A geospatial extension of RDF and an implementation on top of an RDBMS

Kostis Kyzirakos

MaDgIK Lab
Dept. of Informatics & Telecommunications
University of Athens
Joint Work with …

- Prof. Manolis Koubarakis
- Manos Karpathiotakis
Outline

1. Introduction
2. The Data Model stRDF
3. The Query Language stSPARQL
4. The System Strabon
5. The Semantic Registry
6. Experimental evaluation
7. Future Work
The vision of the **Semantic Sensor Web**: annotate **sensor data** and services to enable discovery, integration, interoperability etc.

Sensor annotations involve **thematic, spatial and temporal metadata**.

How about using RDF?
RDF: Resource Description Framework

- W3C recommendation
- RDF is graphical formalism (+ XML syntax + semantics)
  - For representing metadata
  - For describing the semantics of information in a machine-accessible way
  - Resources are described in terms of properties and property values using RDF statements
  - Statements are represented as triples, consisting of a subject, predicate and object. [S, P, O]
Motivation (cont’d)

How about using RDF?

Good idea. But **RDF can represent only thematic metadata** properly. We want to take into account spatial and temporal information to aid sensor publication and discovery.

What can we do about spatial and temporal metadata?

- Answer: Extend RDF to represent spatial and temporal metadata.
Our Approach

- Resource metadata is modeled using stRDF, an extension of RDF, that can be used to represent thematic, spatial and temporal metadata.
- Resource metadata are queried using stSPARQL, an extension to SPARQL for querying stRDF data.
- Design and implement a storage and query evaluation module for stRDF/stSPARQL.
- Design and implement a semantic registry that stores metadata about SSW resources which is used for resource discovery using thematic, spatial and temporal criteria.
Outline

1. Introduction
2. **The Data Model stRDF**
3. The Query Language stSPARQL
4. The System Strabon
5. The Semantic Registry
6. Experimental evaluation
7. Future Work
From RDF to sRDF - Example

ex:sensor1 rdf:type ex:Sensor ;
    ex:measures ex:Temperature ;
    ex:hasLocation ex:location1 .

ex:location1 strdf:hasSpatialExtent
    "POINT(37.94194 23.63722)
    <http://srid.org/ref/epsg/4326/>"
    ^ ^ ogc:WKT .

New Kind of Typed Literals
Spatial Metadata - Example

POLYGON((6 1, 10 1, 10 10, 6 6, 6 1))
1. Introduction
2. The Data Model stRDF
3. The Query Language stSPARQL
4. The System Strabon
5. The Semantic Registry
6. Experimental evaluation
7. Future Work
Example – Dataset

- Collection of sensor metadata
Example – Dataset

- Metadata about a deployed sensor:

```rdfs
ex:sensor1 rdf:type ex:Sensor ;
   ex:measures ex:Temperature ;
   ex:hasLocation ex:location1 .

ex:location1 strdf:hasSpatialExtent
   "POINT(37.94194 23.63722)
    <http://srid.org/ref/epsg/4326/>"
    ^^^ogc:WKT .
```
Metadata about geographical areas:

```xml
<ex:area1 rdf:type ex:UrbanArea ;
    ex:hasName "Athens" ;
    strdf:hasSpatialExtent
"POLYGON((37.74194 23.53722
    … … … … 37.74194 23.53722))
<http://srid.org/ref/epsg/4326/>
"^^ogc:WKT
```
Spatial selection. Find the URIs of the sensors that are inside the rectangle R(37, 23, 38, 24).

```
select ?S
where {?S rdf:type ex:Sensor ;
    ssn:haslocation ?L .
    ?L strdf:hasSpatialExtent ?GEO .
    filter(?GEO inside "POLYGON((37 23,37 24,38 24, 38 23,37 23))<http://srid.org/ref/epsg/4326/>"^^ogc:WKT) }
```
Answer

?S

ex:sensor1
**Example – Queries**

- **Spatial join.** Find the URIs of the sensors that are located inside an urban Area.

```sql
select ?S
where {{?S rdf:type ssn:Sensor ;
          ssn:haslocation ?L .

    ?UA rdf:type ex:UrbanArea ;
          strdf:hasSpatialExtent ?UAGEO .

    filter(?GEO inside ?UAGEO) }
```
Answer

?S

ex:sensor1
What is new in stSPARQL syntax?

- **k-ary spatial terms**
  - quantifier-free formulas (constants)
  - spatial variables
  - projections of *k-ary spatial terms*
  - the result of **set operations** on *k-ary spatial terms*: intersection, union, difference
  - the result of **geometric operations** on *k-ary spatial terms*: boundary, buffer, minimum bounding box

- Metric spatial terms
  - \( \text{VOL, AREA, LEN, MAX, MIN} \)

- **Select clause**: construction of new spatial terms
  - intersection, union, difference, projection of spatial terms

- **Where clause**: Quad patterns to refer to the valid time of a triple

- **Filter clause**:
  - **Spatial predicates** (topological): disjoint, touch, equals, inside, covered by, contains, covers, overlap
  - **Temporal predicates**: before, equal, meets, overlaps, during, starts, finishes
1. Introduction
2. The Data Model stRDF
3. The Query Language stSPARQL
4. The System Strabon
5. The Semantic Registry
6. Experimental evaluation
7. Future Work
The System Strabon

Strabon

Sesame

Query Engine
- Parser
- Optimizer
- Evaluator
- Transaction Manager

Storage Manager
- Repository
- SAIL
- RDBMS

Constraint Engine
- Representation Translator
- Convex Partitioner

PostGIS
Storing stRDF Data

Geometric objects represented with Well Known Text (WKT)

Storage Manager
- Repository
- SAIL
- RDBMS

Constraint Engine
- Representation Translator
  - Input processor → cddlib → Output processor
- Convex Partitioner
  - Input processor → CGAL → Output processor

PostGIS

points, lines, polygons, TINs and polyhedrons
Evaluating stSPARQL Queries

Query Engine
- Parser
- Optimizer
- Evaluator
- Transaction Manager

Constraint Engine
- Representation Translator
- Convex Partitioner

PostGIS

Results (constraints)
Results (WKT)
Results (KML)

stSPARQL query
Spatial selection. Find the URIs of the sensors that are inside the rectangle \( R(37, 23, 38, 24) \).

```sparql
select ?S
where { ?S rdf:type ex:Sensor ;
    ssn:hasLocation ?L .
    ?L strdf:hasSpatialExtent ?GEO .
    filter(?GEO inside "POLYGON((37 23,37 24,38 24, 38 23,37 23))<http://srid.org/ref/epsg/4326/>"^^ogc:WKT) }
```
Evaluating stSPARQL Queries (2/5)

SQL Query

```
SELECT u_S.value
FROM type_2 t0
INNER JOIN haslocation_6 h1 ON (h1.subj=t0.subj)
INNER JOIN hasspatialextent_8 h2 ON (h2.subj=h1.obj)
INNER JOIN geo_values l_GEO ON (l_GEO.id=h2.obj)
LEFT JOIN uri_values u_S ON (u_S.id=t0.subj)
WHERE t0.obj=3
AND (l_GEO.strdfgeo @ ST_GeomFromText('POLYGON ((37 23, 37 24, 38 24, 38 23, 37 23))',4326))
```
Evaluating stSPARQL Queries (3/5)
Evaluating stSPARQL Queries (4/5)
Evaluating stSPARQL Queries (5/5)
1. Introduction
2. The Data Model stRDF
3. The Query Language stSPARQL
4. The System Strabon
5. The Semantic Registry
6. Experimental evaluation
7. Future Work
[SensorGrid4Env] A service oriented architecture for a Semantic Sensor Grid

- **Application Tier**: Services that provide domain specific functionality to consumers & applications.
- **Middleware Tier**: A service that can be accessed by prospective clients who want to manage thematic and spatial metadata.
- **Data Tier**: Services that provide access to streaming and stored data sources.

- Applications
- Application Services
- Semantic Integrator
- Semantic Registry
- Data Source
- Connectivity Bridge
- Concrete Resource
The Semantic Registry stores semantic annotations of available resources:

- sensors
- sensor networks
- data sources
- web services.

Annotations are registered by service providers. Clients discover resources by posing queries.
Interfaces Implemented

- **Application Tier**
  - Application Services
  - Semantic Integrator
  - Semantic Registry
  - Data Source
  - Connectivity Bridge
  - Concrete Resource

- **Middleware Tier**
  - Strabon
    - Service
    - Registration
    - Query
    - Constraint Engine
    - Storage Manager

- **Data Tier**
  - PostGIS
  - Enables consumers to make well-informed and well-formed interactions with the service
  - Enables a metadata producer to register an RDF(S) document about SSW resources
  - Allows a metadata consumer to query the contents of the registry in order to discover relevant resources using stSPARQL

- Application services enable consumers to make well-informed and well-formed interactions with the service.
- Strabon enables a metadata producer to register an RDF(S) document about SSW resources.
- PostGIS allows a metadata consumer to query the contents of the registry in order to discover relevant resources using stSPARQL.
**Example Orchestration**

**Consumer/Producer:** Register the description of an integrated data source that associates wave height with the ship departure times

**Producer:** Register the description of an SDS service that exports the scheduled ship departure times from England

**Producer:** Register the description of an SDS service that observes wind speed and wave height in Southampton

**Consumer:** Discover a Stored Data Service that observes tide height or wave height in Southeast England

---

**Semantic Registry**

- Service
- Registration
- Query

**Strabon**

- Query Engine
- Constraint Engine
- Storage Manager

**PostGIS**
1. Introduction
2. The Data Model stRDF
3. The Query Language stSPARQL
4. The System Strabon
5. The Semantic Registry
6. Experimental evaluation
7. Future Work
Experimental evaluation

- We used synthetic datasets based on OpenStreetMaps
  - Dataset size: 10 million triples (2GB) - 1 billion triples (110GB)
  - Triples with spatial literals: 1.4 million - 140 million triples
- We measured the response time of queries with various thematic and spatial selectivities
Sample data

- osm:Tag
  - rdf:type
    - osm:Tag1
      - osm:hasTag
        - osm:key
          - "1"
        - osm:value
          - "1"
      - osm:hasTag
        - osm:Node23
          - geordf:hasGeometry1
            - "POINT(2.00 3.00)"
      - osm:id
        - "14"
      - rdf:type
        - osm:Node
          - rdf:type
            - osm:Node23
              - osm:hasTag
                - osm:Tag1
                  - osm:hasTag
                    - osm:Tag
LGD dataset – 10 million triples

small grid, warm caches

response time [sec] vs. number of Nodes in query region

key = 1
key = 2
key = 4
key = 16
key = 32
key = 64
key = 128
key = 256
key = 512
key = 1024
LGD dataset – 1 billion triples

small grid, warm caches

response time [sec]

number of Nodes in query region

key = 1
key = 2
key = 4
key = 16
key = 32
key = 64
key = 128
key = 256
key = 512
key = 1024
Outline

1. Introduction
2. The Data Model stRDF
3. The Query Language stSPARQL
4. The System Strabon
5. The Semantic Registry
6. Experimental evaluation
7. Future Work
Future Work

- Evaluate our current implementation.
- Port our current implementation to MonetDB.
- Study the query processing of stRDF/stSPARQL in the MonetDB case.
- Use our implementation to publish public spatial datasets as Linked Open Data.
Thank you for your attention!

Ερωτήσεις;