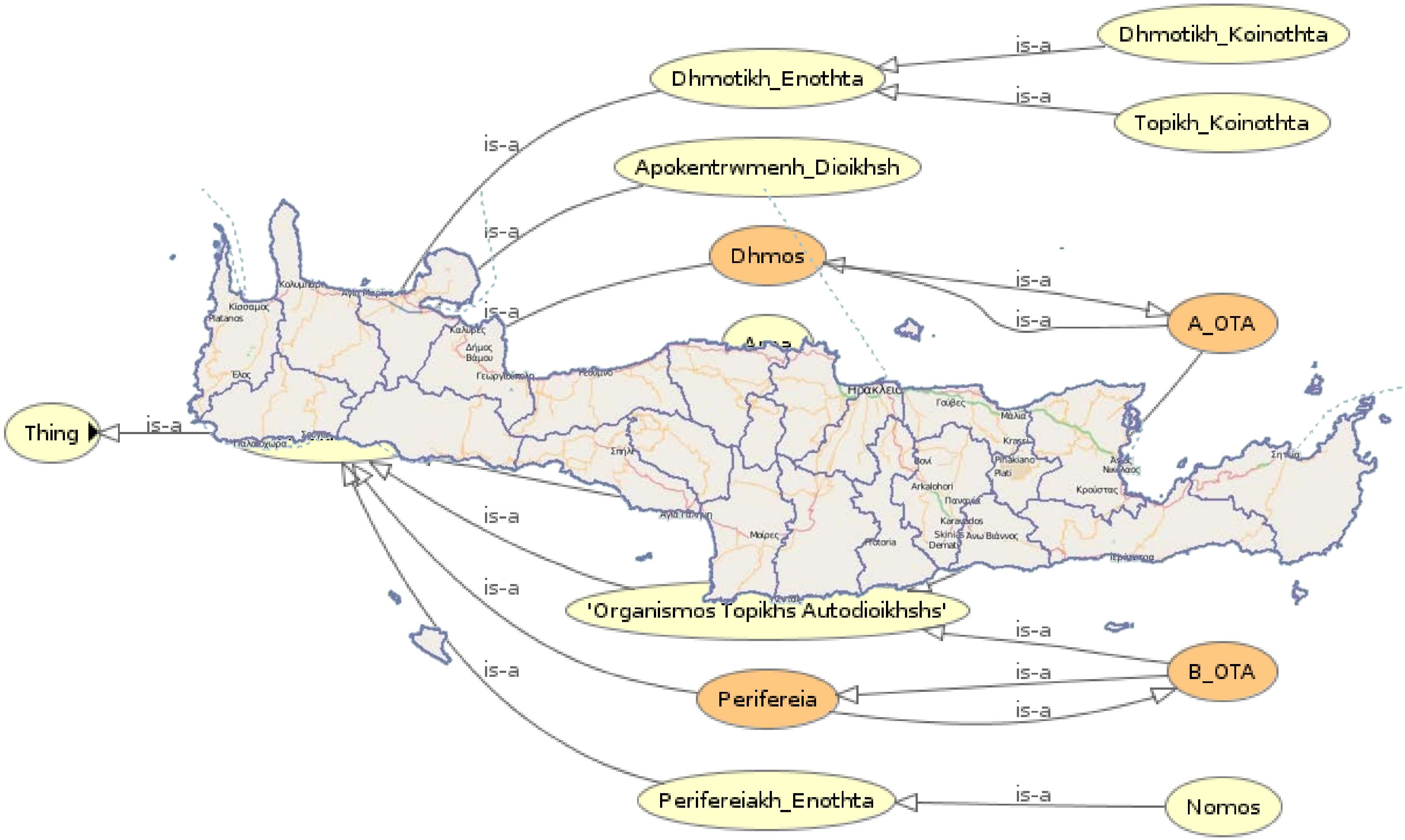
Geonames

```
geonames:260001   rdf:type            gn:Feature;
                  gn:name             "Hersonissos";
                  gn:officialName     "Χερσόνησος"@el;
                  gn:countryCode      "GR";
                  wgs84_pos:lat       "35.30903";
                  wgs84_pos:long      "25.37112";
                  gn:parentCountry    geonames:390903.

geonames:390903   gn:name             "Greece".
```

Greek Administrative Geography

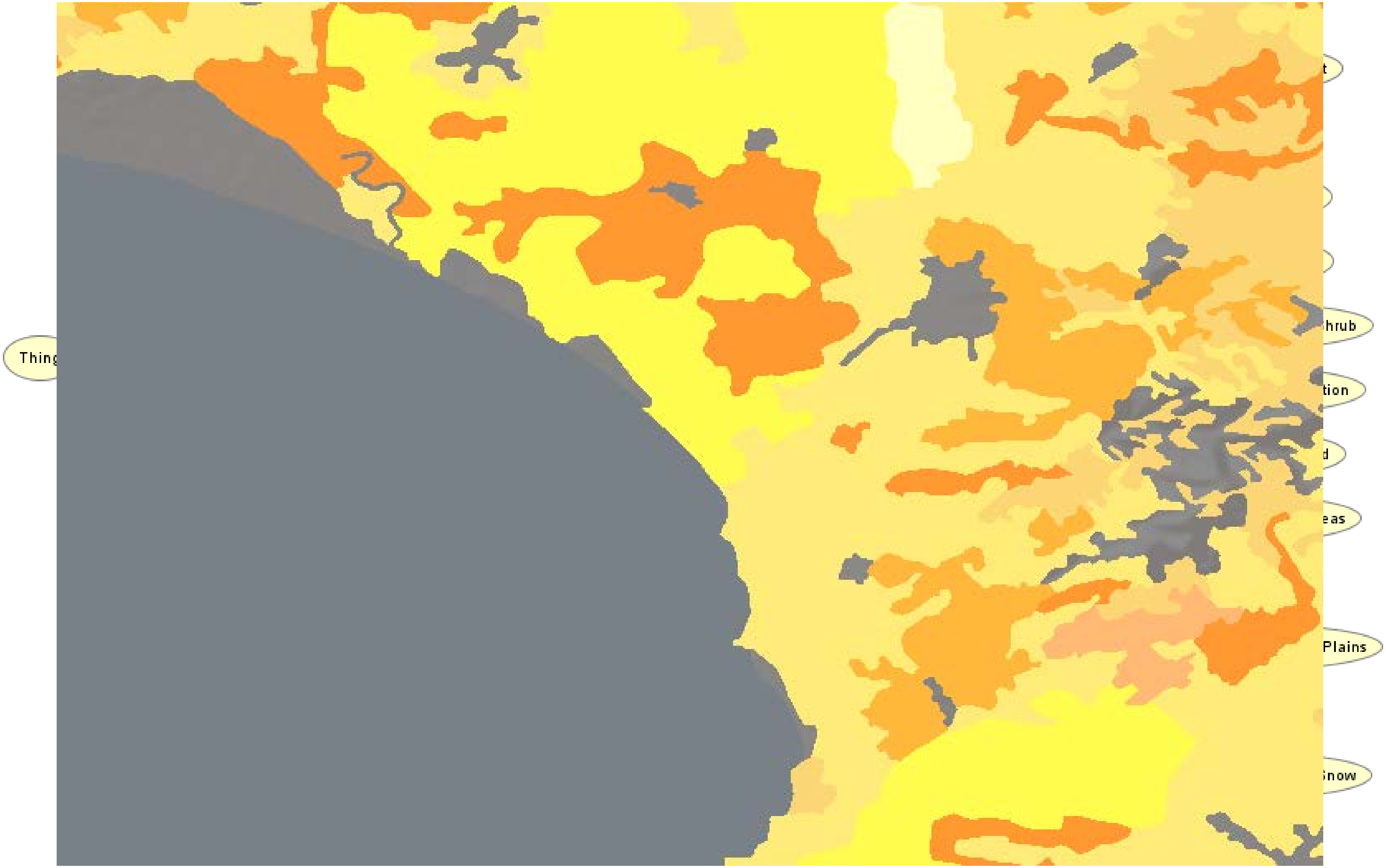
- Kallikrates ontology



Greek Administrative Geography

```
gag:gag003000009002 rdf:type owl:NamedIndividual ;
                    rdf:type gag:Dhmos;
                    rdfs:label "ΔΗΜΟΣ ΧΕΡΣΟΝΗΣΟΥ"@el;
                    rdfs:label "Hersonissos";
                    noa:hasYpesCode "9309"^^xsd:integer;
                    strdf:hasGeometry
                           "MULTIPOLYGON (((
                              25.37 35.34,
                              ...,
                              25.21 35.47)))"^^strdf:WKT;
                    gag:isPartOf gag:gag003000000101.
```

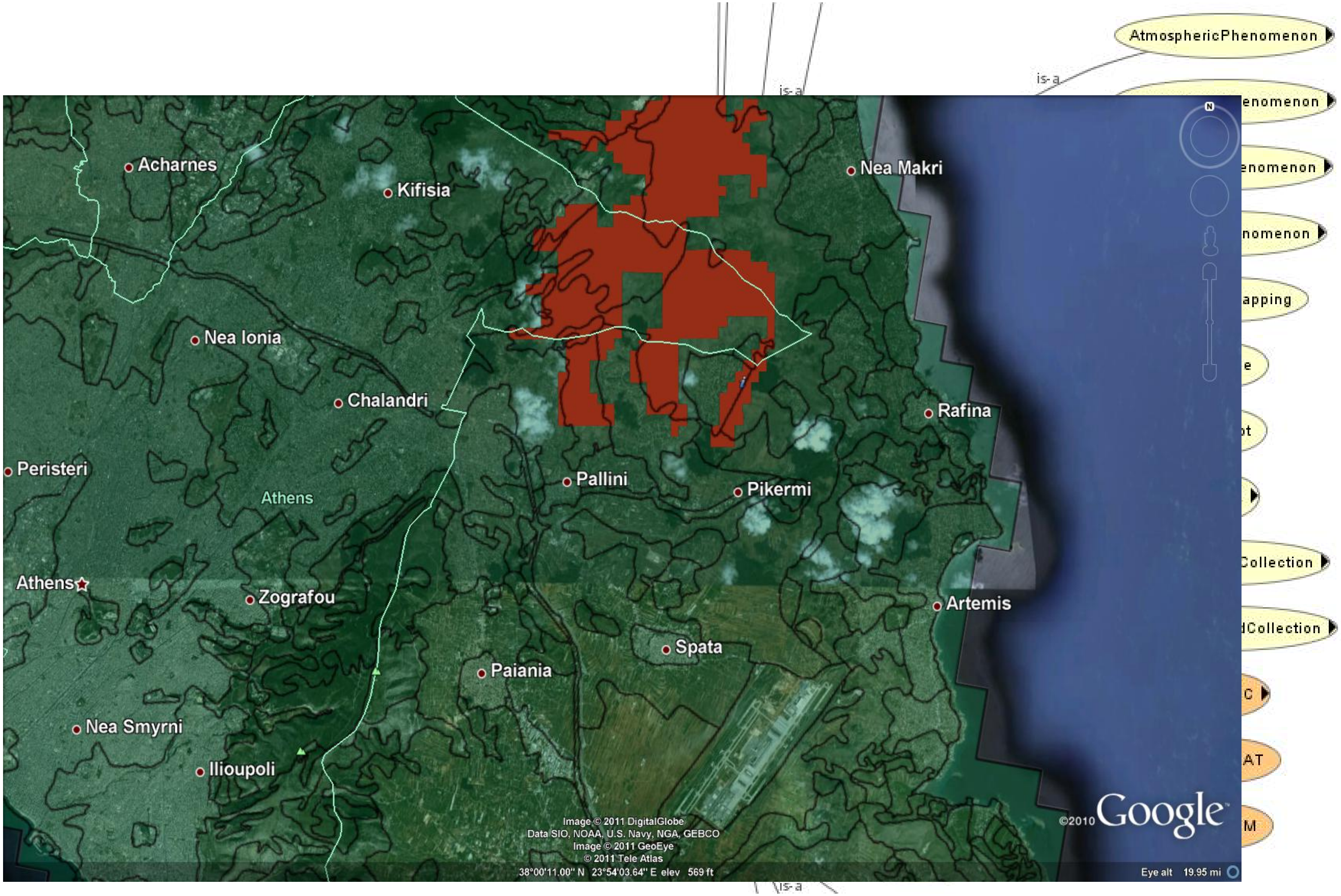
Corine Land Use / Land Cover



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 "255.580790497"^^xsd:double;
                    strdf:hasGeometry "POLYGON((15.53 62.54, ...,
                                    15.53 62.54))"^^strdf:WKT;
                    noa:hasLandUse noa:coniferousForest
```

Burnt Area Products



Burnt Area Products

```
noa:ba_15 rdf:type noa:BurntArea;
          noa:isDerivedFromSatellite "Landsat"^^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**
- **RCC8**

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

GeoNames

```
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.

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

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

Spatial Aggregate

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:** The query language GeoSPARQL

Geospatial data in the Semantic Web

GeoSPARQL

Presenter: Kostis Kyzirakos

GeoSPARQL

GeoSPARQL is a recently completed OGC standard (Perry and Herring, 2012).

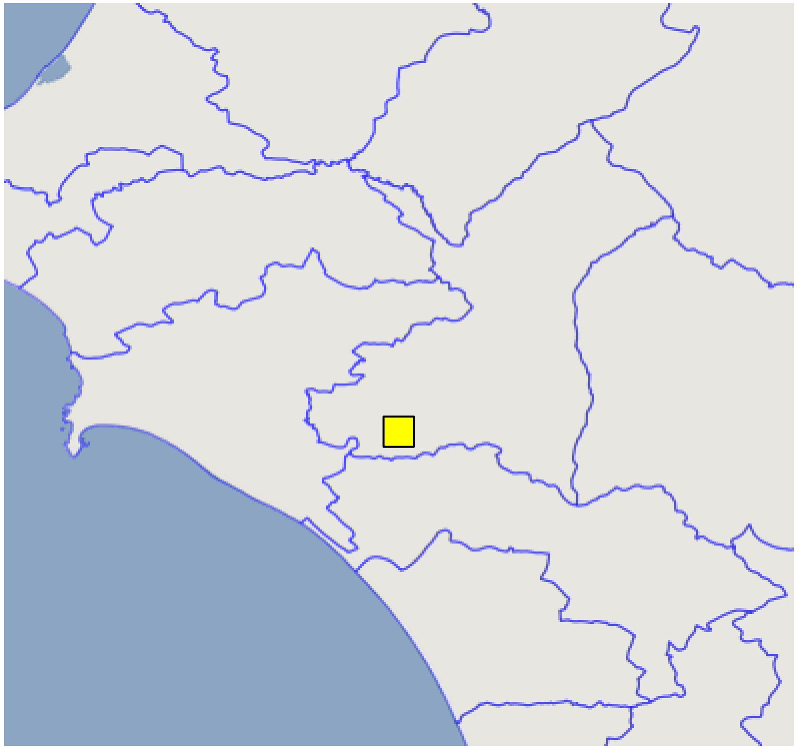
Functionalities **similar to stSPARQL**:

- 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)



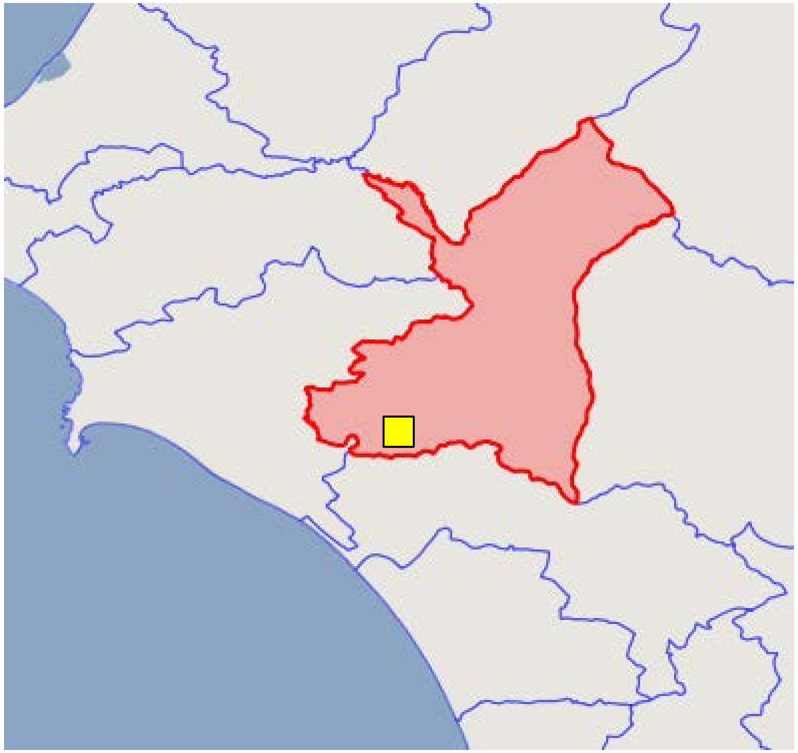
```
geonames:Olympia
  geonames:name "Ancient Olympia";
  rdf:type dbpedia:Community ;
  geo:hasGeometry ex:polygon1.
```

Spatial literal

```
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 data type

Example in GeoSPARQL (2/2)

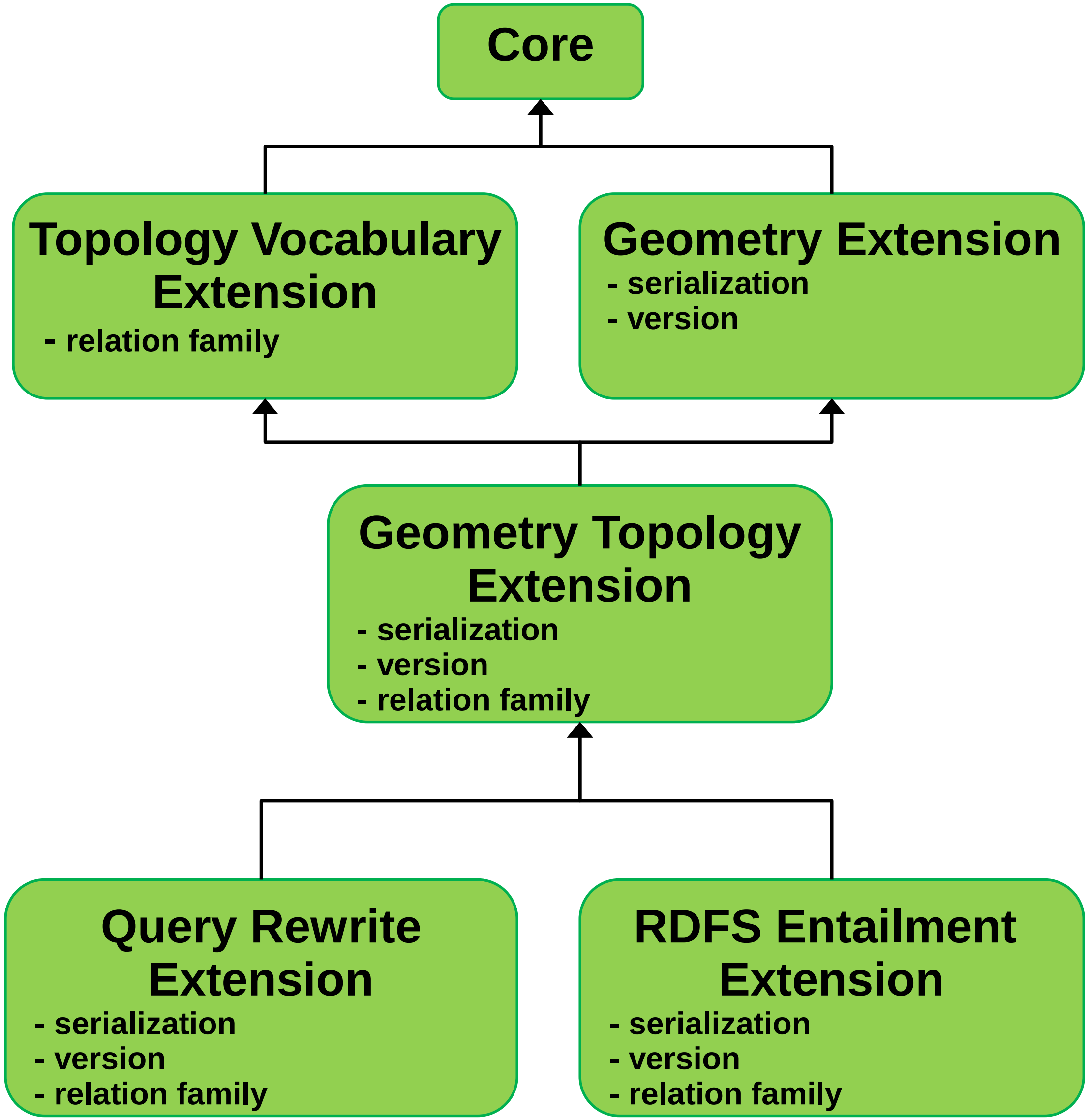


```
gag:OlympiaMunicipality
   rdf:type gag:Municipality;
   rdfs:label "ΔΗΜΟΣ ΑΡΧΑΙΑΣ
               ΟΛΥΜΠΙΑΣ"@el;
   rdfs:label "Municipality of
               Ancient Olympia".
```

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

Asserted topological relation

GeoSPARQL Components



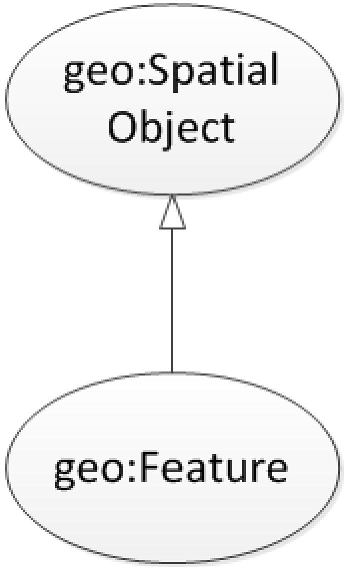
Parameters

- **Serialization**
  - WKT
  - GML
- **Relation Family**
  - Simple Features
  - RCC8
  - 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

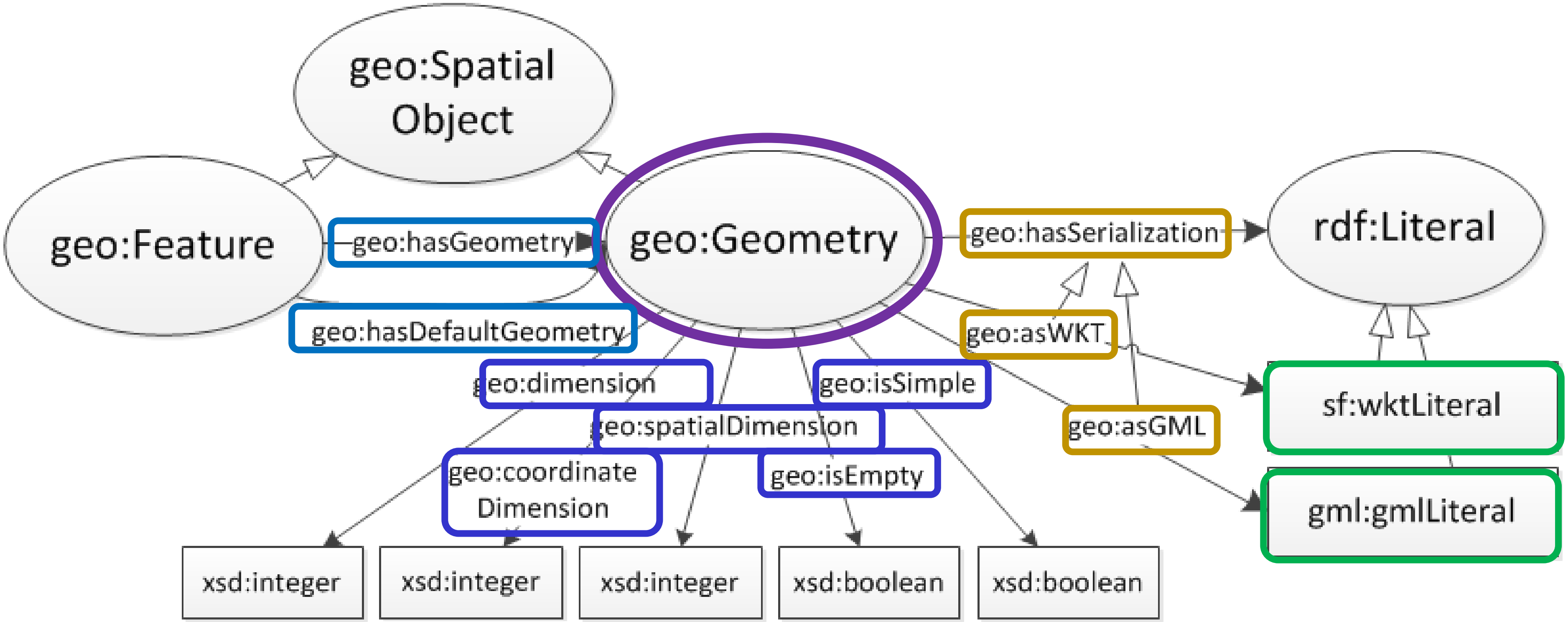
GeoSPARQL representation of the community of Ancient Olympia.

```
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 the top class and has as instances everything that can have a spatial representation.



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: An example

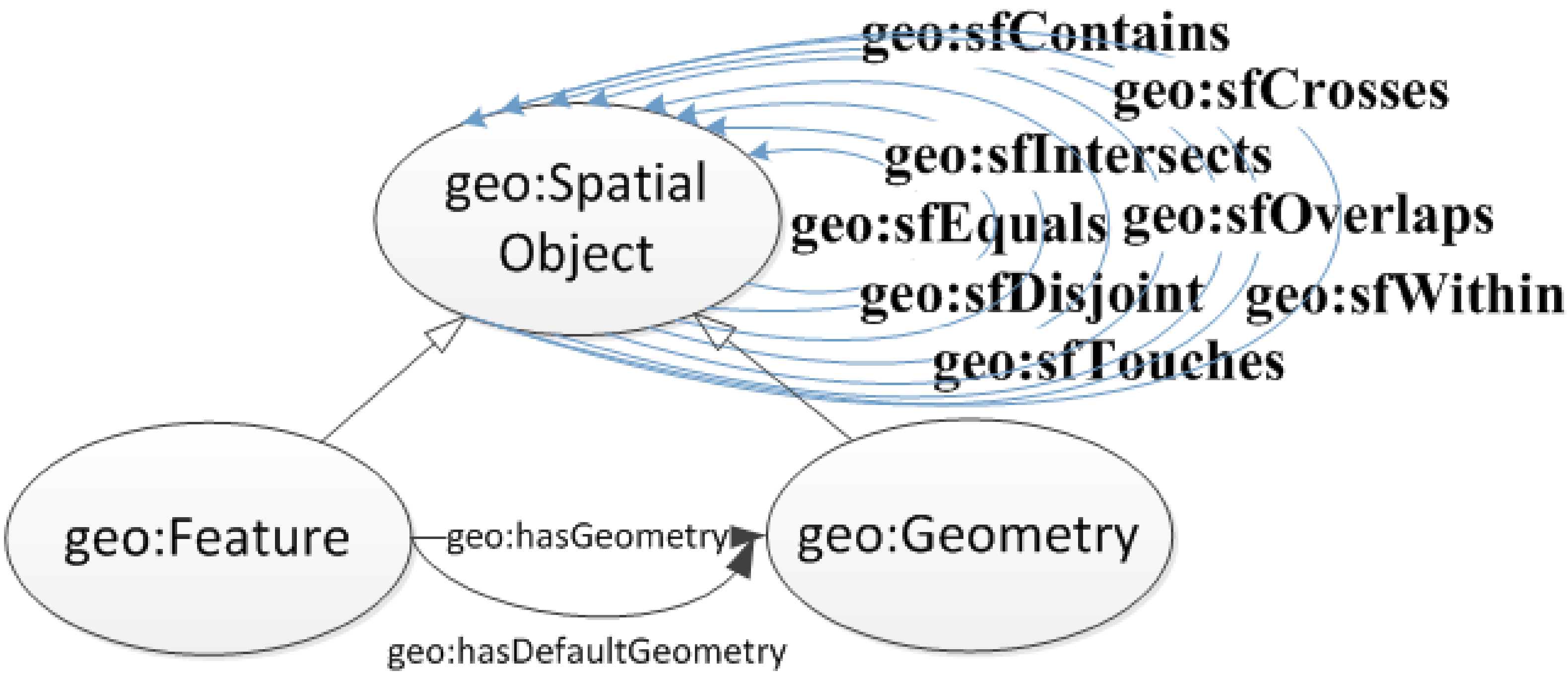
Return the names of communities that are near burnt areas

```
SELECT ?name
WHERE {
  ?c      rdf:type geo:Feature;
          rdf:type dbpedia:Community;
          geonames:name ?name;
          geo:hasGeometry ?cPoly.
  ?cPoly  geo:asWKT ?cGeom.
  ?ba  rdf:type geo:Feature;
       rdf:type noa:BurntArea;
       geo:hasGeometry ?baPoly.
  ?baPoly geo:asWKT ?baGeom.
  FILTER(geof:distance(?cGeom, ?baGeom,
                             uom:metre) < 1500)}
```

Spatial Metric Function

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 RCC8 relations e.g., **`geo:rcc8ec`**

Example

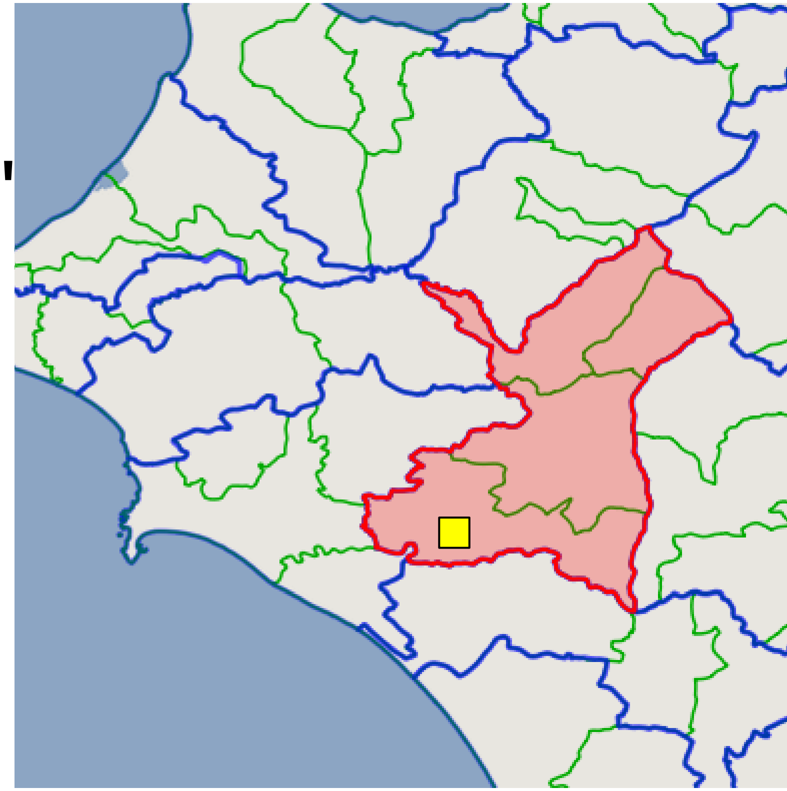
```
geonames:Olympia
  rdf:type dbpedia:Community;
  geonames:name "Ancient Olympia"

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
  - RCC8

GeoSPARQL: An example

Return the names of communities that have been affected by fires

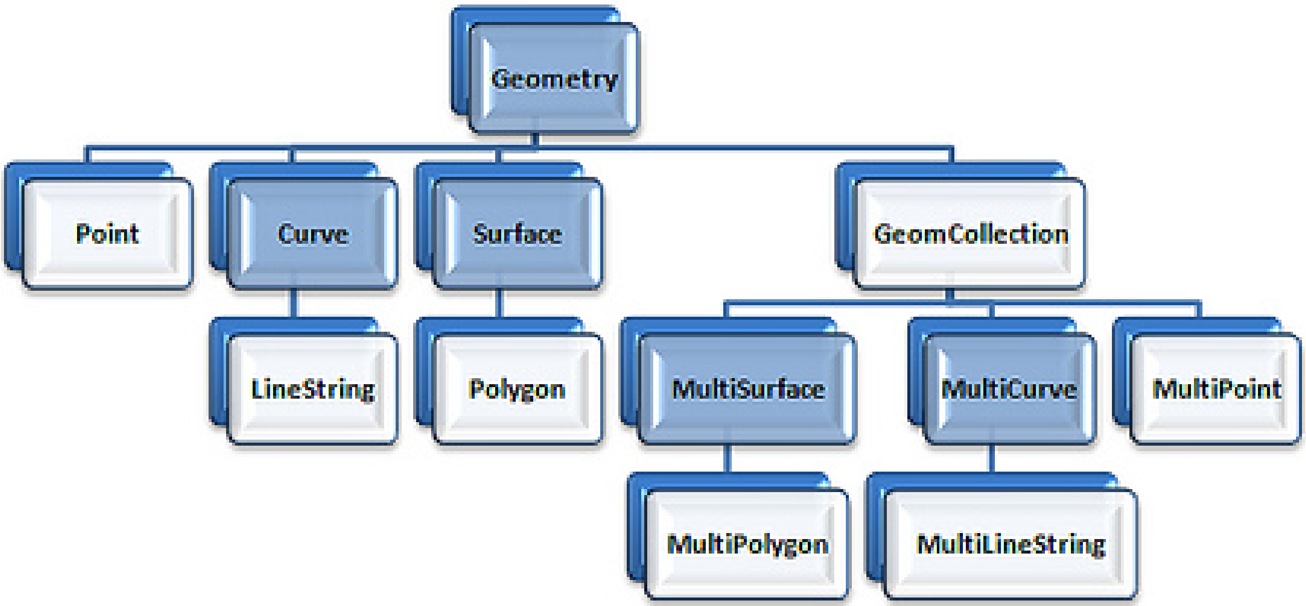
```
SELECT   ?name
WHERE {
  ?community rdf:type dbpedia:Community;
             geonames:name ?name;
             geo:hasGeometry ?cPoly.
  ?cPoly   geo:asWKT ?cGeom.
  ?ba  a noa:BurntArea;
       geo:hasGeometry ?baPoly.
  ?baPoly geo:asWKT ?baGeom.

FILTER(geof:sfIntersects(?cGeom,?baGeom))
}
```

Spatial Function

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 Systems and Applications

Implemented Systems

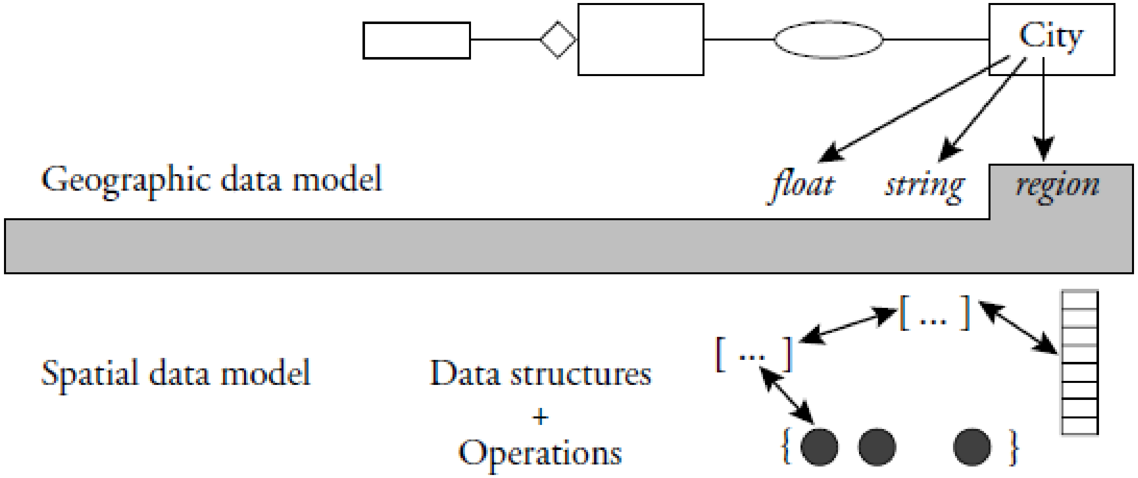
Presenter: Manos Karpathiotakis

Outline

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

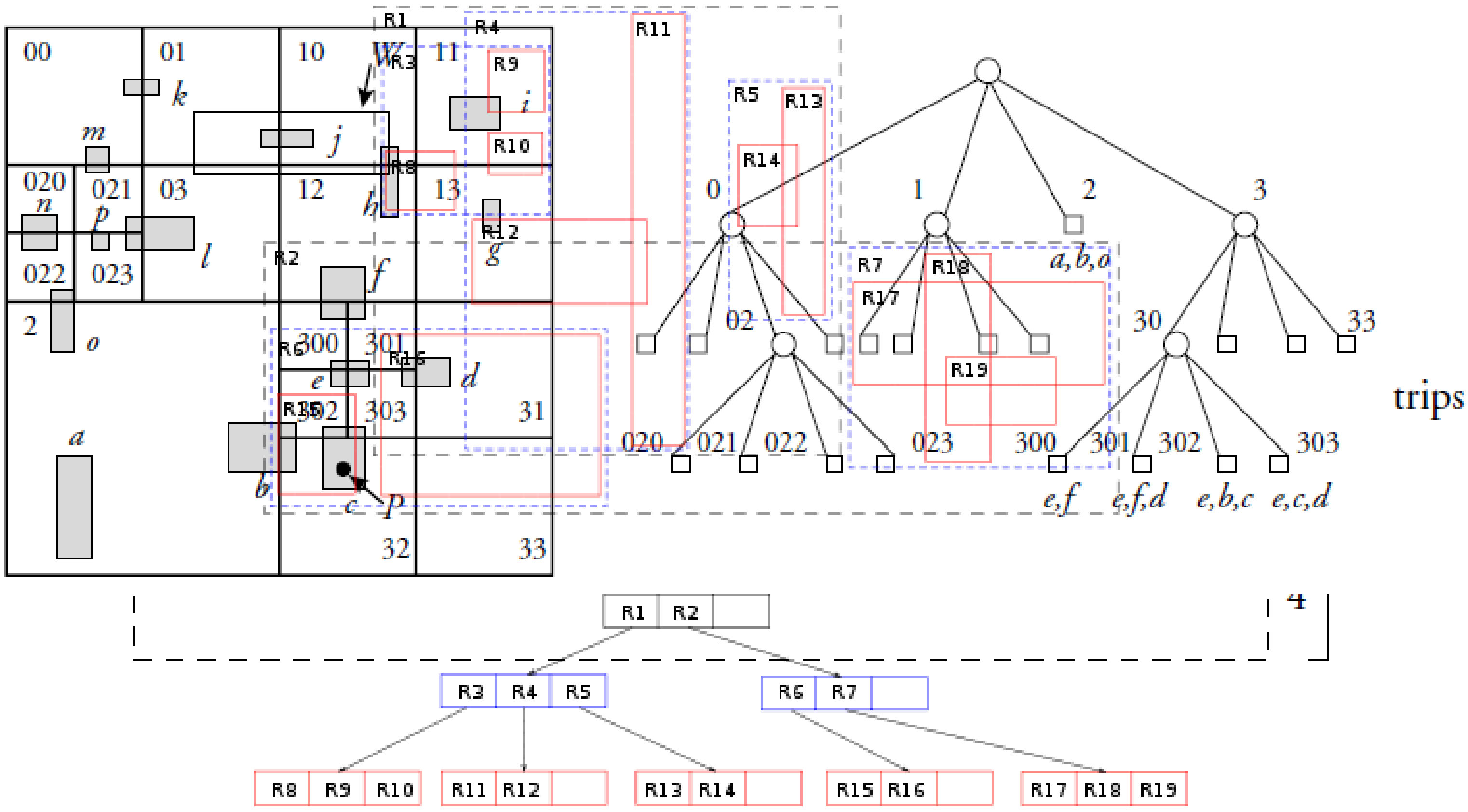
How does an RDBMS handle geometries? (1/2)

- 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

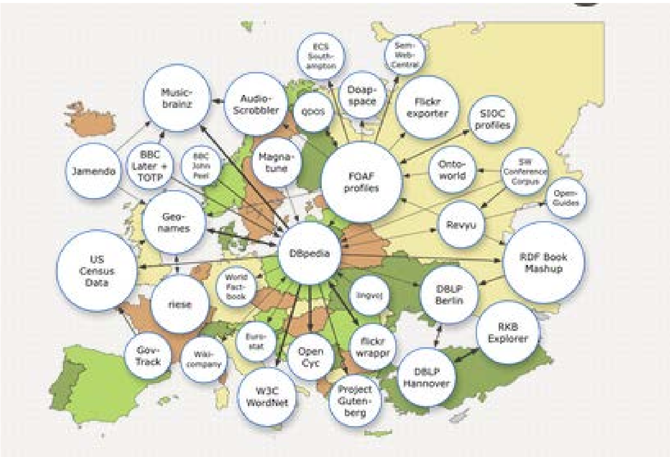
Research Prototypes

- Strabon
- Parliament
- Brodt et al.
- Perry

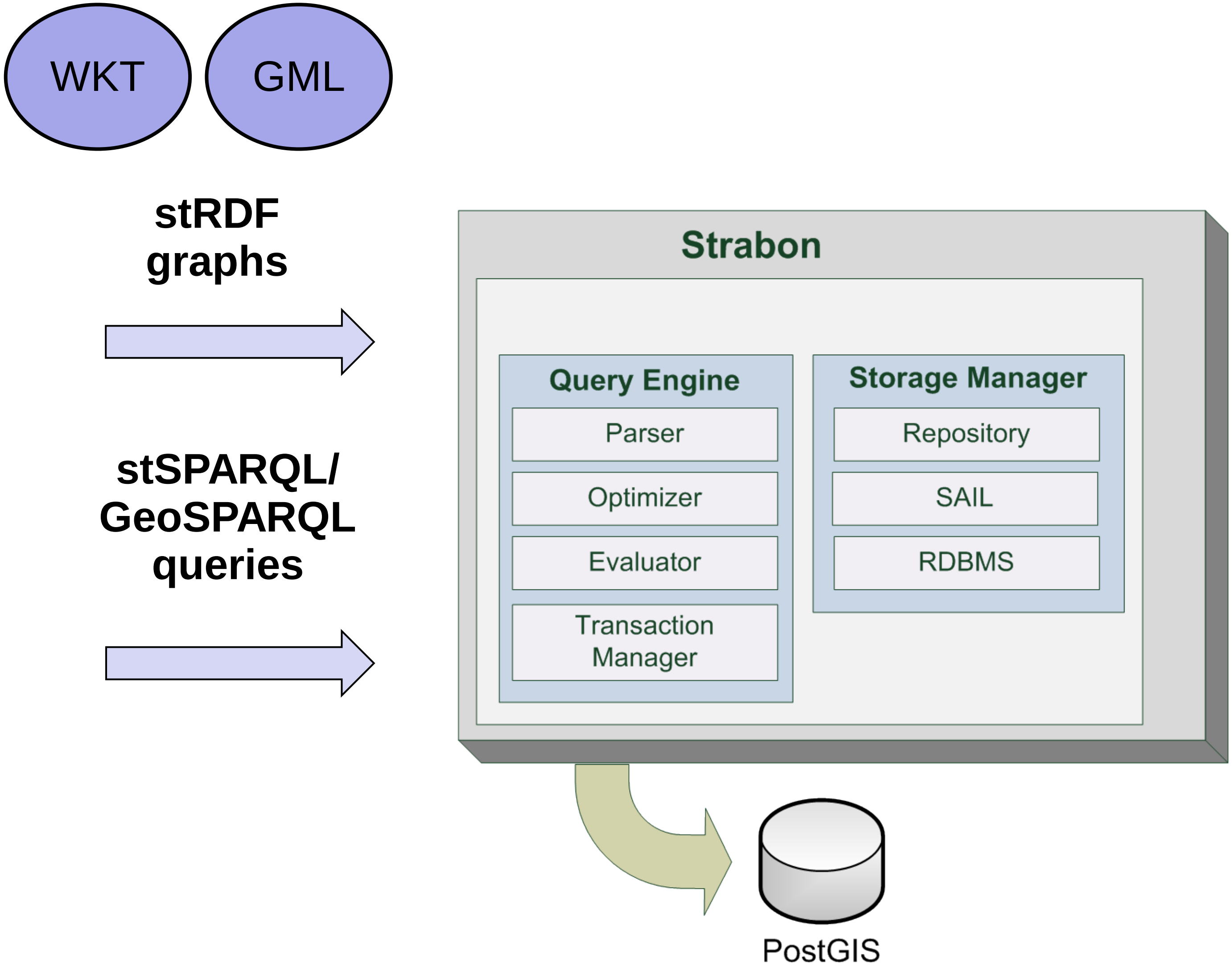
Strabon

- 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, RCC8 families exposed
  - Spatial aggregate functions
- Support for multiple coordinate reference systems

- GeoSPARQL support
  - Core
  - Geometry Extension
  - Geometry Topology Extension



Strabon - Implementation



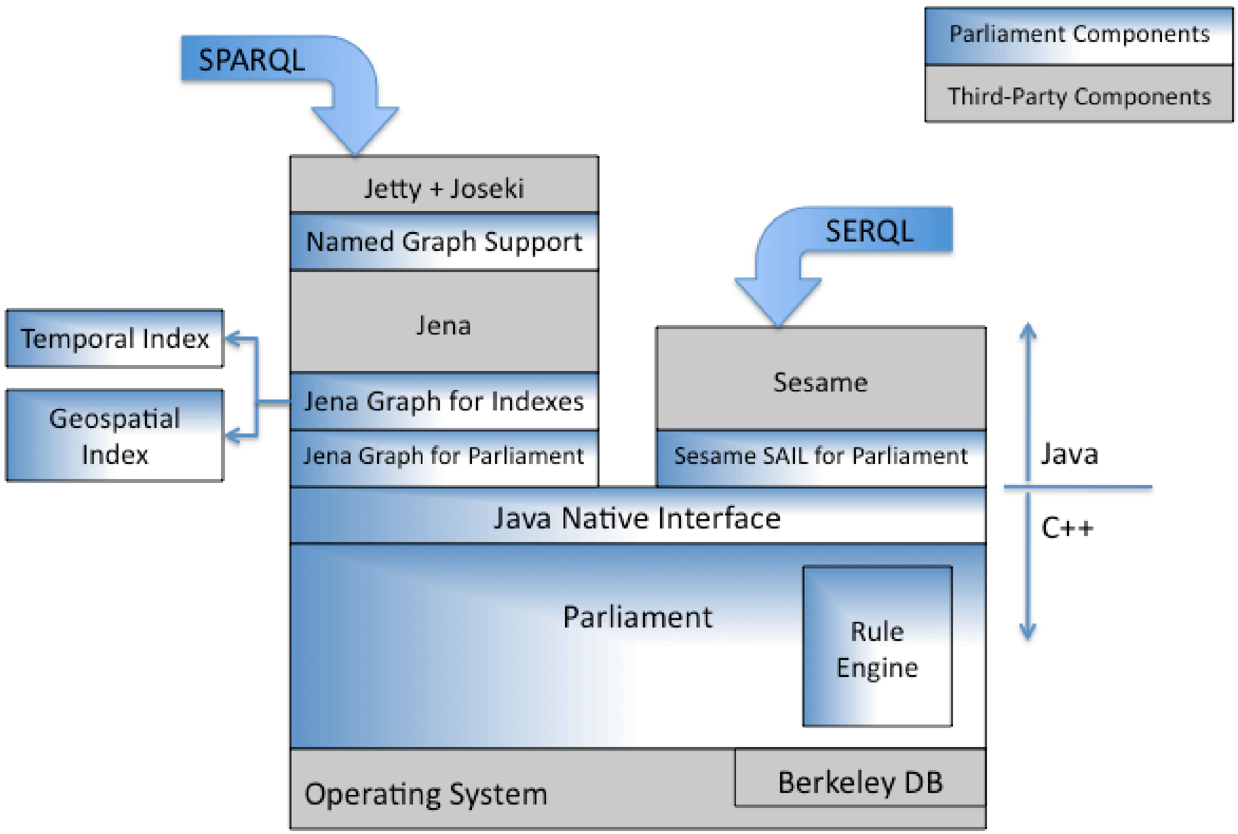
Open Source, available from http://www.strabon.di.uoa.gr/

Parliament

- Storage Engine
- Developed by Raytheon BBN Technologies (Dave Kolas)
- First implementation of GeoSPARQL
  - Geometries represented using typed literals
    - WKT & GML serializations supported
  - Three families of topological functions exposed
    - OGC-SFA
    - Egenhofer
    - RCC8
  - Multiple CRS support

Parliament - Implementation

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



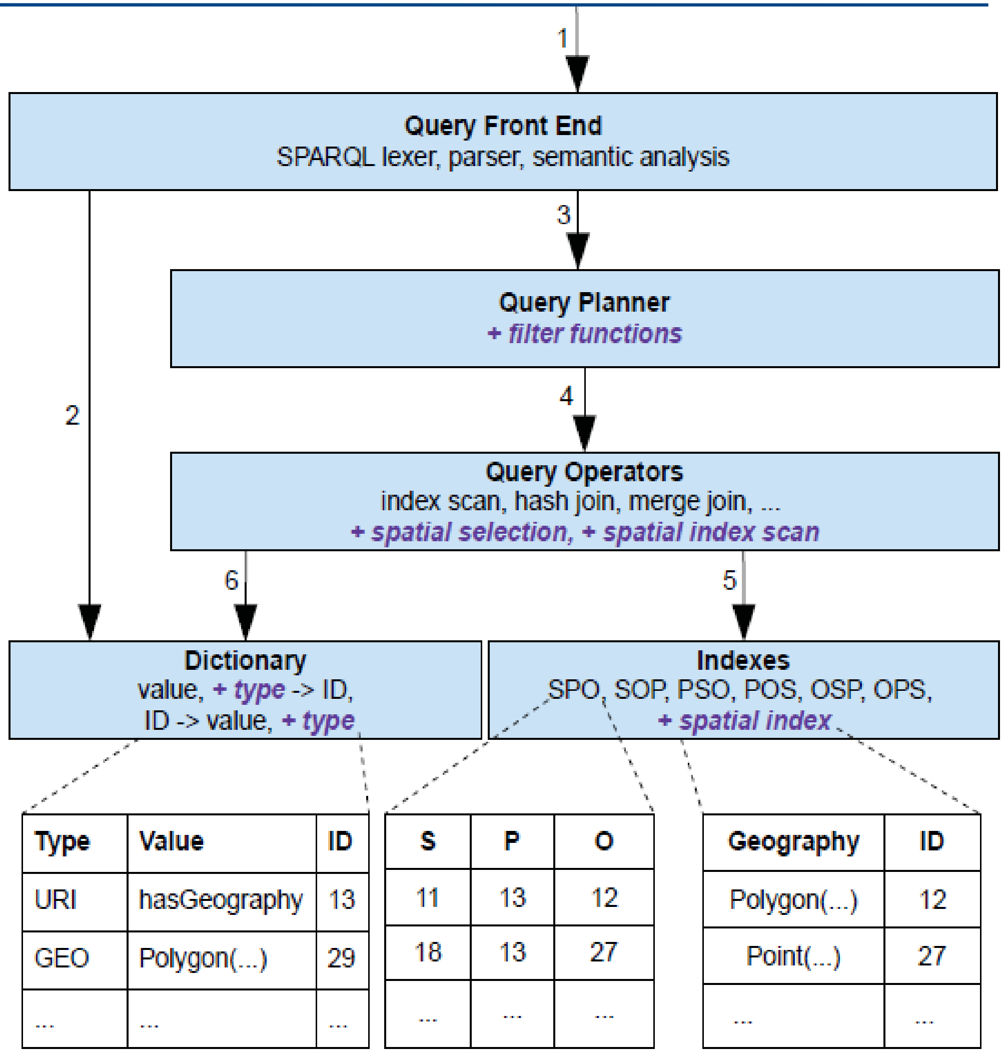
Open Source, available from
http://www.parliament.semwebcentral.org

Brodt et al.

- 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



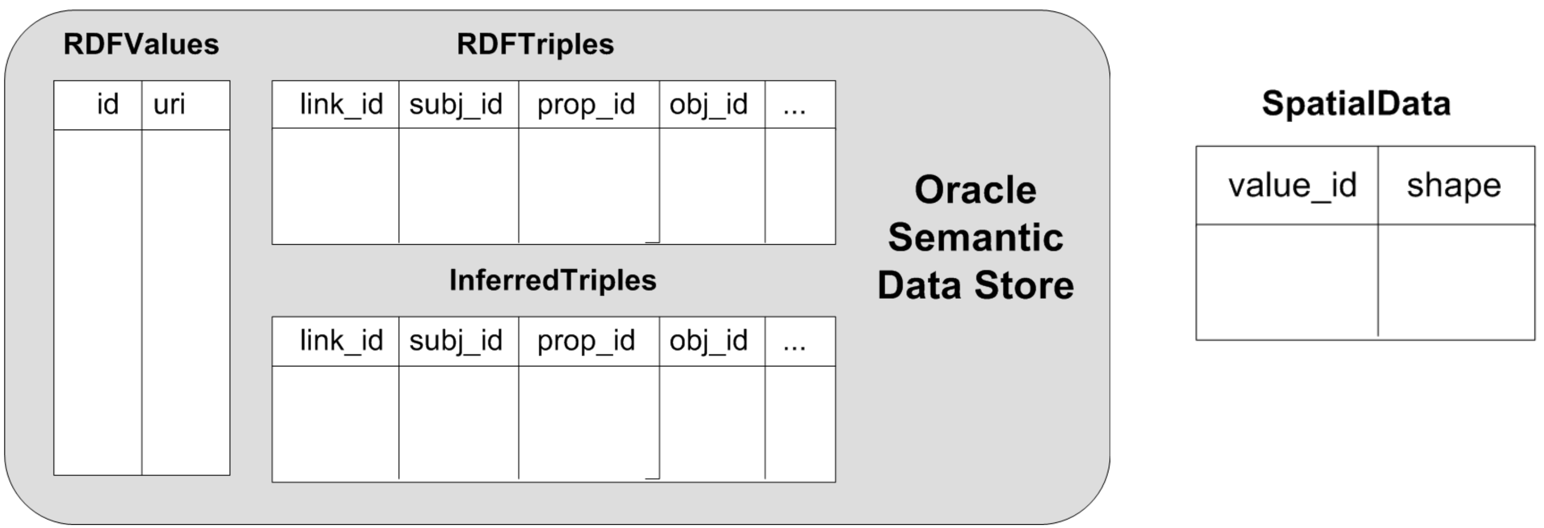
Available upon request

Perry

- 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
  - RCC8 calculus
  - Allen’s interval calculus

Perry

- 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	RCC8
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:** Applications of Linked Geospatial Data

Conclusions

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

- **Next topic:** Applications of Linked Geospatial Data

Applications of Linked Geospatial Data

Presenter: Manos Karpathiotakis

Existing applications of linked geospatial data (1/2)

- Linked Sensor Middleware
  - Utilizing Virtuoso as backend
  - Available at http://lsm.deri.ie
- Ordnance Survey
  - Utilizing an RDF store provided by Talis as backend
  - Available at http://bis.clients.talis.com/
- SemsorGrid4Env
  - Utilizing Strabon as backend
  - Demo available at http://webgis1.geodata.soton.ac.uk/flood.html
- TELEIOS
  - Fire monitoring service performed by the National Observatory of Athens
  - Utilizing Strabon as backend
  - Demo available at http://test.strabon.di.uoa.gr/NOA/
- …

Fire Monitoring Service - Objective

- Design, implement, and validate a **fully automatic fire monitoring processing chain**, for **real time fire monitoring and rapid mapping**, that combines in real-time:

  i) Volumes of Earth Observation image acquisitions.

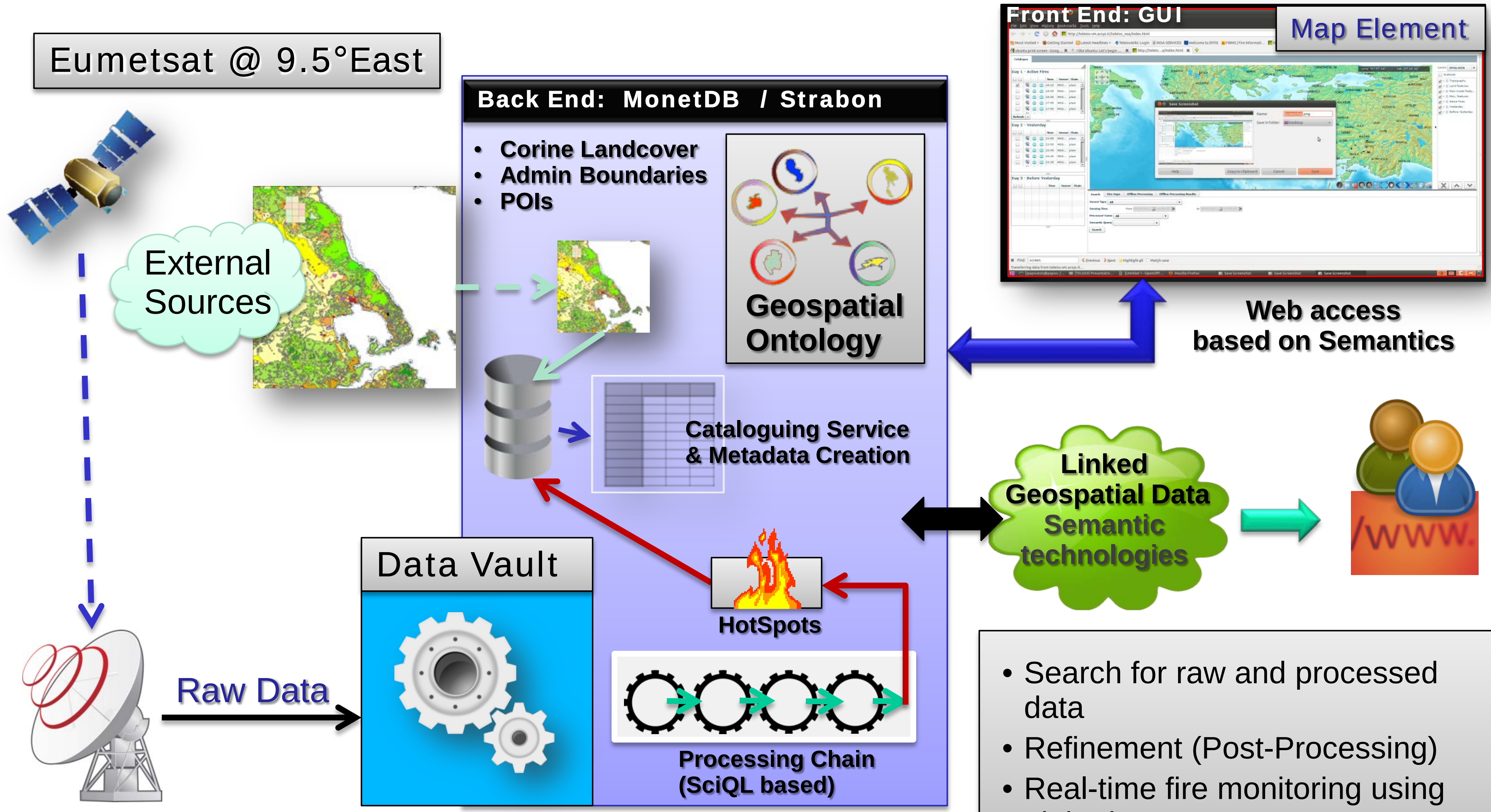
  ii) Volumes of fire monitoring products.

  iii) Models/Algorithms for data exchange and processing.

  iv) Auxiliary geo-information.

  v) Human evidence, in order to draw reliable decisions and generate **highly accurate fire products**.

Fire Monitoring Service

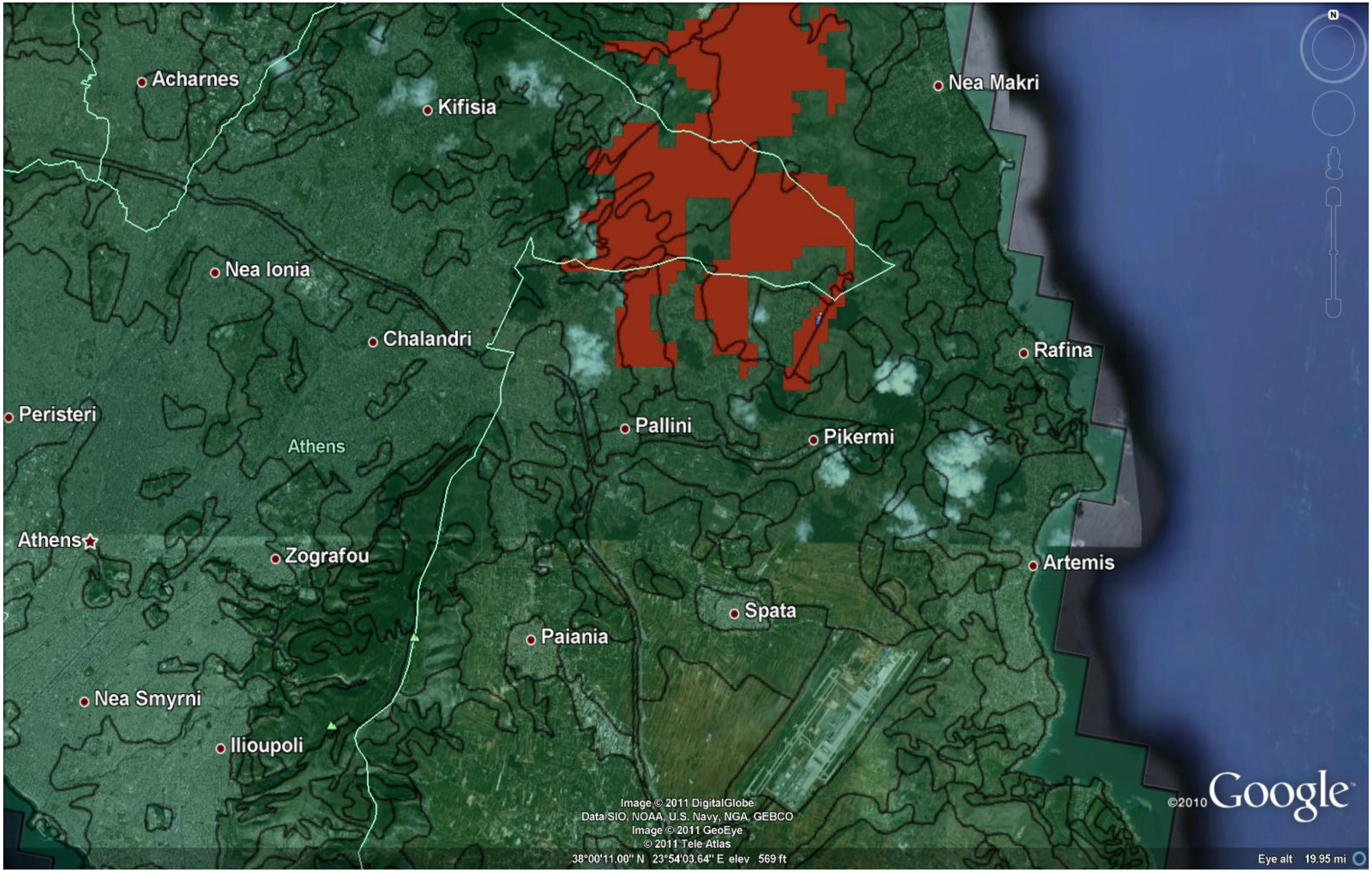


- Search for raw and processed data
- Refinement (Post-Processing)
- Real-time fire monitoring using Linked Data

Requirements of the Fire Monitoring Service

- Need for modeling of
  - Geospatial information
  - Temporal information
  - Product metadata
  - Product content
- Need to link to other data sources
  - GIS data
  - Other information on the Web

Linked Data in the context of TELEIOS

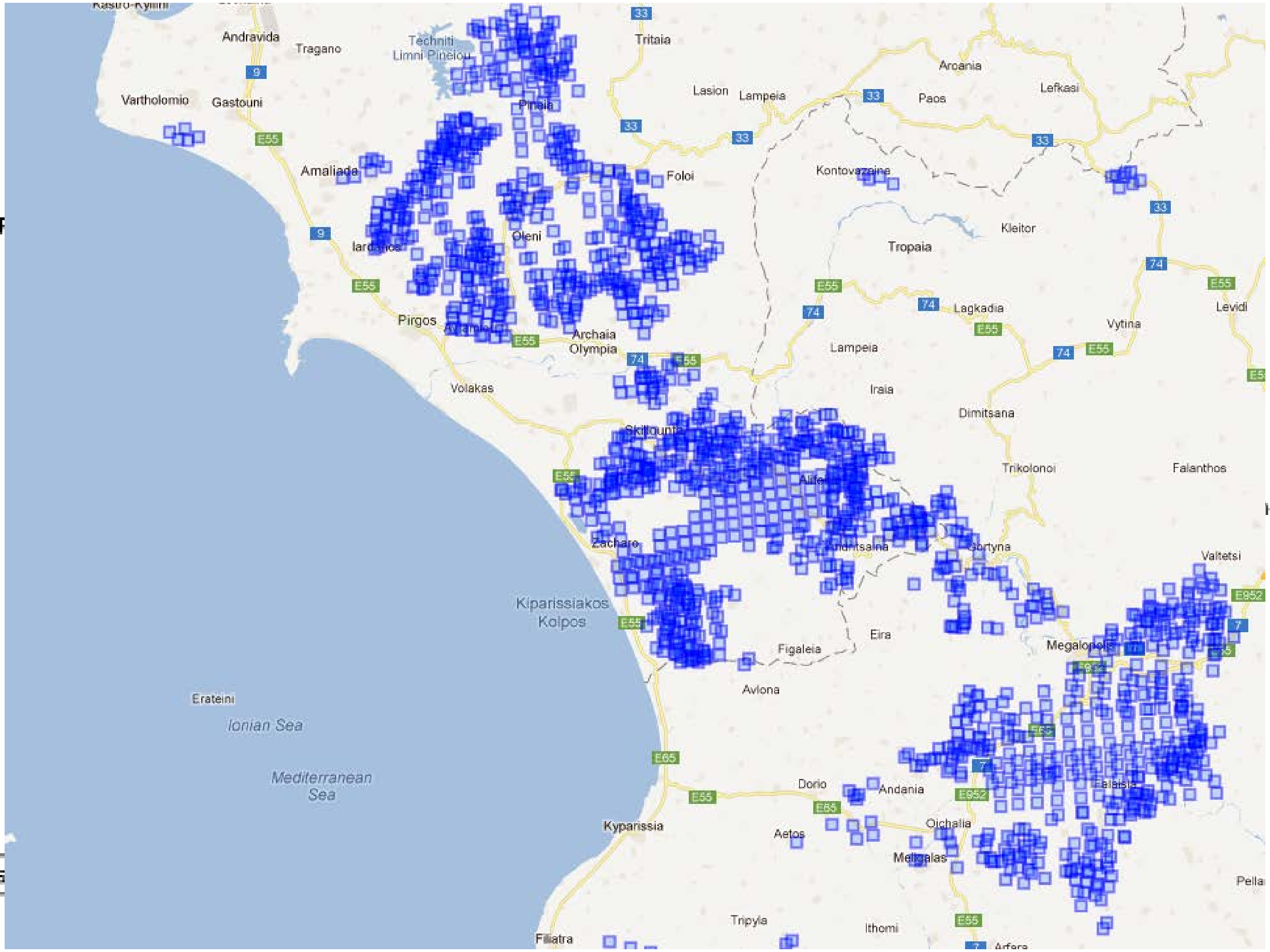


Linked Data used in the Fire Monitoring Service

- Hotspots detected by the National Observatory of Athens (NOA) and other authorities
- Administrative Regions of Greece
- Corine Land Use / Land Cover Nomenclature
- LinkedGeoData
- GeoNames

Linked Open Data (1 / 4)

- Hotspots

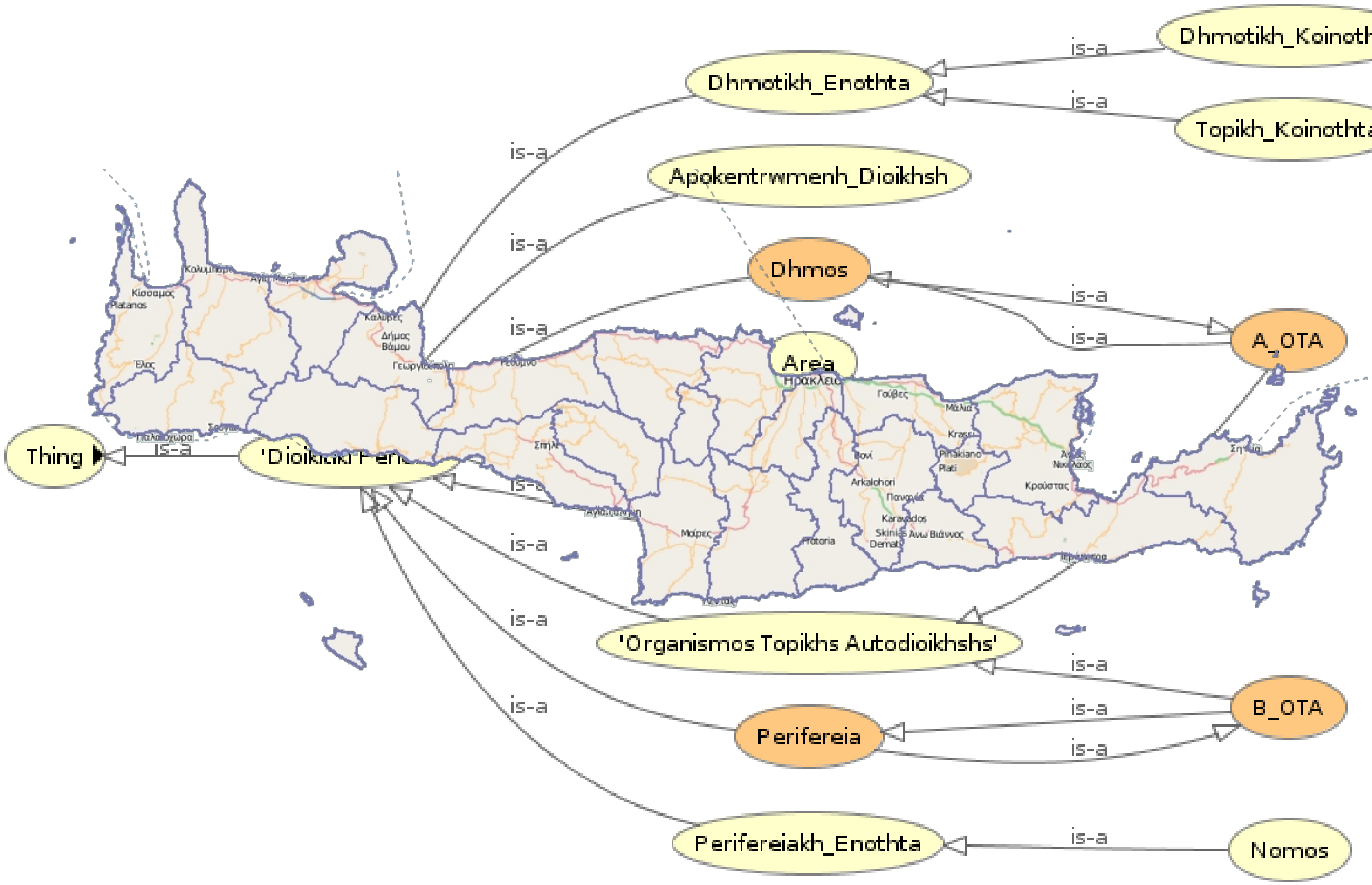


Hotspots

```
noa:Hotspot_15145 rdf:type noa:Hotspot;
                  noa:isDerivedFromSatellite "METEOSAT9"^^xsd:string;
                  noa:isDerivedFromSensor "MSG2"^^xsd:string;
                  noa:hasAcquisitionTime "2007-08-24T14:45:00”
                                         ^^xsd:dateTime;
                  noa:producedFromProcessingChain "StaticThresholds“
                                              ^^xsd:string;
                  noa:hasConfirmation noa:unknown;
                  noa:hasConfidence "0.5"^^xsd:double;
                  noa:hasGeometry
                              "POLYGON((393801.42 4198827.92, ...,
                               393801.42  4198827.92));
                              <http://www.opengis.net/def/crs/EPSG/0/2100>“
                              ^^strdf:WKT.
```

Linked Open Data (2 / 4)

- Greek Administrative Geography

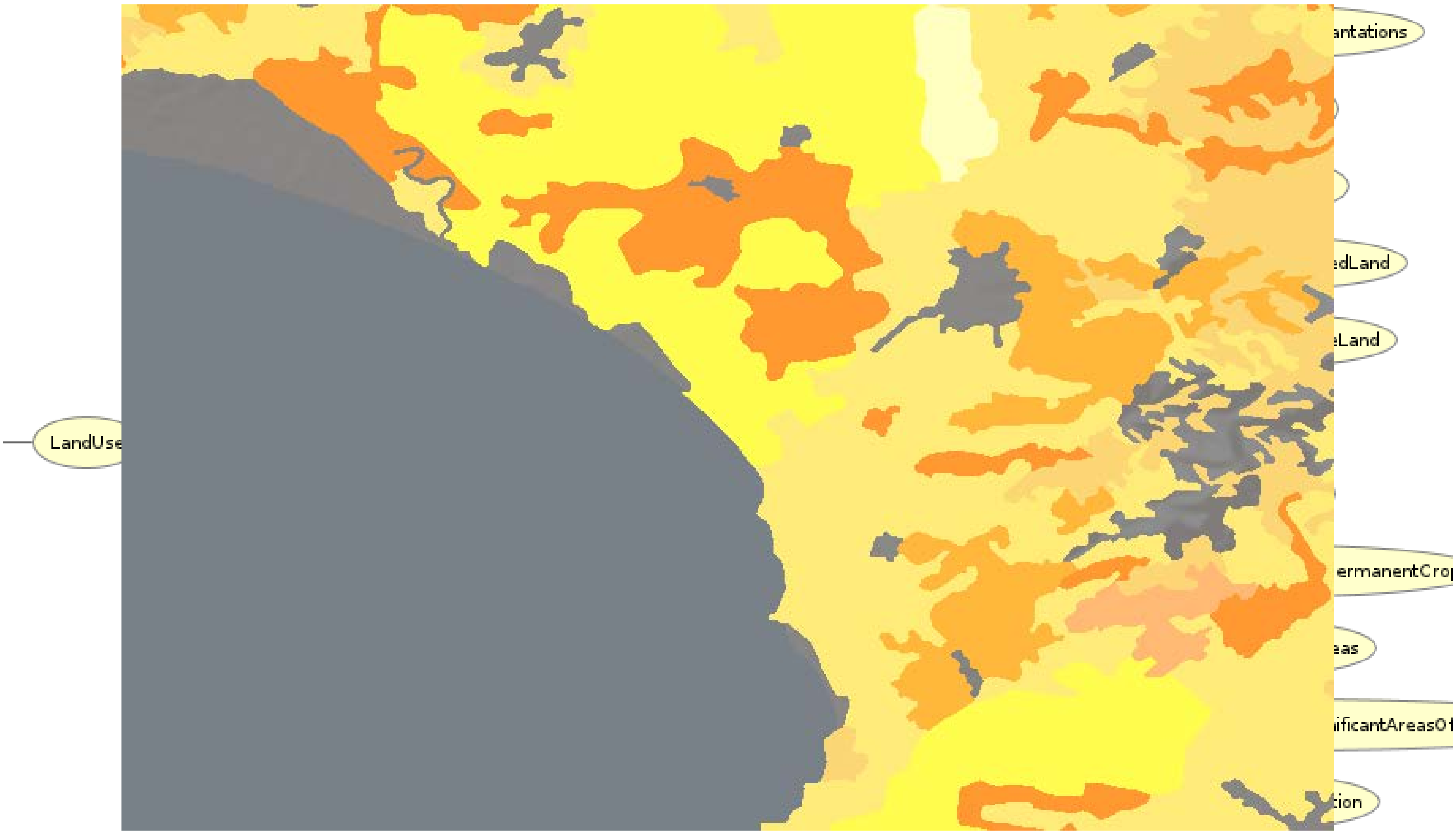


Greek Administrative Geography

```
gag:gag003000009002 rdf:type owl:NamedIndividual ;
                    rdf:type gag:Dhmos;
                    rdfs:label "ΔΗΜΟΣ ΧΕΡΣΟΝΗΣΟΥ"@el;
                    rdfs:label "Hersonissos";
                    noa:hasYpesCode "9309"^^xsd:integer;
                    strdf:hasGeometry "MULTIPOLYGON (((25.37
                                      35.34,...,25.21
                                      35.47)))"^^strdf:WKT;
                    gag:isPartOf gag:gag003000000101.
```

Linked Open Data (3 / 4)

- Corine Land Use / Land Cover

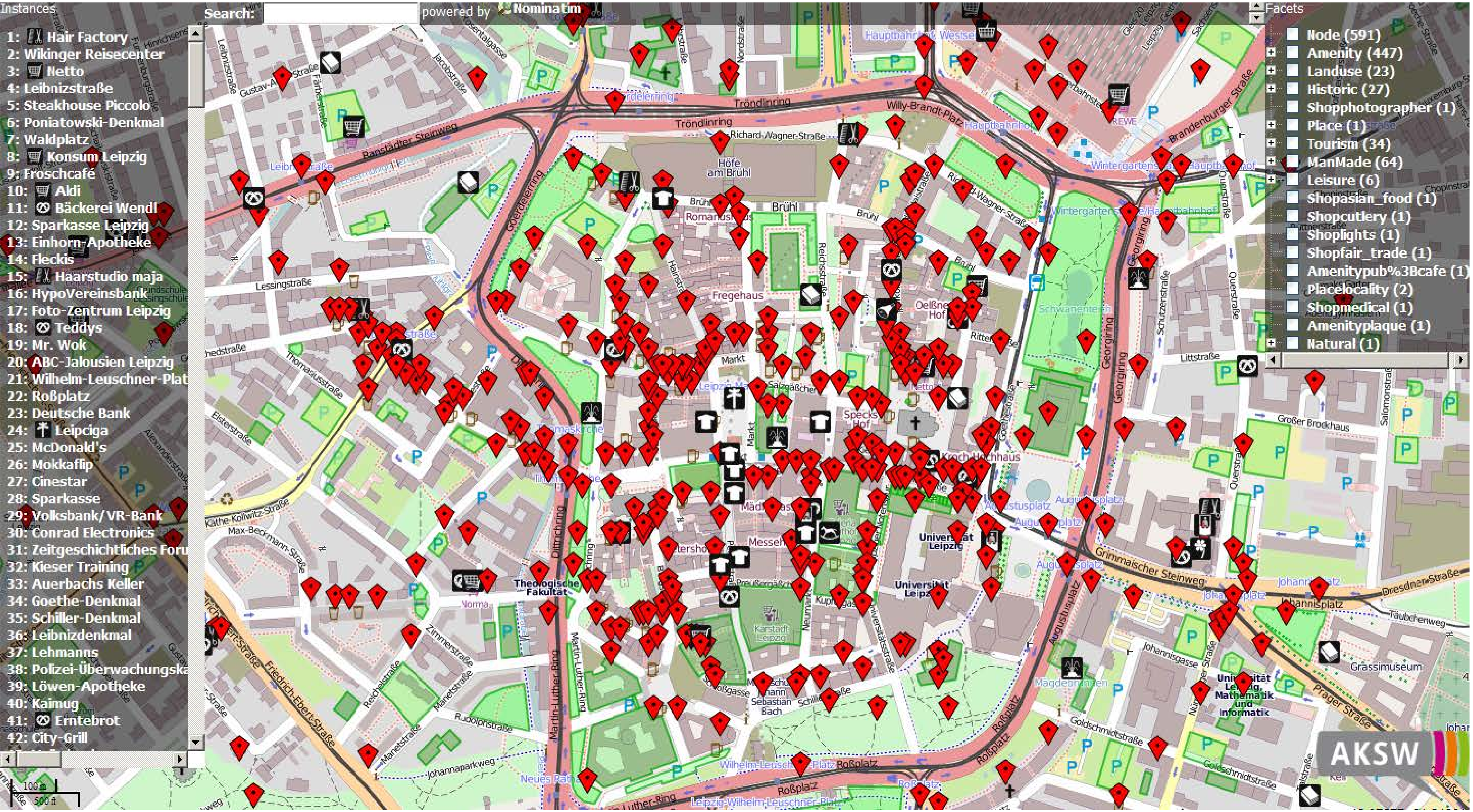


Corine Land Use / Land Cover

```
noa:Area_24015134   rdf:type noa:Area ;
                    noa:hasCode "312"^^xsd:decimal;
                    noa:hasID "EU-203497"^^xsd:string;
                    noa:hasArea_ha "255.580790497"^^xsd:double;
                    noa:hasGeometry "POLYGON((15.53 62.54, ...,
                                    15.53 62.54))"^^strdf:WKT;
                    noa:hasLandUse noa:coniferousForest.
```

Linked Open Data (4/4)

- LinkedGeoData



LinkedGeoData

```
lgd:node741703450 rdf:type lgdo:Node;
                  rdf:type lgdo:Place;
                  rdf:type lgdo:Town;
                  rdfs:label "Зофори"@ru;
                  rdfs:label "Ζωφόροι"@el;
                  rdfs:label "Zofori"@en;
                  lgdo:directType lgdo:Town;
                  wgs84:geometry "POINT(25.2704
                                 35.2061)"^^virtrdf:Geometry;
                  wgs84:lat "35.2060912"^^xsd:double;
                  wgs84:long "25.2703858"^^xsd:double;
                  lgdo:contributor lgd:user153221 .
```

Discovering raw data and products

- Retrieve shapefiles that contain acquisitions taken between 12:00 and 12:30 of August 26, 2007 and acquired by sensor MSG2

```
SELECT ?filename
WHERE {
   ?file rdf:type noa:ShpFile .
   ?file noa:hasFilename ?filename .
   ?file noa:hasAcquisitionTime ?sensingTime .
   FILTER( str(?sensingTime) >= "2007-08-26T12:00:00" ) .
   FILTER( str(?sensingTime) <= "2007-08-26T12:30:00" ) .
   ?file noa:isDerivedFromSensor ?sensor .
   FILTER( str(?sensor) = "MSG2" ) .
   ?file noa:producedFromProcessingChain ?chain .
   FILTER( str(?chain) = "StaticThresholds" ) . }
```

Discovering raw data and products

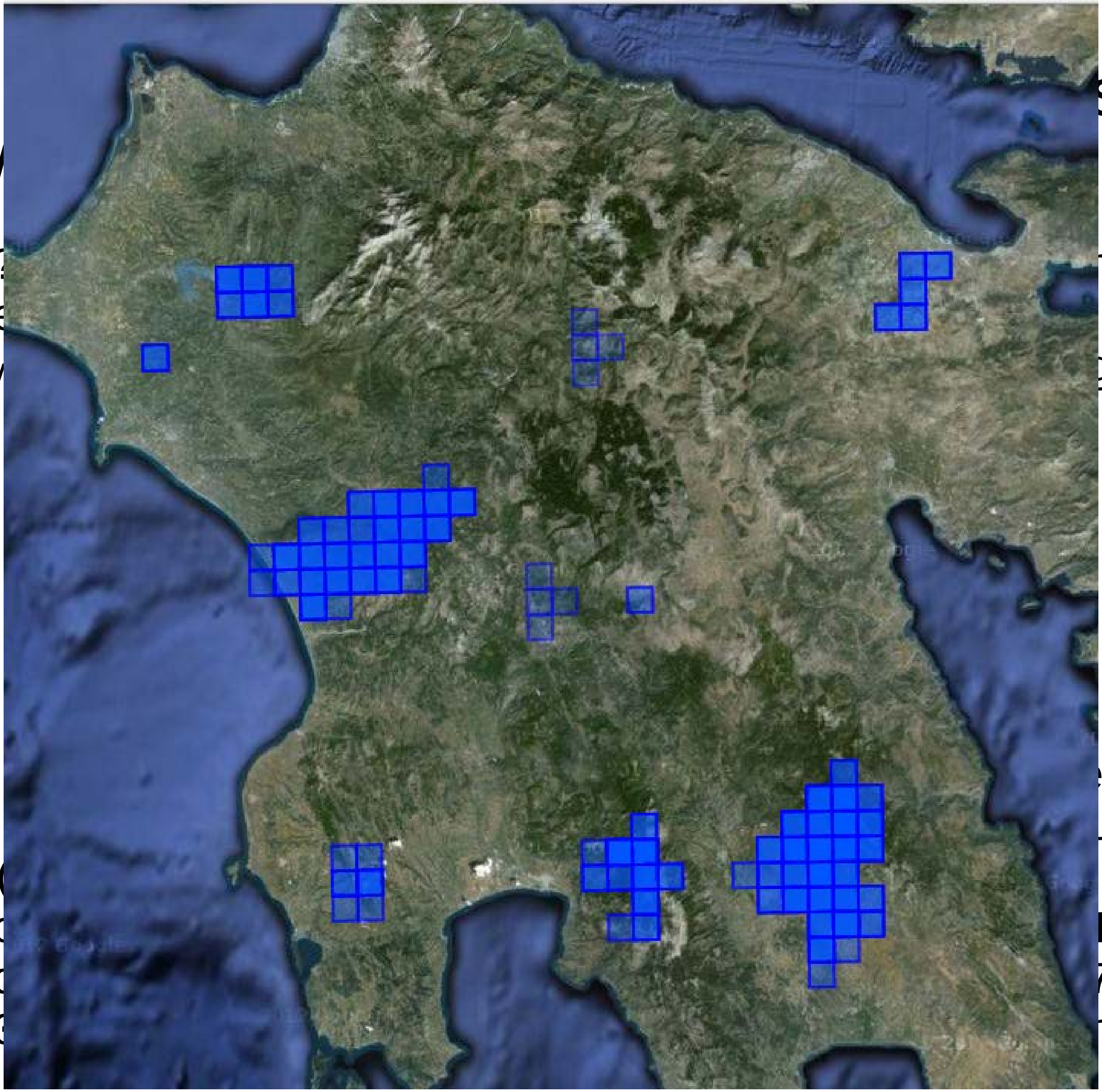
- Retrieve shapefiles that contain acquisitions taken between 12:00 and 12:30 of August 26, 2007 and acquired by sensor MSG2

?filename
MSG2_07-08-26_12:00_StaticThresholds.shp
MSG2_07-08-26_12:15_StaticThresholds.shp
MSG2_07-08-26_12:30_StaticThresholds.shp

Creating a map (1/4)

- Get all … se at 24/08/…

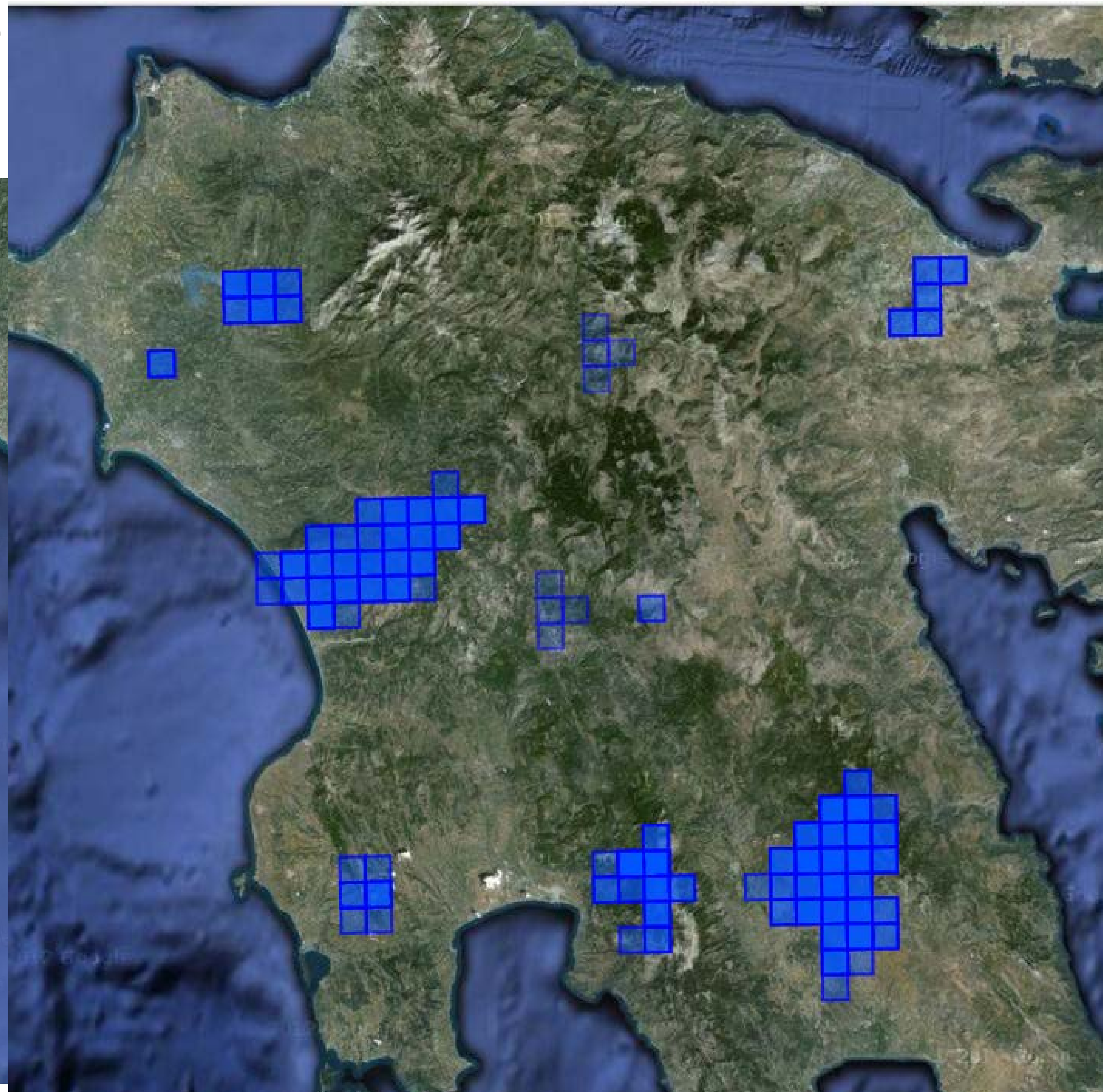
```
SELECT ?h ?…                                   …hSensor
?hSatellite…
<http://www…                                   …geo)
WHERE { ?h …
```



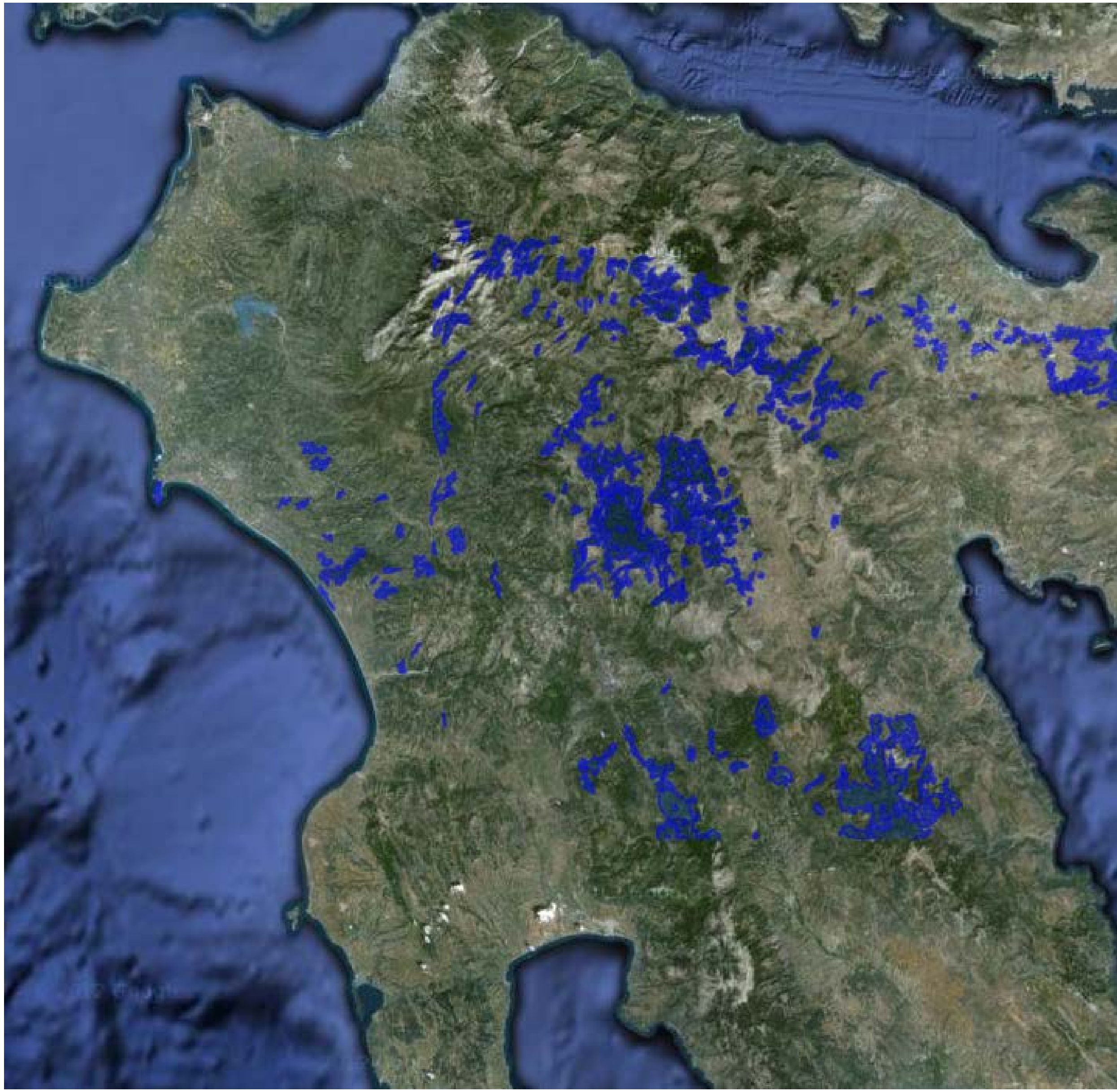
```
                                               … ;
                                               …n .
FILTER(str(…
FILTER(?hA…                                    …Time) .
FILTER(str…                                    …7 38.36, 23.77
36.05, 21.…                                    …hGeo)) .
```

Improve product accuracy

- Delete … e in the sea.



Creating a map (2/4)



- Get all

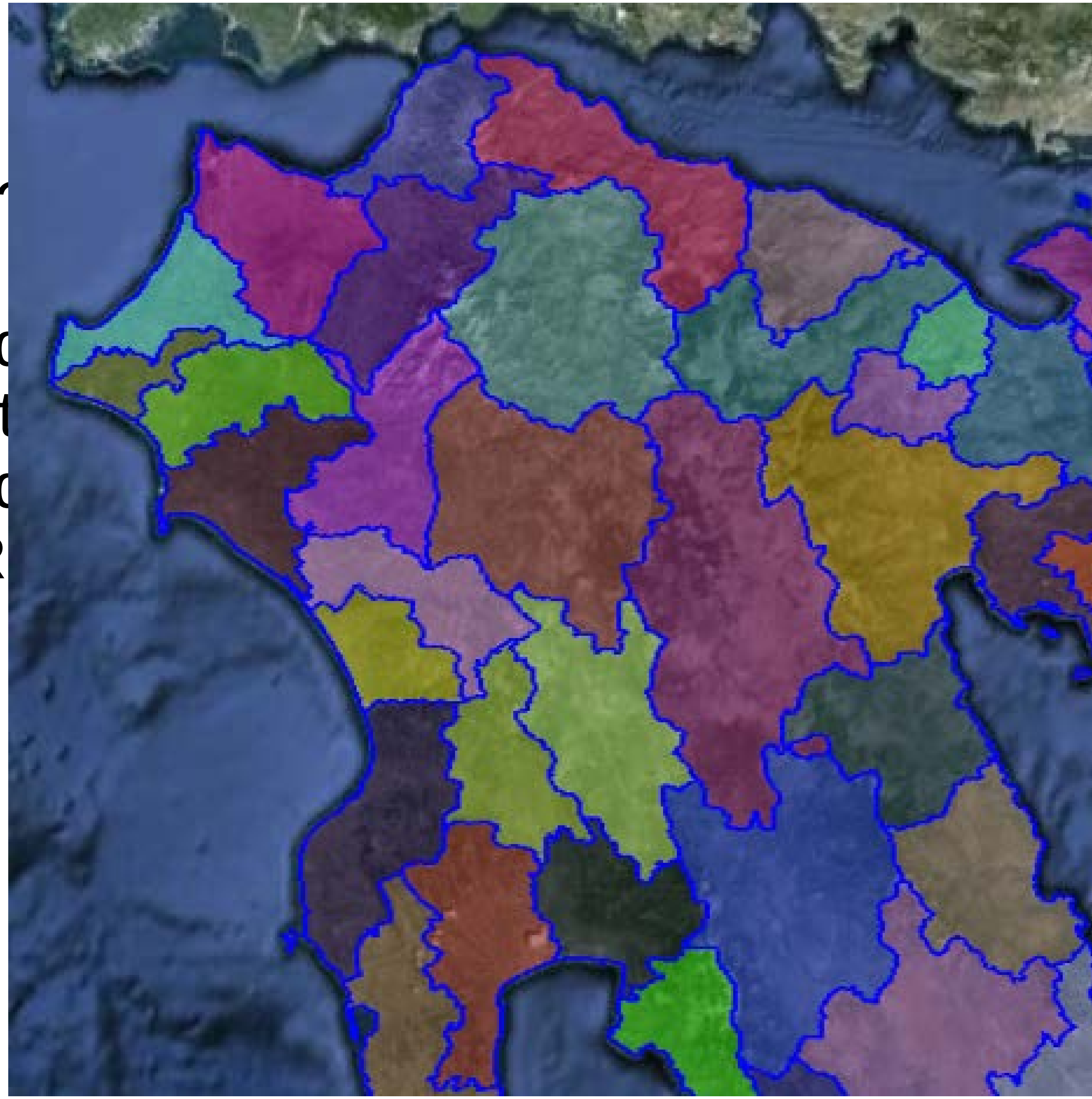
```
SELECT ?
WHERE{ ?

        ?                                  e.
        F

        F                                  (21.027
                                           7 36.05,
                                           36))"
```

Creating a map (3 / 4)

• Get all

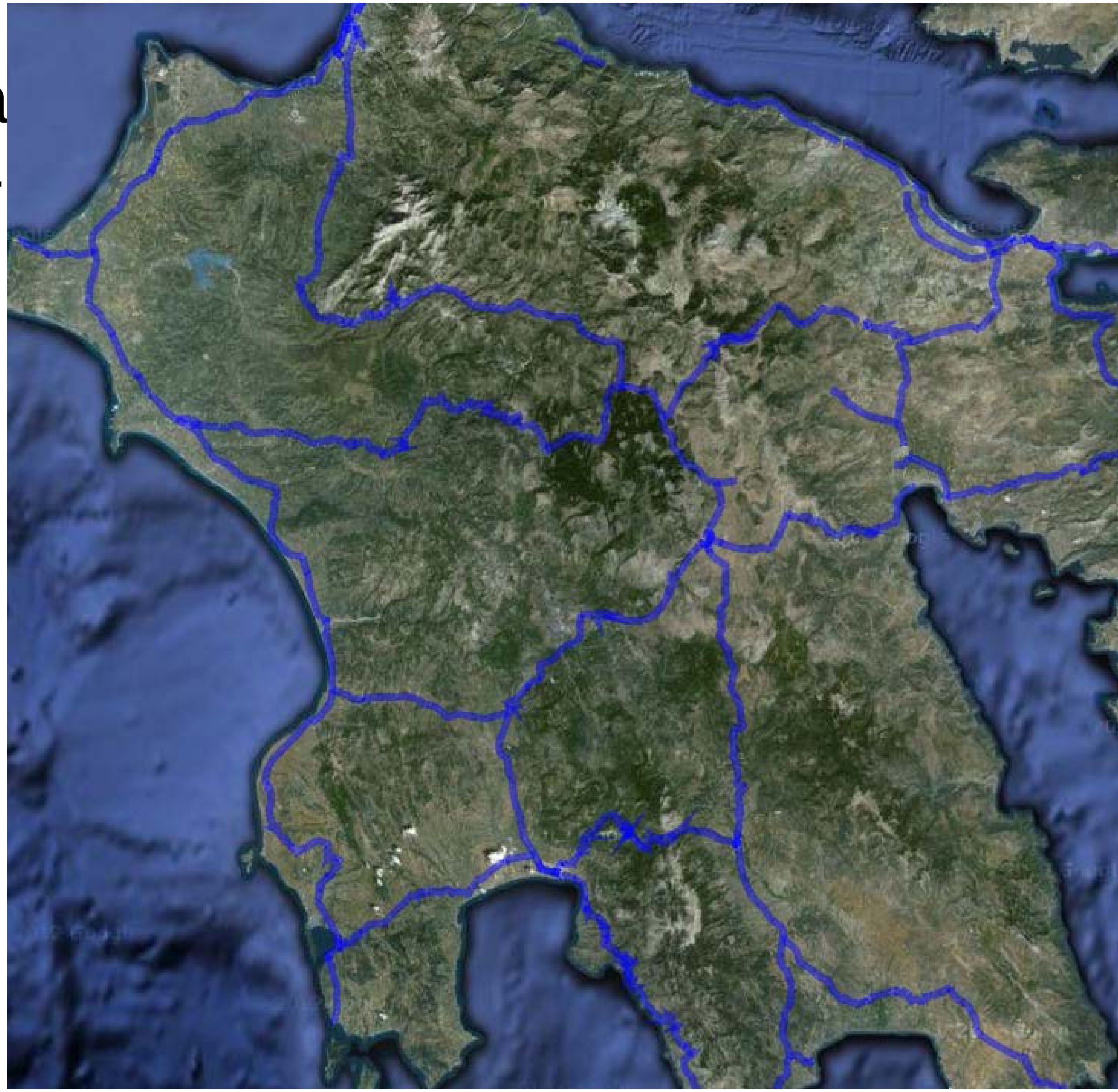


```
SELECT  ?
WHERE {
   ?d   ro
        st
        ro
   FILTER                                027
                                         .05,
                                         "^^
                                         dGeo)).
}
```

Creating a map (4/4)

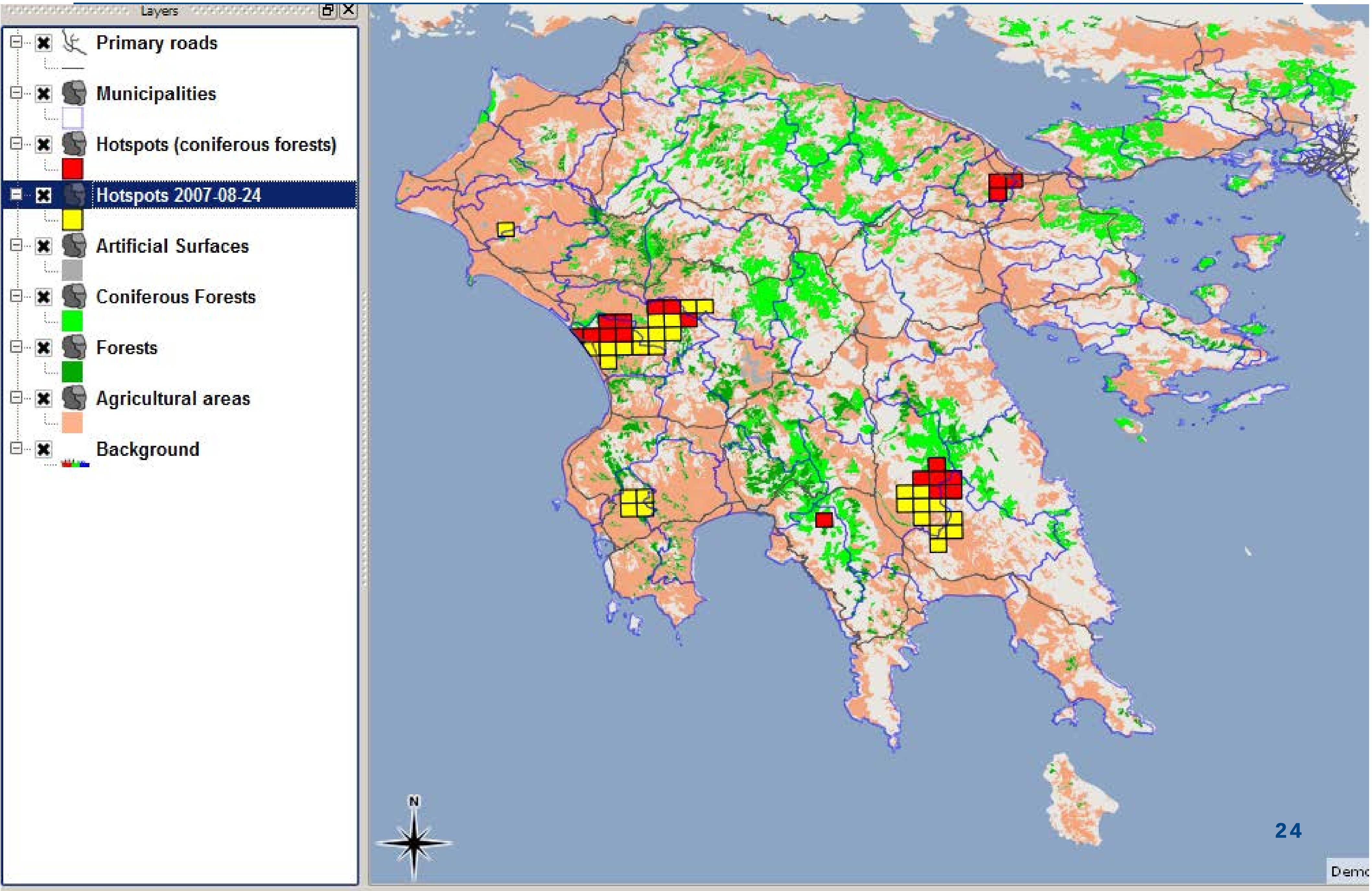
- Get a

```
SELECT
WHERE{
```



```
y) .
GON((
8.36,
6.05,
f:WKT,
}
```

Final map



Layers
Primary roads
Municipalities
Hotspots (coniferous forests)
Hotspots 2007-08-24
Artificial Surfaces
Coniferous Forests
Forests
Agricultural areas
Background
N

Conclusions

- **Applications using Linked Geospatial Data**
  - **Examples of applications**
  - **NOA Hotspot Detection and Fire Monitoring Service**
    - Datasets used
    - Queries leading to final map creation

Conclusions

Presenter: Manolis Koubarakis

What we talked about

- Introduction
- Background in geospatial data modeling
- Geospatial data in the Semantic Web: stSPARQL and GeoSPARQL
- Implemented systems
- Applications

What we did not talk about: Tools

- Tools for **translating** GIS data (e.g., shape files or tables from a geospatial DBMS) into the geospatial extensions of RDF that we presented

What we did not talk about: Reasoning

- **Description logics and ontology languages** for spatial information
  - Theory
  - Reasoners (e.g., RacerPro, PelletSpatial)
  - OWL 2
- Approaches using **rules** (e.g., to do qualitative spatial reasoning).

Invitation

Reasoning Web 2012
Summer School
Vienna, Austria
September 03 - 08 2012
Home Objectives Program Lecturers Application Committee Venue Participants Contact
Lectures
• Data Models and Query Languages for Linked Geospatial Data (Manolis Koubarakis, Manos Karpathiotakis, Kostis Kyzirakos, Babis Nikolaou, Michael Sioutis)
• Semantic Wikis: Approaches, Applications, and Perspectives (François Bry, Sebastian Schaffert, Denny Vrandecic, Klara Weiand)
• OWL 2 Profiles: An Introduction to Lightweight Ontology Languages (Markus Krötzsch)
• Argumentation and the Web (Francesca Toni)
• Federation and Navigation in SPARQL 1.1 (Marcelo Arenas, Jorge Pérez)
• Reasoning with Uncertain and Inconsistent Ontologies on the Semantic Web (Guilin Qi, Jianfeng Du)
• Linked Data Stream Processing (Manfred Hauswirth, Danh Le Phuoc, Josiane Xavier Parreira)
• Datalog and Its Extensions for the Semantic Web (Georg Gottlob, Giorgio Orsi, Andreas Pieris, Mantas Šimkus)
• Reasoning and Query Answering in Description Logics (Magdalena Ortiz, Mantas Šimkus)
• Reasoning and Ontologies in Data Extraction (Sergio Flesca, Tim Furche, Ermelinda Oro)
Platinum Sponsors
Artificial Intelligence
www.elsevier.com/locate/artint
http://www.kr.tuwien.ac.at/events/rw2012/Program.html#CourseGottlobEtal

What we did not talk about: Theory

- **Semantics:** How do we extend the semantics of SPARQL, to give semantics to stSPARQL and GeoSPARQL?
- **Computational complexity of query processing:** What is the complexity of stSPARQL or GeoSPARQL querying?
- Other theoretical issues

Thank you for Attending!

- Questions?
- Feedback?