Querying Brussels Spatiotemporal Linked Open Data

Kevin Chentout and Alejandro Vaisman

Université Libre de Bruxelles
{kchentou,avaisman}@ulb.ac.be

Abstract. The "Open Semantic Cloud for Brussels" (OSCB) project aims at building a platform for linked open data for the Brussels region in Belgium, such that participants can easily publish their data, which can in turn be queried by end users using a web browser to access a SPARQL endpoint. If data are spatial and we want to show them on a map, we need to support this endpoint with an engine that can manage spatial data. For this we chose Strabon, an open source geospatial database management system that stores linked geospatial data expressed in the stRDF format (spatiotemporal RDF) and queries them using stSPARQL (spatiotemporal SPARQL), an extension to SPARQL 1.1. In this paper we show how the SPARQL endpoint is built and the kinds of queries it supports, also providing a wide variety of examples.

1 Introduction

"Open Data" aims at making public sector data easily available and encouraging researchers and application developers to analyze and build applications around that data in order to stimulate innovation, business and public interests. "Linked Data"[1] is a global initiative to interlink resources on the Web using Uniform Resource Identifiers (URI) for accessing the resources, and the Resource Description Framework (RDF) [2] for representing knowledge and annotating those resources. The conjunction of both concepts originated the notion of "Linked Open Data" (LOD). The goal of the "Open Semantic Cloud for Brussels" (OSCB) project[2] is to pave the way for building a platform for LOD for the region of Brussels. The objective is that participants benefit from the OSCB, linked data-based architecture to publish their data. To facilitate this, data will be published in RDF format. The final users should be able to query data in an ubiquitous, intuitive and simple way. In this paper we show how these data can be queried by OSCB users.

Background. RDF is a data model for expressing assertions over resources identified by a URI. Assertions are expressed as *subject-predicate-object* triples, where *subject* is always a resource, and *predicate* and *object* could be a resource

[1] http://www.linkeddata.org
[2] http://www.oscb.be

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 378–387, 2013.

or a string. *Blank nodes* are used to represent anonymous resources or resources without a URI, typically with a structural function, e.g., to group a set of statements. Data values in RDF are called *literals* and can only be *objects*. Many formats for RDF serialization exist. In this paper we use Turtle[3], and assume that the reader is familiar with this notation.

SPARQL is the W3C standard query language for RDF [4]. The query evaluation mechanism of SPARQL is based on subgraph matching: RDF triples are interpreted as nodes and edges of directed graphs, and the query graph is matched to the data graph, instantiating the variables in the query graph definition. The selection criteria is expressed as a graph pattern in the `WHERE` clause, consisting basically in a set of triple patterns connected by the '.' operator.

Contributions and Paper Organization. In OSCB, data providers deliver their data in relational tables or XML documents. To be accessed and linked, these data are mapped to RDF triples using the R2RML[4] standard. These data are used to populate a SPARQL endpoint, which can be accessed using a browser. Since OSCB data are essentially spatiotemporal, we do not only need to extend the R2RML mapping (which we explain in [1]), we also need to empower the endpoint with a spatial-enabled engine. For this, we have chosen Strabon [3], an open-source semantic geospatial database management system (DBMS) that stores linked geospatial data expressed in the stRDF format (spatiotemporal RDF) and queries these data using stSPARQL (spatiotemporal SPARQL). In this paper we describe the data, and how it can be queried using stSPARQL. In Section 2 we introduce Strabon. Then, in Section 3 we describe how we process the data that populates the endpoint. In Section 4 we present a comprehensive set of example queries. We conclude in Section 5.

2 Strabon, stRDF and stSPARQL

The availability of geospatial data in the linked data cloud has motivated research on geospatial extensions of SPARQL. These works have formed the basis for GeoSPARQL, a proposal for an Open Geospatial Consortium (OGC)[5] standard, currently at the "candidate standard stage". *Strabon* [3] is an open-source semantic geospatial database management system that extends the RDF store *Sesame*, allowing it to manage both thematic and spatial RDF data stored in the PostGIS[6] spatial DBMS. In this way, Strabon provides features similar to those offered by geospatial DBMS that make it one of the richest RDF stores with geospatial support available today. These features made us chose Strabon as the spatial RDF data engine for the OSCB project, and the backend for our spatial data-enabled endpoint. In this way, Strabon allows us to support spatial queries and display their result on a map.

[3] `http://www.w3.org/TeamSubmission/turtle/`
[4] `http://www.w3.org/TR/r2rml/`
[5] `http://opengeospatial.org`
[6] `http:postgis.org`

Strabon works over stRDF, a spatiotemporal extension of RDF. In stRDF, the data types `strdf:WKT` and `strdf:GML` represent geometries serialized using the OGC standards WKT and GML. WKT is a widely accepted OGC standard and can be used for representing geometries, coordinate reference systems and transformations between coordinate reference systems. Since our contribution is built on Strabon, we next explain the latter's main features in order to make the paper self-contained.

The stSPARQL language extends SPARQL 1.1 with functions that take as arguments spatial terms and can be used in the SELECT, FILTER, and HAVING clause of a SPARQL query. A spatial term is either a spatial literal (i.e., a typed literal with data type `strdf:geometry` or its subtypes), a query variable that can be bound to a spatial literal, the result of a set operation on spatial literals (e.g., union), or the result of a geometric operation on spatial terms (e.g., buffer). Also, stSPARQL can express spatial selections, i.e., queries with a `FILTER` function with arguments a variable and a constant, and spatial joins, i.e., queries with a `FILTER` function like `strdf:contains(?geoA, ?geoB)`.

The stSPARQL extension functions can also be used in the `SELECT` clause of a SPARQL query. As a result, new spatial literals can be generated on the fly during query time based on pre-existing spatial literals. For example, to obtain the buffer of a spatial literal that is bound to the variable ?geo, we would use the expression `SELECT (strdf:buffer(?geo,0.01) AS ?geobuffer)`. We show examples of how we exploit these features in Section 4.

In stSPARQL aggregate functions over geospatial data are supported, like: (1) `strdf:geometry strdf:union(set of strdf:geometry a)`, returns a geometry that is the union of the set of input geometries. (2) `strdf:geometry strdf:intersection (set of strdf:geometry a)`, returns the intersection of the set of input geometries. (3) `strdf:geometry strdf:extent(set of strdf:geometry a)`, returns the minimum bounding box of the set of input geometries.

3 Building the SPARQL Endpoint

In previous work [1] we showed how we can map spatial data to the stRDF format extending the R2RML[7]. standard. The Strabon-powered stSPARQL endpoint which we analyze in this paper contains three datasets from Belgian organizations: Agenda.be, Bozar, and STIB. These datasets are complemented by external data sources, which provide the context for analysis. These data was obtained from other data publishers. We next present these datasets.

Organizational Data. The **Agenda.be** data reports cultural events in Brussels and the institutions where they take place. It is delivered in two XML documents: events.xml, containing events and institutions.xml. The size of the XML files is approximately 6MB. Applying the spatial R2RML mapping we obtain triples like the following one, containing the spatial coordinates of the

[7] `http://www.w3.org/TR/r2rml/`

institution where the event takes place. Note that we use the WGS84 standard coordinate system for the geographic data reference.

```
...
<http://www.agenda.be/db/Address_of_Institution/3>
      ...
      <http://www.w3.org/2003/01/geo/wgs84_pos#geometry>
              "POINT(4.357527 50.854287);http://www.opengis.
                  net/def/crs/EPSG/0/4326"^^<http://strdf.di.
                  uoa.gr/ontology#WKT> .
```

The **BOZAR database** contains approximately 130MB of data. Addresses are stored in a table denoted location_lng. This table contains information about name, address, zip and city. Its schema has the form (id, lng, field, content). The address information is represented as values of the attribute field. We preprocessed this table to obtain a new one with schema (fullAddress,id). As an example, applying the mapping over the following tuple in this table: $<$ *ABC Factory* 175 *rue Bara* 1070 *Bruxelles Belgique* $>$, results in the triple (corresponding to a location with id=110):

```
<http://bozar.be/db/Locations/110>
      a       <http://starpc18.vub.ac.be:8080/gospl/ontology
          /2#Location> ;
      <http://www.w3.org/2003/01/geo/wgs84_pos#geometry>
              "POINT(4.292185 50.82917);http://www.opengis.
                  net/def/crs/EPSG/0/4326"^^<http://strdf.di.
                  uoa.gr/ontology#WKT> .
```

The **STIB**, *Société des Transports Intercommunaux de Bruxelles* is the main company of public transport in Brussels. Spatiotemporal data from STIB are stored in an SQL database in four tables: Block, Stop, Trip and Tripstop. A Block can be seen as the time elapsed between the moment in which a vehicle leaves the warehouse and returns to it. The Stop table corresponds to a stop (bus, tram or metro) with its description, in French and Dutch, and its location, in x and y coordinates as well as in GPS coordinates (longitude / latitude). A Trip is a part of a block and it is defined as the path between the starting point and the ending point for a route. Finally, the Tripstop describes the stop and the time during a trip when the vehicle reaches the stop. The spatial mapping here is straightforward because we already have the longitude and latitude of a stop. For example, the instance of in Table 1 produces the following triple representing the location of stop 8042.

```
<http://www.stib.be/location/8042>
   a    <http://starpc18.vub.ac.be:8080/gospl/ontology/60#
       Location> ;
   <http://starpc18.vub.ac.be:8080/gospl/ontology/60#
       Location_of_Stop>
         <http://www.stib.be/stop/8042> ;
   ...
   <http://www.w3.org/2003/01/geo/wgs84_pos#geometry>
```

Table 1. Sample of Stop table

stp_identifier	stp_description	ud_stp_desc_flam	stp_longitude	stp_latitude
8042	ARTS-LOI	KUNST-WET	4.36963	50.8453
8032	PARC	PARK	4.36293	50.8458

```
        "POINT(4.36963 50.8453);http://www.opengis.net/def/
            crs/EPSG/0/4326"^^<http://strdf.di.uoa.gr/
            ontology#WKT> .
...
```

External Datasets. To provide context to the analysis we added places of interest in Brussels, e.g., cafes, restaurants, banks, schools, etc. To obtain these data we queried three datasets from the linked open data web: DBPedia, Geonames and LinkedGeoData. **DBPedia** is a dataset containing structured information extracted from Wikipedia. This dataset is available on the Web and can be queried via a SPARQL endpoint[8]. For data extraction, we built a tool in Java, which queries the DBPedia service and returns RDF triples in a file. For example, a portion of a triple representing the Royal Palace looks like:

```
<http://dbpedia.org/resource/Royal_Palace_of_Brussels>
      <http://www.w3.org/2000/01/rdf-schema#label>
              "Royal Palace of Brussels@en" ;
      <http://www.w3.org/2003/01/geo/wgs84_pos#geometry>
              "POINT(4.36222 50.8417);http://www.opengis.net/
                  def/crs/EPSG/0/4326"^^<http://strdf.di.uoa.
                  gr/ontology#WKT> ;
       ...
      <http://xmlns.com/foaf/0.1/isPrimaryTopicOf>
              "http://en.wikipedia.org/wiki/
                  Royal_Palace_of_Brussels" .
```

We got 50 different places in Brussels for which we know the location. The second source we used was **Geonames**[9], a geographical database containing information from over 8 million place. A geonames Java library allows us to access the geonames web services. This added 114 different places for Brussels. Finally, we search data in **LinkedGeoData**, a spatial database derived from OpenStreetMaps, a project aimed at building a free world map database. These data can be accessed and queried via a SPARQL endpoint[10]. Similar to what we did for the DBPedia dataset, we queried a service and build RDF triples from the results. In total, we collected 2355 places from the external sources, which we aded to our basic dataset.

[8] http://dbpedia.org/sparql
[9] http://geonames.org
[10] http://live.linkedgeodata.org/sparql

4 Querying the SPARQL Endpoint

We next give examples of queries that the endpoint supports[11], classifying them according their type. We remark that this classification is just aimed at organizing the presentation and does not pretend to be a query taxonomy. In the queries that follow, we omit the prefixes for the sake of space.

Queries over a Single Dataset. These queries are posed over just one dataset. That means that there is a unique "FROM" clause in the SPARQL query, which is mandatory.
The following query asks for STIB route data. The result is given in Figure 1a.
Query 1. *"Give me all STIB stops corresponding to route 25"*

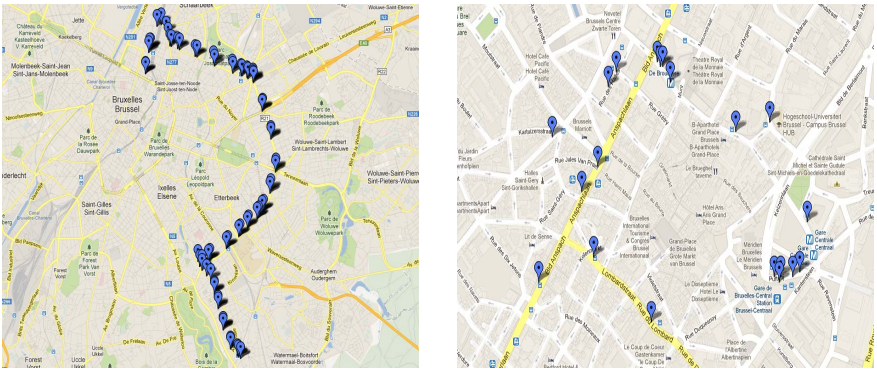
```
SELECT DISTINCT ?stop_loc ?geo ?description
FROM <http://stib.be>
WHERE {
  ?name stib:Route_with_Name "25" .
  ?route_name stib:Route_with_Route_Name ?name.
  ?trip stib:Trip_with_Route ?route_name .
  ?trip stib:Trip_with_Trip_Stop ?trip_stop .
  ?trip_stop stib:Trip_Stop_with_Stop ?stop_loc .
  ?stop_loc stib:Stop_with_Description ?description.
  ?node stib:Location_of_Stop ?stop_loc.
  ?node geo:geometry ?geo.
}
```

In this query, in variable `?stop_loc` we return a link to the triple corresponding to the stop location. In variable `?geo` we return the geometry (the point coordinates, which allows us to display the result in a map), and in `?description` the name of the stop. The rest of the query is self-descriptive.

Queries Including Aggregate Functions. These queries include a `GROUP BY` clause. The following query computes the twenty stops closest to the Brussels central square, displaying the result in a map (Figure 1b).
Query 2. *"Twenty stops closest to the Grand Place"*

```
SELECT distinct ?stop (GROUP_CONCAT(distinct ?line) AS ?
    someLine) (SAMPLE(?description) AS ?someDescription) ?geo
     (strdf:distance(?geo, "POINT (4.3525 50.8467);http://www
    .opengis.net/def/crs/EPSG/0/4326", <http://www.opengis.
    net/def/uom/OGC/1.0/metre>)as ?dist )
FROM <http://stib.be>
WHERE {
  ?stop_loc a stib:Location .
  ?stop_loc geo:geometry ?geo.
  filter(strdf:distance(?geo, "POINT (4.3525 50.8467);http://
      www.opengis.net/def/crs/EPSG/0/4326", <http://www.
      opengis.net/def/uom/OGC/1.0/metre>) < 1000)
```

[11] The complete list of example queries can be found at the endpoint in the address `http://eao4.ulb.ac.be:8080/strabonendpoint/`



(a) **(b)**

Fig. 1. (a) Stops of Tram 25; (b) Twenty stops closest to Brussels central square

```
  ?stop_loc stib:Location_of_Stop ?stop .
  ?stop stib:Stop_with_Description ?description .

  ?trip_stop stib:Trip_Stop_with_Stop ?stop .
  ?trip_stop stib:Trip_Stop_of_Trip ?trip .

  ?trip stib:Trip_with_Route ?route_name .
  ?route_name stib:Route_with_Route_Name ?name.
  ?name stib:Route_with_Name ?line .
 }
GROUP BY ?stop ?geo ?dist
ORDER BY ?dist
LIMIT 20
```

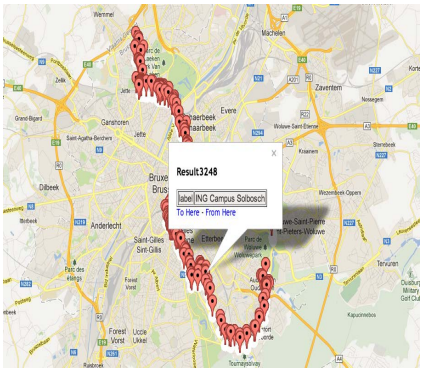
Note that in the `SELECT` clause we compute the distance between a stop (variable `?geo`) and the position of the Grand Place, which we assume the user (or an application querying the endpoint) knows. It is returned in variable `?dist`.

Statistical Queries. This class includes queries returning statistical data obtained from the datasets. These queries are normally also aggregate queries.
Query 3. "*Number of stops per route*"

```
SELECT (str(?res) as ?line) (count(*) as ?total)
FROM <http://stib.be>
WHERE {
  ?name stib:Route_with_Name ?line .
  ?route stib:Route_with_Route_Name ?name.
  ?route stib:Route_with_Stop ?stop .
  BIND( xsd:integer(?line) as ?res)
}
GROUP BY ?res
ORDER BY ?res
```

line	total
"1"	"44"^^<http://www.w3.org/2001/XMLSchema#integer>
"2"	"40"^^<http://www.w3.org/2001/XMLSchema#integer>
"3"	"58"^^<http://www.w3.org/2001/XMLSchema#integer>
"4"	"42"^^<http://www.w3.org/2001/XMLSchema#integer>
"5"	"61"^^<http://www.w3.org/2001/XMLSchema#integer>
"6"	"54"^^<http://www.w3.org/2001/XMLSchema#integer>
"7"	"77"^^<http://www.w3.org/2001/XMLSchema#integer>
"12"	"15"^^<http://www.w3.org/2001/XMLSchema#integer>
"13"	"33"^^<http://www.w3.org/2001/XMLSchema#integer>
"14"	"34"^^<http://www.w3.org/2001/XMLSchema#integer>
"15"	"54"^^<http://www.w3.org/2001/XMLSchema#integer>
"17"	"30"^^<http://www.w3.org/2001/XMLSchema#integer>
"20"	"30"^^<http://www.w3.org/2001/XMLSchema#integer>
"21"	"57"^^<http://www.w3.org/2001/XMLSchema#integer>
"22"	"29"^^<http://www.w3.org/2001/XMLSchema#integer>
"25"	"65"^^<http://www.w3.org/2001/XMLSchema#integer>
"27"	"51"^^<http://www.w3.org/2001/XMLSchema#integer>
"28"	"39"^^<http://www.w3.org/2001/XMLSchema#integer>
"29"	"60"^^<http://www.w3.org/2001/XMLSchema#integer>
"32"	"83"^^<http://www.w3.org/2001/XMLSchema#integer>
"34"	"42"^^<http://www.w3.org/2001/XMLSchema#integer>

(a)



(b)

Fig. 2. (a) Number of stops per route; (b) Places of interest within 200m of a stop of tram 94

The result is displayed in Figure 2a, and it is in textual format. Note that the `BIND` function allows to display the result ordered by route number.

Queries Involving Several Datasets. This class includes queries over many different datasets, that means, there is more than one `FROM` clause in the query. The query below retrieves data from Agenda.be and Bozar, as we can see in the two `FROM` clauses. This is also an aggregate query. Note that we do not distinguish from which dataset the information is retrieved, since the data from Bozar and Agenda.be are organizational, and were mapped using the common vocabulary given by the GOSPL ontology (prefix `gospl`).

Query 4. *"Places which host at least 10 events"*

```
SELECT ?nameInstit ?geo1 ?street ?zip ?city (GROUP_CONCAT(?
    title; separator = ' - / - ') as ?events)
FROM <http://bozar.be>
FROM <http://agenda.be>
WHERE {
  ?event gospl:Event_taking_place_at_Institution ?instit .
  ?instit gospl:Institution_with_Address ?addr .
  ?addr geo:geometry ?geo1 .

  ?event gospl:Event_has_Title ?title .
  FILTER(langMatches(lang(?title), "FR"))
  ?instit gospl:Institution_with_Name ?nameInstit .
  FILTER(langMatches(lang(?nameInstit), "FR"))

  ?addr gospl:Address_with_Street ?street .
  FILTER(langMatches(lang(?street), "FR"))
  ?addr gospl:Address_with_Postal_Code ?zip .
  ?addr gospl:Address_with_City ?city.
  FILTER(langMatches(lang(?city), "FR"))
```

```
 }
GROUP BY ?nameInstit ?geo1 ?street ?zip ?city
Having (count(*) > 9)
```

Queries Drawing Buffers and Lines on a Map. This class includes queries whose output includes drawing a line between two points on a map, or a buffer around a geometric figure. The result of the next query is depicted in Figure 2b.
Query 5. *"Places of interest in a radius of 200m from a stop of line 94"*

```
SELECT (strdf:buffer(?geo1, 200, <http://www.opengis.net/def/
    uom/OGC/1.0/metre>) as ?buf) ?geo ?label
WHERE {
  ?name stib:Route_with_Name "94" .
  ?route stib:Route_with_Route_Name ?name.
  ?route stib:Route_with_Stop ?stop .

  ?stop stib:Stop_with_Description ?description.
  ?stop stib:Stop_with_Location ?loc .
  ?loc geo:geometry ?geo1.

  ?node a ?type .
  ?node geo:geometry ?geo .
  filter(strdf:distance(?geo, ?geo1, <http://www.opengis.net/
      def/uom/OGC/1.0/metre>) < 200)
  ?node rdfs:label ?label .
}
```

Queries Including Spatial and Temporal Conditions. Note that this query also uses a buffer, which is returned in the `SELECT` clause.
Query 6. *"Places hosting events on May 23rd, 2013 in a radius of 200m of a stop of line 3"*

```
SELECT ?geo ?node ?title (GROUP_CONCAT(?description ;
    separator=", ") AS ?someDescription) ?start ?end (strdf:
    buffer(?geo, 200, <http://www.opengis.net/def/uom/OGC
    /1.0/metre>) as ?buf)
FROM <http://agenda.be>
FROM <http://bozar.be>
FROM <http://stib.be>
WHERE {
  ?node gospl:DateTimeSpecification_valid_from_Date ?start .
  ?node gospl:DateTimeSpecification_valid_until_Date ?end .
  FILTER(?start <= "2013-05-23"^^xsd:date && "2013-05-23"^^
      xsd:date <= ?end)
  ?node gospl:Event_has_Title ?title .
  FILTER(langMatches(lang(?title), "FR"))
  ?node gospl:Event_with_Description ?description .
  FILTER(langMatches(lang(?description), "FR"))
```

```
  ?node gospl:Event_taking_place_at_Institution ?inst .
  ?inst gospl:Institution_with_Address ?addr .
  ?addr geo:geometry ?geo .

  ?name stib:Route_with_Name ?line .
  FILTER (?line = "3")
  ?route stib:Route_with_Route_Name ?name.
  ?route stib:Route_with_Stop ?stop .
  ?stop stib:Stop_with_Location ?loc .
  ?loc geo:geometry ?geo1.

  FILTER (strdf:distance(?geo, ?geo1, <http://www.opengis.net
      /def/uom/OGC/1.0/metre>) < 200)
}
GROUP BY ?geo ?node ?title ?start ?end ?buf
```

5 Conclusion

In this paper we have described how a spatial data-enabled SPARQL endpoint has been built in the framework of the OSCB project. The endpoint is powered by the Strabon open-source semantic geospatial DBMS. Data are stored in the stRDF format, and are queried using an spatiotemporal extension to SPARQL, called stSPARQL. We have focused on illustrating the interesting kinds of queries that our solution supports, to give an idea of how external applications could make use of our platform.

Acknowledgement. Alejandro Vaisman has been partially funded by the "Open Semantic Cloud for Brussels (OSCB)" project, funded by Innoviris.

References

1. Chentout, K., Vaisman, A.: Mapping Spatial Data using R2RML (submitted, 2013)
2. Klyne, G., Carroll, J.J., McBride, B.: Resource Description Framework (RDF): Concepts and Abstract Syntax (2004), http://www.w3.org/TR/rdf-concepts/
3. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Strabon: A Semantic Geospatial DBMS. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 295–311. Springer, Heidelberg (2012)
4. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (2008), http://www.w3.org/TR/rdf-sparql-query/