CREATING VIRTUAL SEMANTIC GRAPHS ON TOP OF BIG DATA FROM SPACE

Konstantina Bereta and Manolis Koubarakis

National and Kapodistrian University of Athens

ABSTRACT

We present the system Ontop-spatial for the integration of geospatial data from different sources and different formats using ontologies and mappings. Ontop-spatial answers GeoSPARQL queries over geospatial relational databases storing vector or raster data by performing on-the-fly GeoSPARQL-to-SQL translation using ontologies and mappings. Our experimental evaluation shows that Ontop-spatial outperforms all state-of-the-art geospatial RDF stores.

Index Terms— GeoSPARQL, RDF, Ontology-based Data Access, Geospatial Data Integration

1. INTRODUCTION

Previous projects TELEIOS, LEO and Melodies funded by FP7 ICT, and OBEOS funded by ESA have demonstrated the use of linked data in Earth Observation (EO). The current H2020 project Copernicus App Lab (`http://www.app-lab.eu/`) goes one step further by making data from three Copernicus services (Land, Marine and Atmosphere) available on the Web as linked data to aid their utilization by mobile developers. In previous projects, it has been assumed that EO data are transformed from their original formats (shapefiles, spatially-enabled relational databases, GeoTIFF, NetCDF etc.) into RDF, stored in geospatial RDF stores and queried using geospatial extensions of SPARQL to develop interesting applications. In this paper, we present the system Ontop-spatial which enables the creation of *virtual RDF graphs over EO data stored in their original formats* using ontologies and mappings. Ontop-spatial allows EO data centers to make their data available as linked data that can be queried using the OGC standard GeoSPARQL [1], without first having to translate this data into RDF. Ontop-spatial scales to big geospatial data and it is more efficient than related geospatial RDF stores. Ontop-spatial is available as open source at `https://ontop-spatial.di.uoa.gr`.

This work has been funded by the EU project Copernicus App Lab (730124).

Ontop-spatial adopts the Ontology-Based Data Access (OBDA) paradigm pioneered by the Semantic Web community, and it is the *first geospatial OBDA system*. Ontop-spatial is able to connect to geospatial databases and create geospatial RDF graphs on top of them, using ontologies (that are extensions of the GeoSPARQL ontology) and R2RML mappings. Figure 1 shows graphically the classes of the GeoSPARQL ontology.

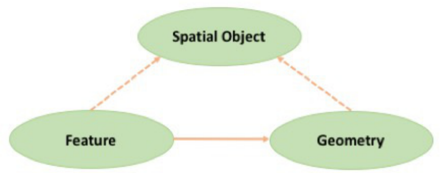


Fig. 1. Classes of the GeoSPARQL ontology

R2RML (`https://www.w3.org/TR/r2rml/`) is the standard language for encoding how relational data is mapped into RDF terms. This virtual approach avoids the need of materialization and facilitates data integration, as it enables users to pose the same GeoSPARQL queries they would pose over the materialized RDF data. GeoSPARQL queries are translated by Ontop-spatial on-the-fly into the respective SQL queries with spatial operators, and are evaluated in the geospatial DBMS. Currently, PostGIS, Spatialite and Oracle Spatial are supported as back-ends. The first version of Ontop-spatial dealing with vector data only has been presented in [2]. This version has been used in three environmental applications in the context of project MELODIES, and in a marine-security application in the context of German national project EMSec.

The contribution of our approach with respect to the Big Data from space dimensions are the following. **Volume:** Ontop-spatial outperforms the state-of-the-art in geospatial RDF stores and is able to process tens of Gigabytes of data containing complicated geometries. **Velocity:** When data gets frequently updated, using traditional triple stores is inefficient, as batches of data need to be converted and materialized as RDF triples each time they arrive. Our approach eliminates as much as possible the need for materializing data and it is suitable for data sources that get frequently updated (e.g., streams). **Variety:** With raster and OPenDAP support in place, Ontop-spatial becomes the first GeoSPARQL query engine that is able to process such a wide variety of geospatial formats, enabling geospatial data integration using ontologies and mappings. **Value:** Exposing geospatial data as virtual RDF triples that can be accessed in the Web though standard (Geo)SPARQL endpoints enables the interlinking of EO data with other data (e.g., open data) increasing their value, as data from multiple geospatial sources can be combined and rich queries can be expressed over them.

2. ONTOP-SPATIAL

Since the publication of [2] which discussed how Ontop-spatial can be used to query vector data, we have extended Ontop-spatial with the ability to query raster data as well. Querying raster data sources using declarative query languages can also be done using array DBMSs such as Rasdaman, MonetDB and SciDB. As GeoSPARQL does not include support for raster data, in our approach we do not deviate from the standard but instead: i) we overload existing vector GeoSPARQL operators such as `geof:sfIntersects` to be used with raster data as well, and ii) in the mappings, we use the raster functions supported by the underlying DBMS (e.g., PostGIS with the raster support).

More recently, work on the SciSPARQL query language showed how to query grid coverages using a hybrid data store composed of Rasdaman and a main-memory RDF store [3]. We deviate from this approach by not extending (Geo)SPARQL with array functionalities but allowing for the encapsulation of raster data functions in the mappings, so that not every raster cell needs to be represented in RDF.

The problem of representing and querying raster data as linked data has also been discussed in the recent working note ”Coverages in Linked Data” by the OGC/W3C Spatial Data on the Web working group (`https://www.w3.org/2015/spatial/wiki/Coverages_in_Linked_Data`).

None of the geospatial extensions of the framework of RDF and SPARQL, such as stRDF and stSPARQL and GeoSPARQL have considered support for raster data. The main challenge that lies behind this is twofold. First, a raster file is associated with a geometry only as a whole. It is not straightforward to associate separate raster cells to a geometry; they have to be vectorized first (i.e., translated into polygons). Second, every raster cell is associated with one or more values. In order to convert all information contained in a raster file into RDF, then multiple triples should describe a raster cell, producing a large amount of triples for a whole raster file. However, not all of this information is needed. In most of the use cases, only the information that derives from a raster file and satifies certain criteria (e.g., value constraints) is all that is needed to be converted into RDF. This means that the raster file needs to be processed and then the results of this processing are useful as RDF, while any other information is redundant. These challenges have discouraged the scientific community from converting and materializing raster data to RDF. The following example describes how raster data can be mapped into virtual RDF data. For the convenience of the reader, we present the mappings using the OBDA native language of Ontop instead of R2RML. as it is more compact and readable, but R2RML is also supported in the system.

```
mappingId chicago2
target geo:{geom} rdf:type f:rastCell;
            geo:asWKT {geom} .
source  select ST_DumpAsPolygons(rast)
as geom  from chicago;
```

In the example described above, a GeoTIFF image has been imported into a PostGIS database as relation `chicago`. The mapping shows how raster data stored in column `rast` are mapped to geometries in WKT format, after they are vectorized, using the PostGIS `ST_DumpAsPolygons` function. This is a procedure that allows domain experts to use all geometries that they may have in a database uniformly, and execute spatial operations involving vector and raster geometries. Domain experts usually perform this vectorization step as part of preprocessing. In the mapping described above, we show how this can be done on-the-fly, using Ontop-spatial.

In the project Copernicus App Lab, Ontop-spatial has also been extended to support data sources made available via OPenDAP services offered by our partner Dutch company RAMANI. OPeNDAP is a framework for accessing scientific data (`https://www.opendap.org/`) which is widely used by Earth scientists, as it is popular in large organizations such as NASA and NOAA. Earth science data can be consumed by using a specific OPenDAP client. To make data provided by OPeNDAP services available as linked data, the data should be downloaded, materialized and then converted into RDF using custom programs, as existing applications that convert geospatial data into RDF do not offer support for OPeNDAP. The approach that we describe in this paper enables the creation of virtual geospatial RDF graphs on top of data that is accessible through OPeNDAP on-the-fly, without materializing the original data or the RDF data.

Ontop-spatial has been extended with an adapter that enables it to retrieve data from an OPeNDAP server, create a table view on-the-fly, populate it with this data and create virtual semantic geospatial graphs over it. To achieve this, Ontop-spatial utilizes the system MadIS (`https://github.com/madgik/madis`) as a back-end. MadIS is an extensible relational database system built on top of the APSW SQLite wrapper. MadIS is a framework that provides a Python interface so that users can easily implement user-defined functions (UDFs) as row, aggregate functions, or virtual tables. We used MadIS in order to create a new UDF, named `Opendap`, that is able to create and populate a virtual table on-the-fly with data retrieved from an OPeNDAP server. In this way, Ontop-spatial enables users to pose GeoSPARQL queries on top of OPeNDAP data sources without materializing any triples or tables.

An example is provided below.

```
mappingId opendap_mapping
target lai:{id} rdf:type lai:Observation ;
       lai:{id} lai:hasLai {LAI}^^xsd:float;
       lai:detectionTime {time}^^xsd:dateTime;
       geosparql:asWKT {wkt}^^geo:wktLiteral .
source select id, LAI, time, wkt
       from (ordered opendap
       url:https://analytics.ramani.ujuizi.com/
       %28https://ramani.ujuizi.com/
       thredds/dodsC/Copernicus-Land-timeseries-
       global-LAI%29/readdods/.LAI/)
       where LAI > 0
```

In this mapping, the `source` is a Leaf Area Index (LAI) dataset with resolution of 100 meters is provided through an OPeNDAP server. The dataset contains observations about the LAI values of areas, as well as the time and location for each observation. The MadIS operator `Opendap` retrieves this data and populates a virtual database table with the schema `(id,LAI,time,wkt)`. The column `id` was not originally in the dataset but it is constructed from the location and time when the observation is taken. The LAI column stores the LAI values of an observation as `float` values. The attribute `time` represents the timestamp of an observation in `datetime` format. In the original dataset temporal values are represented as numeric values. The meaning of these values is described in the metadata. For example, it can be days or months since a fixed timestamp. Unfortunately, this is not a standard representation that we would have available if we had imported the dataset into a geospatial database. Because of the fact that the `Opendap` operator is implemented as an SQL user-defined operator, it can be embedded into any SQL query. So we refined the data that we want to be translated into virtual RDF terms by adding an SQL filter to the query to eliminate the negative or zero LAI values by filtering them out at an intermediate level, so that i) we do not change the values of the original dataset and ii) we provide only the correct values to the users so that they do not need to handle the noise themselves (e.g., by using GeoSPARQL filters or custome code).

The `target` part of the mapping encodes how the relational data can now be mapped into RDF terms. Every row in the virtual table describes an instance of the class `lai:Observation`. The values of the LAI column populate the triples that describe the LAI values of the Observation, and the values of the columns `time` and wkt populate the triples that describe the time and location of the observations accordingly.

Given the mappings provided above, we can pose the the following GeoSPARQL query to retrieve the Leaf Area Index values and the geometries of areas

```
select distinct ?s ?g ?lai where {
?s lai:hasLai ?lai .
?s geo:asWKT ?g }
```

Notably, both translation steps are performed on-the-fly and only after a GeoSPARQL query is posed to the system. This approach goes considerably beyond the previous version of Ontop-spatial that could only connect to an existing database with materialized tables, as well as the default, non-spatial version of Ontop and any other RDB2RDF system. OBDA systems traditionally connect to an existing database with materialized tables and access it before a query is fired in order to collect metadata, etc. The exact schema of the database tables is known beforehand. The approach that we propose in this paper is *schema-agnostic*: Ontop-spatial does not know the schema of the data as there is no database materialized. The virtual table is only created on-the-fly.

Ontop-spatial can be available as a GeoSPARQL endpoint, and thus can be used both by federation and interlinking engines. For example, one can use the interlinking tool Silk (`http://silkframework.org/`) to interlink Copernicus data that is accessible as RDF graphs using Ontop-spatial with linked data that is available using standard (Geo)SPARQL endpoints.

3. EVALUATION

We have evaluated Ontop-spatial by extending the benchmark Geographica with support for the evaluation of OBDA systems. Geographica (`http://geographica.di.uoa.gr`) was initially designed to evaluate the performance of geospatial RDF stores. We compared Ontop-spatial with the state-of-the-art geospatial RDF store Strabon (`http://strabon.di.uoa.gr`) [4]. Strabon has also been developed by our group and has been shown to be the most efficient geospatial RDF store available today [5]. Our evaluation showed that Ontop-spatial generally achieves *significantly better performance* than Strabon, often by orders of magnitude, when a large number of geospatial intermediate results are generated during the evaluation of a query. For example, Ontop-spatial is able to execute spatial selections and spatial joins against a 30 GB dataset that contains complex geometries (i.e., from points to polygons containing thousands of points) in less than a second. A summary of the experiments that we carried out is illustrated in Figure 2.

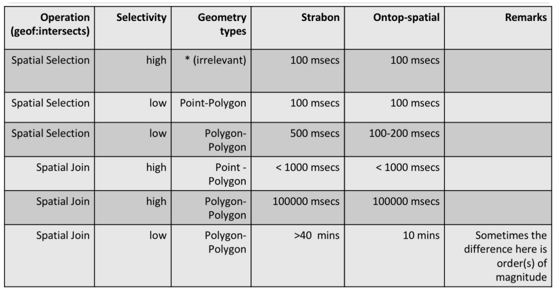
For the experiments, we used as set of spatial selection queries and a set of spatial joins. The queries in both sets contain a spatial filter. In spatial selections, one of the two arguments of the filter is a spatial constant, for example it can be a point or a linestring or a polygon. We experimented with different kinds of spatial constants with variant number of points per geometry in order to construct queries with various spatial selectivity. To construct spatial selections with low selectivity we used polygons that cover a large area so that most of the geometries of the dataset are contained in this area. We did the opposite to construct highly selective queries. We used the same approach for the spatial joins. The difference is that in spatial joins both arguments of the spatial filter are spatial variables, not constants. As shown in Figure 2, both systems have execution times at the same scale in spatial selections, regardless of the spatial selectivity of the queries. The difference in the performance of the two systems increases in spatial selections where the geometries involved are more complex (i.e., polygons).

In spatial joins the difference in execution times between Ontop-spatial and Strabon increases even more, especially in queries with low selectivity where complex geometries are involved. This is because the SQL queries that are produced by Strabon contain larger number of joins in comparison with Ontop-spatial, because of the schema of the database that serves as a back-end of Strabon, and the fact that all geometries are stored in a separate table which is R-tree indexed. In Ontop-spatial, on the other hand, every dataset is imported into the database as a separate table. The geometries are stored in a separate column of this dataset and are indexed using R-tree. So when a spatial join involves geometries from two datasets, only these two tables will be involved in the query evaluation. This issue is very common in RDF stores, as triple stores by nature store information about triples, whereas the relational model is more compact. This is the reason why Strabon, that extends the RDBMS version of Sesame triple store, creates a database that is a lot larger than the one that Ontop-spatial uses.

Both the functionality and the performance achieved by Ontop-spatial make it the system of choice for making geospatial data available using linked data technologies.

4. CONCLUSIONS

We presented an extension of the OBDA paradigm for accessing vector and raster data, as well as data offered through DAP services. Our future work will concentrate on extending Ontop-spatial with the ability to query array databases.



Operation (geof:intersects)	Selectivity	Geometry types	Strabon	Ontop-spatial	Remarks
Spatial Selection	high	* (irrelevant)	100 msecs	100 msecs	
Spatial Selection	low	Point-Polygon	100 msecs	100 msecs	
Spatial Selection	low	Polygon-Polygon	500 msecs	100-200 msecs	
Spatial Join	high	Point - Polygon	< 1000 msecs	< 1000 msecs	
Spatial Join	high	Polygon-Polygon	100000 msecs	100000 msecs	
Spatial Join	low	Polygon-Polygon	>40 mins	10 mins	Sometimes the difference here is order(s) of magnitude

Fig. 2. Overview of evaluation results

5. REFERENCES

[1] "Open Geospatial Consortium. OGC GeoSPARQL - A geographic query language for RDF data," OGC Candidate Implementation Standard, 2012.

[2] K. Bereta and M. Koubarakis, "Ontop of geospatial databases," in *ISWC 2016*.

[3] Andrej Andrejev, Dimitar Misev, Peter Baumann, and Tore Risch, "Spatio-temporal gridded data processing on the semantic web," in *DSDIS 2015*.

[4] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis, "Strabon: A Semantic Geospatial DBMS," in *ISWC*, Philippe Cudr-Mauroux and et al., Eds. 2012, vol. 7649 of *LNCS*, pp. 295–311, Springer.

[5] G. Garbis, K. Kyzirakos, and M. Koubarakis, "Geographica: A Benchmark for Geospatial RDF Stores," ISWC 2013.