TOWARDS TREATING GIS AS VIRTUAL RDF GRAPHS

Emanuele Della Valle[a,b], Hafiz Muhammad Qasim[a] and Irene Celino[b]

[a] Dipartimento di Elettronica e Informazione, Politecnico di Milano, via Ponzio 34/5, Milano, Italy
emanuele.dellavalle@polimi.it, hafizqasim@gmail.com
[b] CEFRIEL, via Fucini 2, 20133 Milano – irene.celino@cefriel.it

KEY WORDS: GIS, SPARQL, Semantic Web

ABSTRACT:

In this paper, we report our experience in addressing practical computational issues influencing the use of Geographic Information Systems and Geo-spatial data from the standpoint of Semantic Web. We discuss the need for treating GIS as Virtual RDF Graphs. In particular, we present a declarative mapping language as well as a prototypical system (namely G2R) that allows SPARQL query involving spatial computation to be executed in a mixed environment where a GIS and a RDF repository are orchestrated. The approach is evaluated by comparing G2R against two similar solution provided by Virtuoso e AllegroGraph.

1. INTRODUCTION

An increasing number of open data sets is becoming available on the Web. Linked Data (LD) plays a central role thanks to projects such as W3C Linked Open Data (LOD) community project[1] that are fostering LD best practice adoption. As of November 2009, 13.1 billions of triples[2] [1] have been published in the LOD cloud.
The most notable example is DBpedia [2] that publishes as LD structured information extracted from the Wikipedia infoboxes[3], but the complete list includes information about scientific publications, geographic locations, people, companies, books, movies, music, television, radio programmes, and online communities.
Note that part this datasets describe real world entities such as monuments, companies, artists, actors, people. This kind of data has a spatial dimension; however the Semantic Web community has devoted very limited attention to exploit this dimension. LinkedGeodata [3] is one of the earliest successful attempts. It exposes as LD 320 million nodes and 25 million ways (June 2009).contained in Open Street Map[4] and it offers simple Web APIs to look for point of interest in a given radius from a geographic position. This feature is also supported by two Semantic Web frameworks: Virtuoso[5] and Allegrograph[6]. However, none of the available framework supports the rich features normally available in a GIS.
In this paper, we report our experience in extending SPARQL (the Semantic Web query language) with typical GIS features by treating a GIS as Virtual RDF Graphs [9-12,15]. In particular, we present a declarative mapping language as well as a prototypical system (namely GIS-to-RDF or simply G2R) that allows SPARQL query involving spatial computation to be executed in a mixed environment where a GIS and a standard SPARQL endpoint are orchestrated.
The rest of the paper is organized as follows: Section 2 provides a minimal background to understand the paper; Section 3 presents a running example that we used throughout the paper; Section 4 provides an architectural overview of G2R; Section 5 describes G2R mapping language; Section 6 shows how G2R rewrites SPARQL queries into SQL statements that use spatial methods; Section 7 briefly reports on our implementation experience, Section 8 provides a comparative evaluation; and, finally, Section 9 concludes.

2. BACK GROUND WORK

In this section, first we briefly present three Semantic Web basic building blocks: RDF, RDF-S and SPARQL. Secondly we discuss approaches that enable treating non-RDF databases as Virtual RDF graphs. Then, we discuss state of the art in adding a spatial dimension to the Semantic Web.

2.1 Semantic Web Building Blocks

2.1.1 RDF and RDF-S

RDF [4] – Resource Description Framework – is the data model, standardize by W3C in the context of the Semantic Web activities, for representing Semantic Web resources. It expresses information as *graphs* consisting of *triples* with subject, property and object. All of them can be identified using IRIs. RDF allows describing a generic resource (the subject of the triples) by stating that a predicate (the property) assumes a given value (the object).
Consider, for instance, the *graph* in Table 1. It describes "St Mark's Clocktower" in Venice. It is taken from DBpedia. The resource `http://dbpedia.org/page/St_Mark%27s_Clocktower` (at line 6) is stated (at line 7) to belong to the category "Visitor attractions in Venice" using the property `skos:subject` [5].
In order to explain the rest of the RDF graph in Table 1, we need to introduce[7] the notion of vocabulary and RDF-S [6] – RDF Schema – as one of the language available for describing vocabularies in the Semantic Web.

---

1 http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/

2 Triples are the minimal building block of Semantic Web, see Section 2 for further explanation.

3 http://en.wikipedia.org/wiki/Wikipedia%3AInfobox

4 Open Street Map (http://www.openstreetmap.org) is a popular Web site for collaborative building an open map of the world.

5 http://www.openlinksw.com/weblog/oerling/?id=1587

6 http://www.franz.com/agraph/support/documentation/current/sparql-geo.html

7 Readers are not supposed to know RDF-S (or more expressive language) for the purpose of this paper; therefore we limit the description to a minimum.

```
1.  @prefix dbpedia:   <http://dbpedia.org/resource/> .
2.  @prefix dbcat:     <http://dbpedia.org/resource/Category:> .
3.  @prefix geo:       <http://www.w3.org/2003/01/geo/wgs84_pos#> .
4.  @prefix rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
5.  @prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .

6.  http://dbpedia.org/resource/St_Mark%27s_Clocktower
7.    skos:subject dbcat: Visitor_attractions_in_Venice;
8.    rdfs:label "St Mark's Clocktower"@en ;
9.    geo:long "12.338912"^^xsd:float ;
10.   geo:lat  "45.434710"^^xsd:float .
```

Table 1 - Part of the RDF graph that described the "Basilica of Sant'Ambrogio" in DBpedia.

A vocabulary is a collection of properties (such as the `skos:subject` property that we exemplified above) and concepts for which the semantics is defined using an ontological language. RDF-S, as light ontological language, allows defining classes, and properties. For instance, RDF-S is used to model the W3C vocabulary for geo positioning [7]. Among other things, this vocabulary introduces the properties `geo:lat` and `geo:long` that respectively represent latitude and longitude using WGS84 as a reference datum. Use of these two properties is shown at line 9 and 10 in Table 1.
RDF-S also offers a small vocabulary to describe resources. e.g., `rdf:label` which is largely used to attach a textual label to a resource; see line 8 in Table 1.

2.1.2 SPARQL

SPARQL [8] – Simple Protocol And RDF Query Language – is the language, proposed by W3C, for querying RDF data published on the Web. SPARQL offers a syntactically SQL-like language for querying RDF graphs.
A SPARQL query is composed by five parts (see Figure 1): zero or more prefix declarations, a query result clause, zero or more `FROM` or `FROM NAMED` clauses, a `WHERE` clause and zero or more query modifiers.

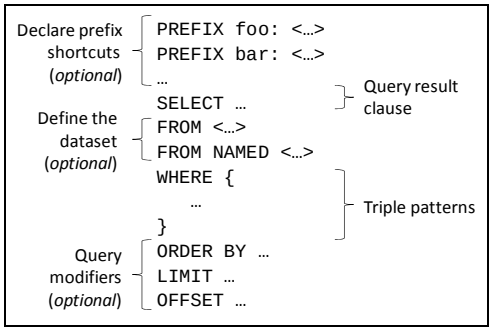

Figure 1 The anatomy of a SPARQL Query

The optional `PREFIX` declarations introduce shortcuts for long IRIs as normally done when working with XML namespaces. Such prefixes can be used in the `WHERE` clause.
The query result clause specifies one of the four form of the results: `SELECT`, `ASK`, `CONSTRUCT` and `DESCRIBE`. `SELECT` queries, the only one we treat in this paper, provide answers in a tabular form as if the SPARQL query were a SQL query executed against a relational database. The optional set of `FROM` or `FROM NAMED` clauses define the dataset against which the query is executed.
The `WHERE` clause is the core of a SPARQL query. It is specified in terms of a set of triple patterns. As extensively explained in the following sections, these triple patterns are used to select the triples composing the result.
Finally, the set of optional query modifiers operate over the triples selected by the `WHERE` clause, before generating the result. As in SQL, the clause `ORDER BY` orders the results set, the `LIMIT` and `OFFSET` allow getting results in chunks.

2.2 Treating non-RDF Database as Virtual RDF Graphs

As Semantic Web technologies are getting mature, there is a growing need for RDF applications to access the content of non-RDF, legacy databases without having to replicate the whole database into RDF.
Since 2003, several solutions have been conceived and implemented [9-12,15]. Conceptually, they are very similar. They provide (in slightly different ways) declarative languages to describe mappings between relational database schemata and RDF-S vocabularies (or more expressive ontological languages). Once the mapping is ready, they can use it to rewrite SPARQL query in SQL. We refer interest readers to [9] and [10] for a comprehensive explanation of the mapping languages of D2RQ and Virtuoso and of the query rewriting algorithms.

2.3 State-of-the-Art in Spatial Dimension of Semantic Web

As we report in Section 1, the Semantic Web community has devoted very limited attention to the spatial dimension of Linked Data. In this section we briefly present two successful, but limited, attempts in this direction: Virtuoso and Allegrograph.

2.3.1 Virtuoso

Virtuoso is a hybrid middleware and database engine that among many other features supports RDF and SPARQL. In particular, Virtuoso includes a spatial extension to SPARQL inspired by SQL MM spatial specification [13]. Spatial indexing is supported by a two dimensional R-tree implementation. The geometries supported are limited to two dimensions, with a choice of WGS 84 latitude and longitude coordinates with haversine distances or a flat 2 dimensional plane for spatial reference system.
Virtuoso supports few spatial data types (i.e., points and geometry) and provides some basic functions for spatial processing compatible with SQL/MM spatial standard [13], i.e., `st_contains`, `st_distance`, `st_intersects`, and `st_within`. The proposed SPARQL extension is SPARQL compliant as it use extensible value testing (see Section of [8]). For instance, the SPARQL query in Table 2 counts the objects of each class occurring within 100 km of <0, 52>, a point near London.

```
1.  PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
2.  SELECT  ?class COUNT (*)
3.  WHERE  {
4.    ?m  geo:geometry  ?geo .
5.    ?m  rdf:type ?class  .
6.    FILTER ( bif:st_intersects( ?geo, bif:st_point(0, 52), 100))
7.  } GROUP BY  ?class
```

Table 2 – An example of spatial enhanced SPARQL query supported by Virtuoso.

2.3.2 AllegroGraph

AllegroGraph is a Semantic Web application framework. Among other features, AllegroGraph supports an encoded data-type for geospatial coordinates, and extends SPARQL to support for geospatial queries.
It introduces a special syntax to match variables by the means of: (a) `RADIUS`, taking a cartesian point and a numeric radius, or (b) `BOUNDINGBOX`, taking two points, or (c) `HAVERSINE`, taking a spherical point and a radius. Moreover, it defines extension functions to measure cartesian and haversine distance. Table 3

shows an example of spatial enhanced SPARQL query supported by AllegroGraph.

```
1.  SELECT * {
2.    GEO
3.    SUBTYPE ...
4.    RADIUS (POINT(?lon, ?lat), ?rad) {
5.      # Some patterns, possibly mentioning ?lon, ?lat, ?rad.
6.    }
7.    WHERE {
8.      ?p foo:placename "Home" ;
9.          geo:lat ?lat ;
10.         geo:lon ?lon .
11.     OPTIONAL {  ?p geo:radius ?rad .  }
12.   }
13. }
```

Table 3 – An example of spatial enhanced SPARQL query supported by AllegroGraph.

3. RUNNING EXAMPLE

In this section, we present a simple running example that requires both a light reasoning support and the features of a GIS.

As we stated in the Section 1, a large part of LD has a spatial dimension. Consider, as a notable example, the squares and the monuments of a city. In DBpedia the categories "squares and plazas by city"[8] and "visitor attraction by city"[9] exist. They both have a subcategory for each city, which in turn contains other subcategories. Actual squares and visitor attractions are leafs in these two disjoint trees of categories. Each square or visitor attraction is described with a large set of information including their latitude and longitude. Moreover, geo-tagged photos of squares and monuments already interlinked with DBpedia resources can be obtained using simple solutions like flickr™ wrappr [14]. Finally, the shape of the squares and of the buildings that contains the visitor attraction can be retrieved from OpenStreetMap.

Given this information, our running example requires to design a system able to answer the following question:

> *Which are the monuments of Italy that overlook a square in which more than 100 photos where geo-tagged?*

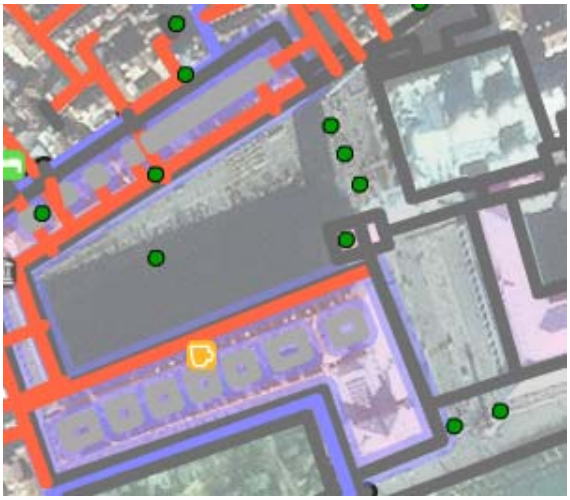

Figure 2 St Mark's Square as in OpenStreetMap. The lines in overlay represent the shape of the buildings, the boundaries of the square and the streets. The dots are point of interest.

[8] http://dbpedia.org/page/Category:Squares_and_plazas_by_city

[9] http://dbpedia.org/page/Category:Visitor_attractions_by_city

Figure 3 St Mark's Square as in Flikr, more than 2 hundred thousand photo are geotag with latitude and longitude compatible with Venice.

The query is difficult to answer for many reasons. It requires selectively traversing the transitive closure of the two DBpedia category trees in order to retrieve the squares and the visitor attractions of Italian cities. It requires retrieving the shape of the squares from OpenStreetMap (see Figure 2, for an example related to Venice), retrieving the photos from Flikr (see Figure 3, for an example related to venice), and checking if at least the geotags of 100 photos are *contained in* one of the squares. Finally, it requires retrieving from OpenStreetMap also the shape of the buildings that contains the visitor attractions and checking if the these shapes *touch* the shape of the squares (i.e., the building overlooks the square).

As an answer, for instance concerning St Mark square in Venice, we may expect:

- St Mark's Basilica
  http://dbpedia.org/page/St_Mark%27s_Basilica
- St Mark's Clocktower
  http://dbpedia.org/page/St_Mark%27s_Clocktower
- Doge's Palace
  http://dbpedia.org/page/Doge%27s_Palace

4. ARCHITECTURE

Answering the query formulated in Section 3 requires a system with the following characteristics:

1. ability to process RDF and to access and process LD such as DBpedia;
2. minimal reasoning support to compute the transitive closure of the two trees of categories;
3. (last but not least) support for comparing geometries, such as contains and touches.

Two extreme solutions for such a system can be envisioned:

- adding geometric data types and methods to a native RDF repository and extend SPARQL to handle such data types and invoke such methods; or
- providing access to non-RDF, GIS databases, using the Virtual RDF approach [9], thus extending both the mapping language and SPARQL to handle GIS data types and geometric methods.

AllegroGraph is an example of the first approach; Virtuoso is an early, but limited attempt, of the second approach strictly coupled with its SQL engine. Our GIS-to-RDF (G2R) approach is an attempt to provide a solution that can be coupled with any GIS using a mapping file.

The architecture of G2R is illustrated in Figure 4. The G2R Engine is in the centre of the figure. It enables both remote and local standard SPARQL clients to query data in a GIS as long as the G2R Engine is provided with an appropriate mapping file.

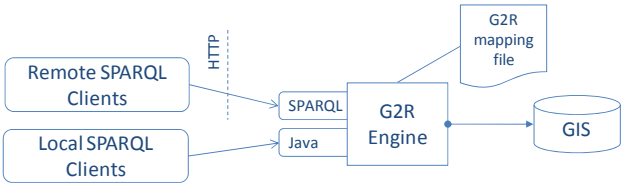

Figure 4 the architecture of G2R Engine

The mapping file format is an extension of D2RQ format [15]. G2R mapping language supports any SQL92 compatible database (thanks to D2RQ mapping language) and allows coupling any GIS for which a one-to-one mapping to SQL/MM spatial methods and data types can be written.

5. G2R MAPPING LANGUAGE

As we anticipated in Section 4, G2R mapping language extends D2RQ mapping language [9].
Both are declarative languages for describing the relation between a relational database schemata and RDF-S vocabularies (or OWL ontologies). A database schema is mapped in a RDF-S vocabulary using `d2rq:ClassMap` and `d2rq:PropertyBridge`. A ClassMap specifies how to retrieve instances of a given class from the database. PropertyBridges, instead, specify how to retrieve the pair of <property,value> for each instance, and how to relate instances with a given property.
For instance, consider the ER model in Figure 5; it represents a building in OpenStreetMap. Mapping it to RDF requires a ClassMap and a PropertyBridge.
The ClassMap maps the entity "Building" in a RDF-S Class, e.g., `http://linkedgeodata.org/vocabulary#building`; and virtually creates instances using the following IRI pattern: `http://ex.g2r.org/buildings/@@ID@@`. The PropertyBridge can map the attribute "name" of the entity "Building" in `rdfs:label` and virtually creates triples such as:

```
ex:25466045 rdfs:label "St Mark's Clocktower".
```

Figure 5 An ER model of a building in OpenStreetMap.

The attribute "area" cannot be mapped using D2RQ, because it use the geometric data type polygon. G2R offers an extension of `d2rq:PropertyBridge`, named `g2r:SpatialPropertyBridge`, that allows to map the attribute "area" to any property whose range is a `g2r:Geometry`.

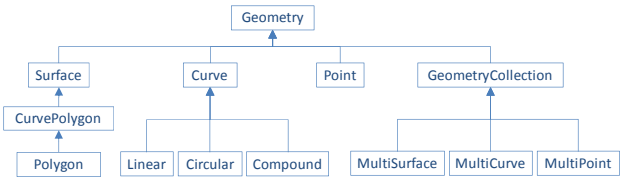

Figure 6 G2R hierarchy of classes representing SQL/MM spatial standard data types.

G2R support all geometry data types defined in SQL/MM spatial standard (see Section 2.2 of [13]). G2R hierarchy of classes is illustrated in Figure 6. For instance, the SpatialGeometryBridge in Table 4 maps the attribute "area" (identified at line 4 using the property `g2r:spatialColumn`) into the user defined property `ex:hasSurface` (see line 3) whose domain is a `g2r:Polygon` (as specified at line 5).

```
1.  map:area a g2r:SpatialPropertyBridge;
2.      d2rq:belongsToClassMap map:Building;
3.      d2rq:property ex:hasSurface;
4.      g2r:spatialColumn "area";
5.      d2rq:datatype g2r:Polygon .
```

Table 4 – An example of G2R mapping applied to the attribute "area" of the entity "Building" shown in Figure 5.

Last, but not least, G2R offers the full range of spatial methods specified in SQL/MM spatial standard (see Section 2.3 of [13]). It includes methods to:

- convert to and from exchange data formats such as GML (i.e., `g2r:AsGML`);
- retrieve properties of the geometry such as the length (i.e., `g2r:Length`), or the coordinates of a point in the geometry (i.e., `g2r:_N` where `N=1` is the start point);
- compare two geometries such as
  - `g2r:isDisjoint` that tests whether two geometries do not intersect;
  - `g2r:intersects`, `g2r:crosses`, and `g2r:overlaps` that test whether the interiors of the geometries intersect;
  - `g2r:touches` that tests whether two geometries touch at their boundaries, but do not intersect in their interiors; and
  - `g2r:isWithin` and `g2r:contains` that test whether one geometry is fully within the other.
- generate new geometries such as
  - `g2r:Point`, `g2r:Polygon`, and similar ones for all the other geometric data types that creates geometric instances;
  - `g2r:createBuffer` that generates a buffer at a specific distance around the given geometry;
  - `g2r:createConvexHull` that computes the convex hull for a geometry; and
  - `g2r:Difference`, `g2r:Intersection`, `g2r:Union` that construct the difference, intersection, or union between the point sets defined by two geometries.

All these methods can be invoked within a standard SPARQL query in the FILTER clause leveraging the Extensible Value Testing of SPARQL (see Section 11 of [8]). For an example, see lines 9 and 10 in Table 5.

6. QUERYING G2R

As explained in Section 4, a G2R Engine can be queried using a standard SPARQL query. For instance, a large part of the running example query, presented in Section 3, can be encoded in SPARQL as shown in Table 5.

```
1.  SELECT ?squareName ?attractionName
2.  FROM
3.  WHERE {
4.    ?square rdfs:label ?squareName .
5.    ?square ex:hasSurface ?squareSurface .
6.    ?attraction rdfs:label ?attractionName .
7.    ?attraction ex: hasSurface ?attractionSurface .
8.    ?photo ex:geoTagged ?point .
9.    FILTER (g2r:contains(?squareSurface, ?point) &&
10.             g2r:touches (?squareShape, ?attractionShape))
```

```
11. } GROUP BY ?squareName
12. HAVING (COUNT(?photo) > 100).
```

Table 5 – An example of SPARQL query that G2R can evaluate leveraging a GIS like PostGIS.

Let's assume that all data are stored in PostGIS[10] as three tables: squares, buildings and photos. Let's also assume that all three tables have been mapped already in RDF as illustrated in Section 5 using G2R mapping language. When the G2R Engine receives the SPARQL query in Table 5, it rewrites such a query in the SQL statement shown in Table 6. Note that such statement employs the spatial method `ST_Within` and `ST_Touches`.

```
1. SELECT s.name, b.name,
2. FROM buildings AS b, squares AS s, photos AS p
3. WHERE b.area.ST_Within(p.geoTagged) = 1 AND
4.       b.area.ST_Touches (s.area) = 1
5. GROUP BY s.name
6. HAVING COUNT (p.geoTagged) > 100
```

Table 6 – The SQL statement obtained by G2R Engine when processing the SPARQL query in Table 5.

The missing part of the running example query, presented in Section 3, is the one that transitively traverse the DBpedia category trees of "squares and plazas by city" and "visitor attraction by city" selecting only squares and visitor attraction from Italy. The complete SPARQL query, which makes use of SPARQL 1.1 [16] subquery support, is illustrated in Table 7. The two subqueries are highlighted in grey.

```
1.  SELECT ?squareName ?attractionName
2.  FROM
3.  WHERE {
4.    ?square rdfs:label ?squareName .
5.    { SELECT ?square
6.      WHERE {
7.                ?square skos:subject dbcat:squares_and_plazas_by_city
    .
8.                ?square owl:sameAs ?squareInGeoNames .
9.                FILTER (regex(str(?squareInGeoNames), "geonames")
10.               ?squareInGeoNames gn:inCountry gn:IT . }
11. }
12.   ?square ex:hasSurface ?squareSurface .
13.   ?attraction rdfs:label ?attractionName .
14. { SELECT ?attraction
15.   WHERE {
16.             ?attraction skos:subject dbcat:visitor_attraction _by_city
    .
17.             ?attraction owl:sameAs ?attractionInGeoNames .
18.           FILTER          (regex(str(?attractionInGeoNames),
    "geonames")
19.             ?attractionInGeoNames gn:inCountry gn:IT . }
20. }
21.   ?attraction ex: hasSurface ?attractionSurface .
22.   ?photo ex:geoTagged ?point .
23.   FILTER (g2r:contains(?squareSurface, ?point) &&
24.           g2r:touches (?squareShape, ?attractionShape))
25. }
26. GROUP BY ?squareName
27. HAVING (COUNT(?photo) > 100).
```

Table 7 – A SPARQL query that capture the running example query presented in Section 3.

Executing each of the two subqueries requires a SPARQL endpoint that evaluates them under RIF entailment regimes (see section 2.7 of [17]), because it has to process the two production rules shown in Table 8.

[10] PostGIS is an open source software program that adds support for geographic objects to the PostgreSQL database. Interested readers can read more at the following location http://www.postgis.org/.

```
1. IF (?i skos:subject ?x) and (?x skos:broader?y)
   THEN (?i skos:subject ?y)
2. IF (?x skos:broader ?y) and (?y skos:broader ?z)
   THEN (?x skos:broader ?z)
```

Table 8 – Two rules that capture a possible interpretation of SKOS vocabulary.

The first rule declares that if a resource belongs to a category (i.e., is related to a category by `skos:subject`), then it also belongs to the its direct super-category (i.e., a category related to the stated one by `skos:broader`). The second one allows to transitively traverse the hierarchy of categories following the `skos:broader` properties.

7. IMPLEMENTATION EXPERIENCE

Using the LarKC platform [18], we designed and implemented a G2R Engine able to process queries of the kind shown in Table 8.

The LarKC platform has a pluggable architecture in which it is possible to exploit techniques and heuristics from diverse areas such as databases, machine learning, cognitive science, Semantic Web, and Geographic Information Systems. A LarKC application consists of a number of pluggable components arranged in a workflow executed by the LarKC platform. LarKC plug-ins cover a variety of tasks, each one concretized by a different plug-in type:

- *Identification* of sources of information potentially useful to answer the query issued by the client;
- *Fetching* information from the identified sources;
- *Selection* of a relevant subset of the fetched information;
- *Translation* of information from the source format in RDF or translation of queries from SPARQL to source specific query language; and
- *Reasoning* on the collected information in order to provide answers to the query issued by the client.

Following the approach described in [19], when G2R receives the queries, first of all, it uses the Linked Data Search Engine Sindice [20] to *identify* potentially useful resources in DBpedia, and it *fetch* them. Secondly, the G2R Engine uses a *query translator* to look for subqueries like the those highlighted in Table 7; if it finds them, it evaluates them using as *reasoner* the Jena General Purpose Rule Engine [21] which support SPARQL under RIF entailment regime. Finally, for each square and visitor attraction in the same city, it rewrites the SPARQL query shown in Table 5 into the SQL statement shown in Table 6. It does so by using another *query translator* that interprets the mapping file designed by the user. The statement is evaluated by PostGIS and results are *translated* back into a SPARQL variable binding which is sent back to the client.

8. COMPARATIVE EVALUATION

A comparison of G2R, Virtuoso and AllegroGraph, on the bases of the spatial features supported, is illustrated in Table 9 and 10.

Data Type	G2R	Virtuoso	AllegroGraph
Geometry	√	√	
- Surface	√		
-CurvePolygon	√		
- Polygon	√		
- Curve	√		
- Linear	√		
- Circular	√		
- Compound	√		

- Point	√	√	√
- GeometryCollection	√		
- MultiSurface	√		
- MultiCurve	√		
- MultiPoint	√		

Table 9 – A comparison between G2R, Virtuoso and AllegroGraph supported data types.

Data Type	G2R	Virtuoso	AllegroGraph
convert to and from exchange data formats	√	√[1]	
retrieve properties	√	√[2]	
compare two geometries	√		
- is disjoint	√		
- intersects	√	√	
- crosses	√		
- overlaps	√		
- touches	√		
- is within/contains	√	√[4]	√[4]
generate new geometries	√	√[3]	
	√		
	√		
	√		

[1] limited to WKT representations
[2] limited to isGeometry
[3] limited to Point data type
[4] limited to points contained in a given radius

Table 10 – A comparison between G2R, Virtuoso and AllegroGraph supported spatial methods.

As one can read from the table, the approach of G2R to treat GIS as Virtual RDF Graphs pays off. Instead of re-implementing GIS support in Semantic Web framework, G2R allows for mapping spatial data types into any RDF-S vocabulary (or OWL ontology) and to benefit from a large set of already implemented method that operates on spatial data types.

9. CONCLUSIONS

In this paper, we reported our experience in adding a spatial dimension to the Semantic Web and, in particular, to SPARQL. We discussed the need for treating GIS as Virtual RDF Graphs, instead of re-implementing typical GIS functionalities in existing Semantic Web frameworks.

The core of our contribution is a declarative mapping language that extends D2RQ mapping language [15] to support spatial data types.

We also present our prototypical implementation of a G2R Engine based on the LarKC framework. Such prototype allows SPARQL query involving spatial computation to be executed in a mixed environment where a GIS (i.e., PostGIS) and a SPARQL endpoint, operating under RIF entailment regime (i.e., Jena General Purpose Rule Engine), are orchestrated.

The approach is evaluated by comparing G2R against two similar solution provided by Virtuoso e AllegroGraph. A broader qualitative comparison with other existing solutions as well as s quantitative comparison on a large data set is next in our research agenda.

10. ACKNOWLEDGMENTS

This research has been partially supported by the LarKC EU co-funded project (ICTFP7-215535).

11. REFERENCES

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. Int. J. Semantic Web Inf. Syst., 5(3):1-22, 2009.

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - a crystallization point for the web of data. J. Web Sem., 7(3):154-165, 2009.

[3] Sören Auer, Jens Lehmann, Sebastian Hellmann: LinkedGeoData: Adding a Spatial Dimension to the Web of Data. International Semantic Web Conference 2009: 731-746

[4] Patrick Hayes, RDF Semantics. W3C Recommendation 10 February 2004. Available on line at http://www.w3.org/TR/rdf-mt/

[5] Miles, A and Bechhofer, S.: SKOS Simple Knowledge Organization System. W3C Proposed Recommendation 15 June 2009. Available online at http://www.w3.org/TR/2009/PR-skos-reference-20090615/

[6] Brickley, Dan and Guha, Ramanatgan V.: RDF Vocabulary Description Language 1.0: RDF Schema , February (2004) .

[7] Brickley, D.: WGS84 Geo Positioning: an RDF vocabulary. February, 2004. Available online at http://www.w3.org/2003/01/geo/

[8] Prud'hommeaux, E. and Seaborne, A. (ed.): SPARQL Query Language for RDF, W3C Working Draft 4 October 2006. Available online at http://www.w3.org/TR/2006/WD-rdf-sparql-query-20061004/

[9] Chris Bizer, Richard Cyganiak, Jörg Garbers, Oliver Maresch, Christian Becker, The D2RQ Platform v0.7 - Treating Non-RDF Relational Databases as Virtual RDF Graphs. User Manual and Language Specification. August, 2009. Available online at http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/

[10] Orri Erling, Ivan Mikhailov. Mapping Relational Data to RDF in Virtuoso Virtuoso. Available online at: http://virtuoso.openlinksw.com/wiki/main/Main/VOSSQLRDF

[11] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, David Aumueller: Triplify: light-weight linked data publication from relational databases. WWW 2009: 621-630

[12] SquirrelRDF. http://jena.sourceforge.net/SquirrelRDF/

[13] Stolze, Knut: SQL/MM Spatial - The Standard to Manage Spatial Data in a Relational Database System, Vol. 26 GI (2003), S. 247-264 .

[14] Christian Becker, Chris Bizer. flickr™ wrappr. Available online at http://www4.wiwiss.fu-berlin.de/flickrwrappr/

[15] C. Bizer. D2R MAP - A Database to RDF Mapping Language. In WWW (Posters), 2003.

[16] Steve Harris and Andy Seaborne, SPARQL 1.1 Query. W3C Working Draft 22 October 2009. Available online at http://www.w3.org/TR/2009/WD-sparql11-query-20091022/

[17] Birte Glimm and Bijan Parsia, SPARQL 1.1 Entailment Regimes. W3C Working Draft 22 October 2009. Available online at http://www.w3.org/TR/2009/WD-sparql11-entailment-20091022/

[18] Fensel, D, van Harmelen F, Andersson B, Brennan P, Cunningham H, Della Valle E, Fischer F, Huang Z, Kiryakov A, Lee T K, School L, Tresp V,Wesner S,Witbrock M, and Zhong N. Towards LarKC: A Platform for Web-scale Reasoning. In Proceedings of the Second IEEE International Conference on Semantic Computing (ICSC 2008), Santa Clara, California

[19] Emanuele Della Valle, Irene Celino, Daniele Dell'Aglio: The Experience of Realizing a Semantic Web Urban Computing Application. T. GIS 14(2): 163-181 (2010)

[20] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, Giovanni Tummarello: Sindice.com: a document-oriented lookup index for open linked data. IJMSO 3(1): 37-52 (2008)

[21] Dave Reynolds. Jena 2 Inference support. Available online at http://jena.sourceforge.net/inference/