Geospatial data in RDF – 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

SPAUUK

- 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

[Kolas and Self, 2007]
Early works

SPARQL-ST

- 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.

[Source: Perry, 2008]
# stRDF and stSPARQL

- 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

[Koubarakis and Kyzirakos, 2010]
Example
Example with simplified geometries
Example in stRDF

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

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 .
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 \texttt{strdf:WKT} and \texttt{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 \texttt{strdf:WKT} consist of an optional URI identifying the coordinate reference system used.

  e.g., "\texttt{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
Geonames

Map center: N 48° 12' 21" E 16° 27' 49"

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

Terms of Use

Map | Satellite | Hybrid | Terrain

Data Models and Query Languages for Linked Geospatial Data
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

[Graph showing the ontology with classes and subclasses like Thing, Dhmos, Area, Country, Island, Perifereia, Nomos, Dhmotikh_Enothta, Apokentrwmenh_Dioikhsh, Topikh_Koinothta, and Dhmotikh_Koinothta connected with is-a relationships.]
Greek Administrative Geography

Kallikrates ontology
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

- Thing
  - LandUse
    - ForestsAndSemiNaturalAreas
      - Forest
        - MixedForest
        - ConiferousForest
        - NaturalGrassland
        - TransitionalWoodlandShrub
        - SclerophyllousVegetation
        - MoorsAndHeathland
        - SparselyVegetatedAreas
        - BurntAreas
        - BeachesDunesAndSandPlains
        - BareRock
        - GlaciersAndPerpetualSnow
      - ShrubsAndOrHerbaceousVegetationAssociations
        - OpenSpacesWithLittleOrNoVegetation
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 "MULTI POLYGON(((
  393801.42 4198827.92, ..., 393008 424131)))";
<http://www.opengis.net/def/crs/EPSG/0/2100>"^^strdf:WKT.
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
  \[ \text{xsd:int strdf:Dimension(strdf:geometry A)} \]
  \[ \text{xsd:string strdf:GeometryType(strdf:geometry A)} \]
  \[ \text{xsd:int strdf:SRID(strdf:geometry A)} \]

- Get the desired representation of a geometry
  \[ \text{xsd:string strdf:AsText(strdf:geometry A)} \]
  \[ \text{strdf:wkb strdf:AsBinary(strdf:geometry A)} \]
  \[ \text{xsd:string strdf:AsGML(strdf:geometry A)} \]

- Test whether a certain condition holds
  \[ \text{xsd:boolean strdf:IsEmpty(strdf:geometry A)} \]
  \[ \text{xsd:boolean strdf:IsSimple(strdf:geometry A)} \]
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

  \[
  \text{strdf:geometry} \quad \text{strdf:Boundary}(\text{strdf:geometry A}) \\
  \text{strdf:geometry} \quad \text{strdf:Envelope}(\text{strdf:geometry A}) \\
  \text{strdf:geometry} \quad \text{strdf:Intersection}(\text{strdf:geometry A, strdf:geometry B}) \\
  \text{strdf:geometry} \quad \text{strdf:Union}(\text{strdf:geometry A, strdf:geometry B}) \\
  \text{strdf:geometry} \quad \text{strdf:Difference}(\text{strdf:geometry A, strdf:geometry B}) \\
  \text{strdf:geometry} \quad \text{strdf:SymDifference}(\text{strdf:geometry A, strdf:geometry B}) \\
  \text{strdf:geometry} \quad \text{strdf:Buffer}(\text{strdf:geometry A, xsd:double distance})
  \]

- Spatial metric functions

  \[
  \text{xsd:float} \quad \text{strdf:distance}(\text{strdf:geometry A, strdf:geometry B}) \\
  \text{xsd:float} \quad \text{strdf:area}(\text{strdf:geometry A})
  \]

- Spatial aggregate functions

  \[
  \text{strdf:geometry} \quad \text{strdf:Union}(\text{set of strdf:geometry A}) \\
  \text{strdf:geometry} \quad \text{strdf:Intersection}(\text{set of strdf:geometry A}) \\
  \text{strdf:geometry} \quad \text{strdf:Extent}(\text{set of strdf:geometry A})
  \]
StSPARQL: Geospatial SPARQL 1.1

Select clause
- Construction of new geometries (e.g., \texttt{strdf:buffer(?geo, 0.1)}
- Spatial aggregate functions (e.g., \texttt{strdf:union(?geo)})
- Metric functions (e.g., \texttt{strdf:area(?geo)})

Filter clause
- Functions for testing topological spatial relationships between spatial terms (e.g., \texttt{strdf:contains(?G1, strdf:union(?G2, ?G3))})
- Numeric expressions involving spatial metric functions (e.g., \texttt{strdf:area(?G1) \leq 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., \texttt{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))
}
```
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.
  ?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
Isolate the parts of the burnt areas that lie in coniferous forests.

```sparql
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 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:** Geospatial data in RDF - GeoSPARQL
Bibliography

[Kolas and Self, 2007]

[Perry, 2008]

[Koubarakis and Kyzirakos, 2010]
Geospatial data in RDF – GeoSPARQL

Presenter: Kostis Kyzirakos
GeoSPARQL

GeoSPARQL is a recently completed OGC standard

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 data type

Spatial literal
Example in GeoSPARQL (2/2)

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


Asserted topological relation
GeoSPARQL Components

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

Core

- Topology Vocabulary Extension
  - relation family

- Geometry Extension
  - serialization
  - version

- Geometry Topology Extension
  - serialization
  - version
  - relation family

- Query Rewrite Extension
  - serialization
  - version
  - relation family

- RDFS Entailment Extension
  - serialization
  - version
  - relation family
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.

```sparql
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**
  

- **Spatial metric functions**
  
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

```turtle
@prefix gag: <http://example.com/gag/> .
@prefix geonames: <http://www.geonames.org/> .

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".

```

**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.
}
```
GeoSPARQL: An example

Find the municipality that contains the community of Ancient Olympia

```sparql
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
  - 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

\[
\text{ex:f1 geo:hasGeometry ex:g1.}
\]

\[
\text{geo:hasGeometry rdfs:domain geo:Feature.}
\]

we can infer the following triples:

\[
\text{ex:f1 rdf:type geo:Feature .}
\]

\[
\text{ex:f1 rdf:type geo:SpatialObject .}
\]
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

  \[
  \text{geof:sfWithin(AthensWKT, GreeceWKT)}
  \]

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

  \[
  \text{dbpedia:Athens geo:sfWithin dbpedia:Greece}
  \]

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

For all \( f_1 \), \( f_2 \), \( g_1 \), \( g_2 \), \( g_1\text{Serial} \), \( g_2\text{Serial} \)

\( (f_1[\text{geo:sfWithin} \rightarrow f_2] : - \)

Or (

- \( f_1[\text{geo:defaultGeometry} \rightarrow g_1] \)
- \( f_2[\text{geo:defaultGeometry} \rightarrow g_2] \)
- \( g_1[\text{ogc:asGeomLiteral} \rightarrow g_1\text{Serial}] \)
- \( g_2[\text{ogc:asGeomLiteral} \rightarrow g_2\text{Serial}] \)

\( \text{External} (\text{geo:sfWithin} (g_1\text{Serial}, g_2\text{Serial})) \)

- \( f_1[\text{geo:defaultGeometry} \rightarrow g_1] \)
- \( g_1[\text{ogc:asGeomLiteral} \rightarrow g_1\text{Serial}] \)
- \( f_2[\text{ogc:asGeomLiteral} \rightarrow g_2\text{Serial}] \)

\( \text{External} (\text{geo:sfWithin} (g_1\text{Serial}, g_2\text{Serial})) \)

- \( f_2[\text{geo:defaultGeometry} \rightarrow g_2] \)
- \( f_1[\text{ogc:asGeomLiteral} \rightarrow g_1\text{Serial}] \)
- \( g_2[\text{ogc:asGeomLiteral} \rightarrow g_2\text{Serial}] \)

\( \text{External} (\text{geo:sfWithin} (g_1\text{Serial}, g_2\text{Serial})) \)

- \( f_1[\text{ogc:asGeomLiteral} \rightarrow g_1\text{Serial}] \)
- \( f_2[\text{ogc:asGeomLiteral} \rightarrow g_2\text{Serial}] \)

\( \text{External} (\text{geo:sfWithin} (g_1\text{Serial}, g_2\text{Serial})) \)
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 } }
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]
Implemented RDF Stores with Geospatial Support

Presenter: Kostis Kyzirakos
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, RCC-8 families exposed
  - Spatial aggregate functions
- Support for multiple coordinate reference systems

- GeoSPARQL support
  - Core
  - Geometry Extension
  - Geometry Topology Extension
Strabon - Implementation

- **WKT**
- **GML**

stRDF graphs

stSPARQL/GeoSPARQL queries

Strabon

- **Query Engine**
  - Parser
  - Optimizer
  - Evaluator
  - Transaction Manager

- **Storage Manager**
  - Repository
  - SAIL
  - RDBMS

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

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., 2010]
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
  - 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
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


Does not include the spatial capabilities extension
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](https://dev.opensahara.com/projects/useekm)
<table>
<thead>
<tr>
<th>System</th>
<th>Language</th>
<th>Index</th>
<th>Geometries</th>
<th>CRS support</th>
<th>Comments on Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strabon</td>
<td>stSPARQL/</td>
<td>R-tree-over-GiST</td>
<td>WKT / GML support</td>
<td>Yes</td>
<td>• OGC-SFA</td>
</tr>
<tr>
<td></td>
<td>GeoSPARQL*</td>
<td></td>
<td></td>
<td></td>
<td>• Egenhofer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• RCC-8</td>
</tr>
<tr>
<td>Parliament</td>
<td>GeoSPARQL</td>
<td>R-Tree</td>
<td>WKT / GML support</td>
<td>Yes</td>
<td>• OGC-SFA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Egenhofer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• RCC-8</td>
</tr>
<tr>
<td>Brodt et al.</td>
<td>SPARQL</td>
<td>R-Tree</td>
<td>WKT support</td>
<td>No</td>
<td>OGC-SFA</td>
</tr>
<tr>
<td>(RDF-3X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perry</td>
<td>SPARQL-ST</td>
<td>R-Tree</td>
<td>GeoRSS GML</td>
<td>Yes</td>
<td>RCC-8</td>
</tr>
<tr>
<td>AllegroGraph</td>
<td>Extended SPARQL</td>
<td>Distribution</td>
<td>2D point geometries</td>
<td>Partial</td>
<td>• Buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sweeping technique</td>
<td></td>
<td></td>
<td>• Bounding Box</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Distance</td>
</tr>
<tr>
<td>OWLIM</td>
<td>Extended SPARQL</td>
<td>Custom</td>
<td>2D point geometries</td>
<td>No</td>
<td>• Point-in-polygon</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(W3C Basic Geo Vocabulary)</td>
<td></td>
<td>• Buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Distance</td>
</tr>
<tr>
<td>Virtuoso</td>
<td>SPARQL</td>
<td>R-Tree</td>
<td>2D point geometries</td>
<td>Yes</td>
<td>SQL/MM (subset)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(in WKT)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uSeekM</td>
<td>SPARQL</td>
<td>R-tree-over-GiST</td>
<td>WKT support</td>
<td>No</td>
<td>OGC-SFA</td>
</tr>
</tbody>
</table>
Conclusions

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

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

[Kyzirakos et al, 2010]

[Kyzirakos et al, 2012]

[Battle and Kolas, 2011]
Bibliography

[Brodt et al, 2010]

[Perry, 2007]