Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

GeoSPARQL query support for scientific raster array data

Shahed Bassam Almobydeen, José R.R. Viqueira *, Manuel Lama

Centro Singular de Investigación en Tecnoloxías Intelixentes (CITIUS), Universidade de Santiago de Compostela (USC), Rúa de Jenaro de la Fuente Domínguez, 15782, Santiago de Compostela, Spain

ARTICLE INFO

Keywords:
Geospatial linked data
Scientific linked data
Array linked data
Raster linked data
GeoSPARQL
Spatial query processing

ABSTRACT

This paper presents the design of a GeoSPARQL query processing solution for scientific raster array data, called GeoLD. The solution enables the implementation of SPARQL endpoints on top of OGC standard Web Coverage Processing Services (WCPS). Thus, the semantic querying of scientific raster data is supported without the need of specific raster array functions in the language. To achieve this, first Coverage to RDF mapping solutions were defined, based on the well-known W3C standard mappings for relational data. Next, the SPARQL algebra is extended with a new operator that delegates part of the GeoSPARQL query in WCPS services. Query optimization replaces those parts of the SPARQL query plan that may be delegated to a WCPS service by instances of such new WCPS operator. A first prototype has been implemented by extending the ARQ SPARQL query engine of Apache Jena. Petascope was used as the WCPS implementation on top of the Rasdaman raster array database. An initial evaluation with real meteorological data shows, as it was initially expected, that the approach outperforms an existing reference relational database based GeoSPARQL implementation.

1. Introduction

Three main sources of geospatial big data may be identified, namely, volunteered geographic information (VGI), data generated by mobile devices, and data generated by earth observation and modelling infrastructures. The first two types of data have usually the form of vector features, i.e., data about geospatial entities. On the other hand, a large amount of the environmental data that is currently being generated comes from remote sensing devices and environmental modelling processes, and it has the form of very large raster coverages, i.e., very large numerical arrays with spatial and temporal dimensions.

The open data infrastructures used in the geospatial (Foley, 2009) and environmental domain (Nativi et al., 2015) provide data discovery facilities and data access mechanisms for vector features (Vretanos, 2010) and raster coverages (OGC, 2018; Baumann, 2009). Important to achieve interoperability in these infrastructures is the definition of standard interfaces and formats, like those proposed by the Open Geospatial Consortium (OGC)[1] and UNIDATA.[2] The use of these infrastructures is mainly restricted to experts in Geographic Information Systems (GIS) and to scientists of the environmental domain.

General purpose open data infrastructures, on the other hand, are based on semantic web and linked data (Bizer et al., 2009) standards defined by the World Wide Web Consortium (W3C).[3] Therefore, to enable geospatial and environmental data to be accessible through these infrastructures and used by ITC practitioners, both their data models (Resource Description Framework—RDF (Schreiber and Raimond, 2014) and Web Ontology Language—OWL (Hitzler et al., 2012)) and their data access interface (SPARQL Protocol and RDF Query Language Harris and Seaborne, 2013) must be extended with geospatial capabilities. Many applications could benefit from the available geospatial and environmental data if they were appropriately accessible through standard Linked Data technologies. One such example is tourism, where already existing linked data repositories might be combined with geospatial entity-based data of locations, hotels, restaurants, or site seeing and with meteorological predictions modelled as large spatiotemporal coverages. Queries such as "What is the predicted average of temperature of each municipality of Spain for the next week?" might be formulated and resolved by combining vector feature sources (municipality geometries), raster coverage sources (meteorological data), and linked data sources like DBPedia.

Geospatial vector features may already be represented and manipulated using a geospatial extension of SPARQL, called GeoSPARQL (Perry and Herring, 2012), proposed by the OGC. Efficient implementations of GeoSPARQL are also available (Kyzirakos et al., 2012), which leverage mature spatial database technologies. Raster coverage data access

* Corresponding author.
E-mail address: jrr.viqueira@usc.es (J.R.R. Viqueira).
1 https://www.ogc.org/.
2 https://www.unidata.ucar.edu/.
3 https://www.w3.org/.

https://doi.org/10.1016/j.cageo.2021.105023
Received 16 November 2020; Received in revised form 6 November 2021; Accepted 13 December 2021
Available online 4 January 2022

through linked data infrastructures is usually restricted to the querying of their vector-based metadata (Koubarakis et al., 2012). Some specific extensions (Andrejev et al., 2013; Homburg et al., 2020) have been designed to query raster data with SPARQL. However, their raster capabilities rely on collections of raster-specific functions defined on a raster data type, which demand specific training to use them, narrowing their use again to geographic and environmental data related experts.

In this paper, we present the design and first prototype implementation of a GeoSPARQL query processing approach for scientific raster array data, called GeoLD. Raster coverage data is mapped to RDF, without requiring any additional data type, using coverage to RDF mapping solutions, which are based on the already existing mapping approaches for relational data proposed by the W3C, namely direct mapping (Arenas et al., 2012) and R2RML (Das et al., 2012). Data querying is done with GeoSPARQL statements, without the need for any additional raster or array function or predicate. Therefore, it is expected that it may be used without any additional training. Enabling raster coverage access in GeoSPARQL automatically provides support for federated vector–raster geospatial querying in SPARQL. However, further work is still required to achieve an efficient implementation of such federated querying. The efficient access to raster coverage data is based on the incorporation of a new operator in the SPARQL query engine, which enables the delegation of part of the query in standard Web Coverage Processing Services (WCPS) (Baumann, 2009, 2010). As it was expected, the implementation outperforms the reference GeoSPARQL implementation (Kyzirakos et al., 2012). GeoLD is also more efficient than currently available SPARQL raster extensions such as SciSPARQL (Andrejev and Risch, 2012; Andrejev et al., 2013) and GeoSPARQL+ (Homburg et al., 2020) in the data access stage, despite not requiring specific array syntax in its language, due to the lack of efficient array data storage and access approaches in their current implementations.

The main contributions of this work are the following: (i) the definition of coverage data to RDF mapping solutions, based on W3C standards; (ii) the definition of a SPARQL query processing approach in the form of a new SPARQL operator that enables the delegation of raster data access in OGC standard WCPS services; and (iii) the definition of a query optimization strategy that identifies maximum SPARQL query trees to be replaced by WCPS operator instances.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the main components of the system architecture. Two approaches to map raster coverages to RDF are discussed in Section 4. The query GeoSPARQL query processing solution is explained in Section 5. Section 6 shows performance evaluation results, and Section 7 concludes the paper and outlines future work.

2. Related work

Discovering, accessing, and browsing are key functionalities that must be provided by open data infrastructures to enable data-driven decision making. Semantic web technologies play a key role in the enablement of those functionalities by supplying amongst others: (i) ground data representation models and languages, namely RDF (Schreiber and Raimond, 2014) and OWL (Hitzler et al., 2012); (ii) ontologies to represent different types of metadata, such as DCAT (Albertoni et al., 2020) for general dataset metadata, PROV-O (Belhajjamey et al., 2013) for provenance metadata, and DQV (Debattista et al., 2016) for quality metadata; (iii) a set of specifications and languages called SPARQL that include a query language (Harris and Seaborne, 2013), formats for query results, a syntax for federated queries over external data sources (Prud'hommeaux and Buil-Aranda, 2013), a transactional language to perform updates and HTTP based protocols for client-service communications; and (iv) languages to support the mapping between the RDF and the relational model (Das et al., 2012; Arenas et al., 2012).

Spatial Data Infrastructures (SDIs) (Foley, 2009) must also provide services and/or components to support data discovery, access, and browsing. ISO and OGC standards are commonly adopted to achieve interoperable SDIs. ISO 19115 (ISO, 2014) is used for geographic metadata representation and the OGC CSW (Nebert et al., 2016) service specification for metadata querying. Geospatial vector features are accessed through web services with WFS (Vretanos, 2010) interface, and their geometric properties are usually implemented following the Simple Feature Access (SFA) (Herring, 2011). Basic geospatial raster coverage data access support is given by the WCS (OGC, 2018) web service interface, and more complex queries with some declarative processing are supported by the Web Coverage Processing Service (WCPS) (Baumann, 2009, 2010). Two recent examples of projects that include amongst their objectives the development of Spatial Data Infrastructures are TRAFAIR (http://trafair.eu/) and RADAR-ON-RAIA (http://radaronraia.eu/), the former in the scope of urban scale air quality observation and prediction and the later for HF-Radar ocean data.

Challenges for the application of semantic technologies in the geospatial domain have already been identified (Patroumpas et al., 2014), and the importance of their incorporation in SDIs has already been stressed ten years ago (Janowicz et al., 2010). During the last years, many efforts were devoted to the creation of ontologies, and it is expected that new projects will focus more on their use to solve the problems that arise during data discovery and access (Narock and Wimmer, 2017). Examples of ontologies in the geospatial domain arise in many different application domains, including hydrology (Essawy et al., 2017), geology (Ma et al., 2011) or mining (Ma et al., 2010). More generic are the attempts to achieve geospatial provenance metadata ontologies (Ma et al., 2014; Jiang et al., 2018) and their use for the representation of image classification rules (Andrés et al., 2017). The Semantic Web for Earth and Environment Technology Ontology (SWEET) (Raskin and Pan, 2005), created by NASA, is an example of a general-purpose vocabulary in the environmental domain.

Initial solutions that apply semantic technologies to the data discovering stage have already been reported in the literature (Zhao et al., 2009; Yue et al., 2007; Stock et al., 2012; Athanasis et al., 2009; Gahegan et al., 2009). In general, geospatial metadata is represented and encoded with ontologies, and semantic technologies are next applied to address semantic matching with query specifications. Additionally, the semantic representation of the metadata enables a more precise semantic interpretation of the retrieved datasets. Regarding data access, the main advantage is that new possibilities are open for the resolution of semantic conflicts during data integration (Wang et al., 2018; Buccella et al., 2009; Lutz et al., 2009; Graybeal et al., 2012; Regueiro et al., 2015, 2017).

Recently, challenges related to the development of geospatial semantic technologies have been identified in the scope of the construction of smart environmental open data infrastructures (Viqueira et al., 2020). Regarding data representation and querying, RDF support for geospatial vector features and related query capabilities is already provided by the GeoSPARQL standard (Perry and Herring, 2012), which is implemented by some existing data management tools (Kyzirakos et al., 2012). The RDF Cube Vocabulary (Cyganiak and Reynolds, 2014) has been designed to represent multidimensional data in data warehouses; however, it is not adequate to efficiently support the sampling spatial and temporal dimensions of coverages. GeoSPARQL may be used to represent and query geospatial coverage metadata, but it is not suitable for their data (Koubarakis et al., 2012). Finally, to the best of our knowledge, only SciSPARQL (Andrejev and Risch, 2012; Andrejev et al., 2013) and GeoSPARQL+ (Homburg et al., 2020) aim at providing SPARQL support for scientific array data and raster coverages. Raster arrays are represented with new array or raster data types. Such a nested representation of the arrays into a conventional data model has also been adopted by spatial DBMSs, which show poor performance when they are compared with specialized raster engines. Besides, the user must know a set of array data operators, in addition to the underlying SPARQL syntax, to be able to perform

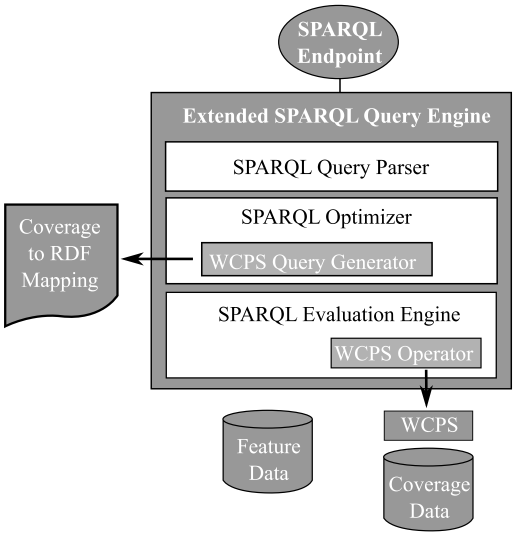


Fig. 1. Extended SPARQL query engine architecture.

queries. Regarding data integration and federated querying, already existing relational mappings (Das et al., 2012; Arenas et al., 2012) and relevant geospatial extensions may be used either to query directly spatial databases with SPARQL (Bereta et al., 2019) or to generate RDF from geospatial formats (Kyzirakos et al., 2014). Few geospatial federated query solutions have been proposed (Green et al., 2008), and they are also restricted to vector feature data.

3. System architecture

The architecture of GeoLD, the extended SPARQL query engine designed in the present work, is graphically shown in Fig. 1. As with any other query engine, first the query must be parsed to generate an appropriate tree of RDF algebra operations. Next, an optimizer chooses the best evaluation plan, i.e., a tree of operator algorithms that it is estimated to reach the best performance. Finally, the evaluation engine executes the query plan to generate the expected results from the input data. Few more details of each of the query engine layers are described below, focusing on the contributions of the present work with respect to already existing SPARQL query engines.

The *SPARQL Query Parser* is a conventional SPARQL parser that identifies the user defined functions and predicates specified in the GeoSPARQL standard (Perry and Herring, 2012). The output of this component is a tree whose nodes represent SPARQL algebra operators (Harris and Seaborne, 2013). Examples of these operators are the BGP, which generates an RDF triple (subject, predicate, object) sequence from a Basic Graph Pattern, and JOIN, which enables the combination of two sequences of triples.

The *SPARQL Optimizer* processes the SPARQL algebra tree generated by the parser to obtain an efficient query evaluation plan. An evaluation plan is also a tree, but now each node contains an algorithm that provides an implementation for a specific SPARQL algebra operator. A main contribution of the present work in this component is the implementation of an optimizer (WCPS Query Generator) that identifies parts of the evaluation plan that may be delegated to be executed by an underlying OGC WCPS service. Those parts are rewritten in the query plan with a new operator (*WCPS Operator*). To help to the identification of those tree parts and also to enable the mapping from the OGC raster coverage model to RDF, relevant mapping languages have been defined in this work. Further details are given in Section 4.

The *SPARQL Evaluation Engine* coordinates the execution of the algorithm of each node of the query plan to generate the expected output sequence of RDF triples. A main contribution of the present work with respect to already existing SPARQL query engines is the implementation of the new WCPS Operator. In the current implementation, this operator enables the querying of a WCPS service to perform spatial, temporal, or spatio-temporal filtering over an existing raster coverage, and to generate an output sequence of RDF triples, according to the specified coverage to RDF mapping. Further details related to the query rewriting process that generates WCPS operator nodes and to the implementation of this operator are given in Section 5.

4. Mapping geospatial raster coverages to RDF

The proposed solution for the definition of mappings between geospatial raster coverages and RDF triples is based on the already existing mappings languages defined by W3C between relational data and RDF, namely the direct mapping (Arenas et al., 2012) and R2RML (Das et al., 2012). A brief description of these relational to RDF mappings is first given. Let obs(sensor, time, rain) be the scheme of a relation that records time series of rainfall measurements generated by different sensors. Let also $(sensor, time)$ be the primary key. For each tuple (s, t, r) of obs, a direct mapping generates the following RDF triples:

<obs/sensor=s;time=t> rdf:type <obs>.
<obs/sensor=s;time=t> <obs#sensor> s.
<obs/sensor=s;time=t> <obs#time> t.
<obs/sensor=s;time=t> <obs#rain> r.

The generation of triples for foreign keys and tuples of relations without primary keys are omitted due to space limitations.

Contrary to the direct mappings, which use the relation and attribute names as part of the output RDF vocabulary, the R2RML language enables the user to control the vocabulary of the output. Each relation is mapped to RDF using rules called *TriplesMap*. Each rule has three parts: (i) a *LogicalTable* that may be either a table name, view name, or SQL statement, which defines the input tuples to be mapped; (ii) a *SubjectMap* that generates the triple that specifies the rdf:type of each tuple and the subject part of all the triples. Usually, IRIs are generated using templates where primary key attributes are used as parameters; (iii) a sequence of *PredicateObjectMap* that generates the predicate and object part of all the triples. Each *PredicateObjectMap* is further decomposed into two parts, one that enables the generation of the triple predicate and another that generates the triple object.

Geospatial coverages are conceptually modelled as collections of mappings. Each coverage field (or attribute) is modelled as a mapping, whose domain is the Cartesian Product of the coverage dimensions (a spatial dimension and optionally also a temporal one) and whose range is the data type of the field (in practice most of the raster coverages have fields of real types). Fig. 2(a) illustrates a spatio-temporal coverage called $Meteo$ that contains two fields of meteorological data, namely $Temp$ (Temperature) and Hum (Humidity). Based on the above conceptual modelling assumption, the proposed direct mapping approach for coverages generates, for each coverage raster cell, one triple to represent its type, and one triple for each field to represent their values. Fig. 2(b) shows the graphical representation of the triples generated for two of the raster cells of the $Meteo$ coverage. It is first noticed that $Meteo$ is defined as an subclass of *geo:Coverage*, which in turn is a subclass of the GeoSPARQL *geo:Feature* class. Each raster cell is defined as an instance of $Meteo$, with a geometric property (*hasLocation*), a temporal property (*hasTime*), and a property of a conventional data type for each coverage field. OGC Well Known Text (WKT) representation is used for geometric literals; therefore, the generated RDF is fully compatible with the GeoSPARQL ontology.

The Coverage to RDF Mapping Language (C2RML) gives more control to the user over the RDF generation process, enabling both slicing and trimming over the input coverage and also letting the user define its own RDF vocabulary. *TripleMap* rules of C2RML are similar to those of R2RML. The main difference is that R2RML *LogicalTable* is replaced by a new *Coverage* section, which specifies the name of the input coverage and optionally defines trimming and/or slicing operations over its

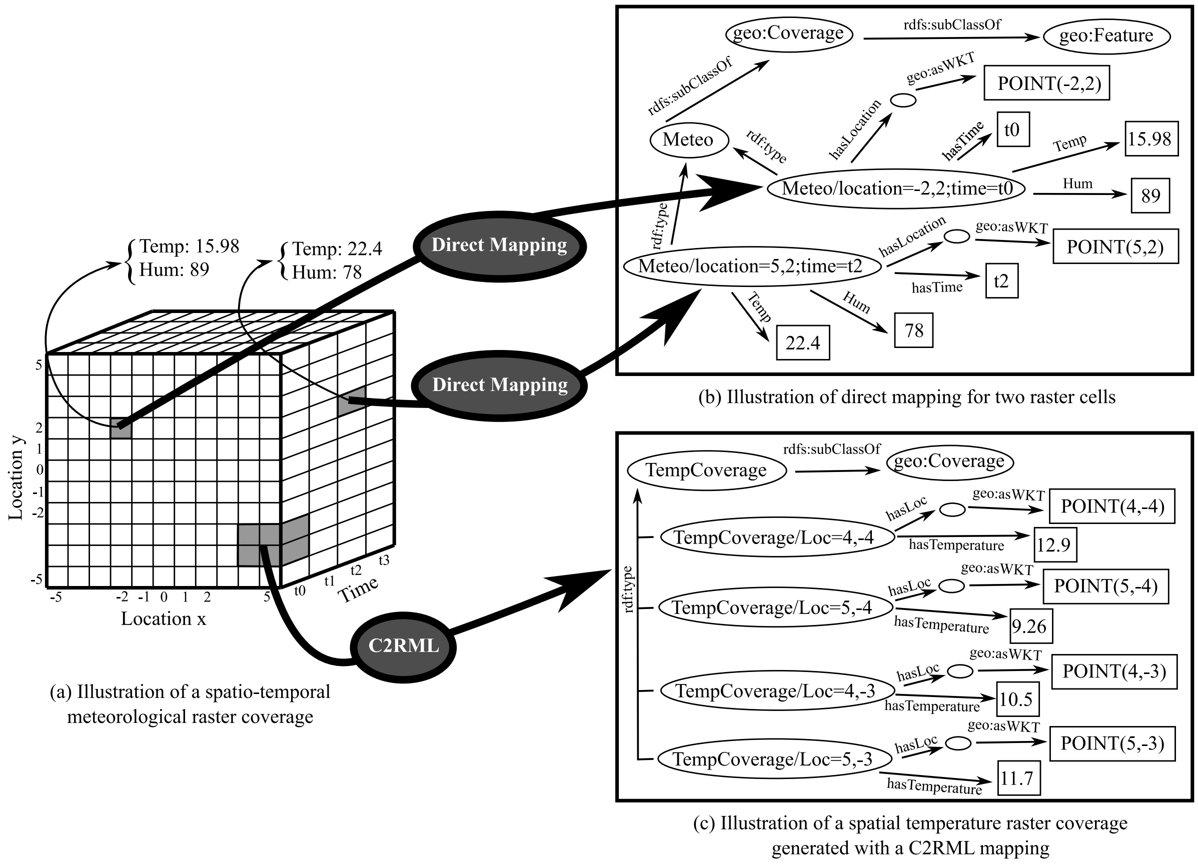


Fig. 2. Illustration of geospatial raster coverage to RDF mappings.

dimensions. Fig. 2(c) illustrates the RDF generated by a specific C2RML mapping applied to coverage $Meteo$. In this example, the *Coverage* section of the mapping defines a trimming over the spatial dimension of the coverage (*Location x* between 4 and 5 and *Location y* between −4 and −3). It also specifies a slicing over the temporal dimension (time = t0). To generate the subjects of all the RDF triples the *SubjectMap* uses the following template

$TempCoverage/Loc = \{Location\}$

where the parameter $Location$ is replaced by the coordinates of each raster cell. Finally, two *PredicateObjectMap* are used to generate the $hasLoc$ and $hasTemperature$ properties, from the coverage spatial dimension and the $Temp$ (Temperature) field, respectively.

5. Query processing

The description of the query processing solution that enables the efficient querying of raster coverages is described in the following subsections. Broadly speaking, first the query optimizer identifies those parts of the query execution plan that may be delegated to the underlying WCPS. Those parts of the query plan are replaced by calls to a new WCPS operator. In the current version, this new operator supports only filtering, including dimension trimming and slicing and field filtering. The current implementation is based on the combination of ARQ,[4] the SPARQL Processor of Apache Jena, with Petascope (Aiordăchioaie and Baumann, 2010), the OGC standards-based coverage geospatial server of the raster server rasdaman (Baumann et al., 1998).

[4] https://jena.apache.org/documentation/query/index.html.

5.1. SPARQL algebra

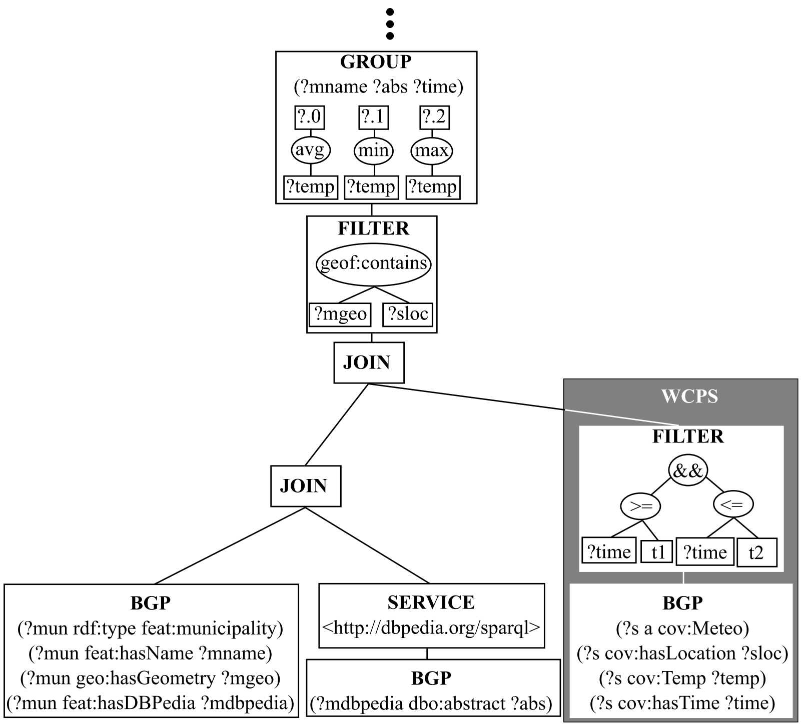
The SPARQL specification (Harris and Seaborne, 2013) includes the definition of an algebra for the evaluation of the queries. A full description of all the SPARQL algebra operations is out of the scope of this paper, thus the discussion will restrict to the most important ones required to evaluate the running example GeoSPARQL query of Fig. 3(a). This query combines data from three sources: (i) a geospatial feature collection *feat:municipality* that records data of municipalities, including their name, geometry and reference to their DBPedia node; (ii) DBPedia, which contains additional data of municipalities, including a textual description in property *dbo:abstract*; (iii) the raster coverage *cov:Meteo* illustrated in Fig. 2(a). For each municipality, the data of coverage *cov:Meteo* is queried, restricting the time dimension to values between $t1$ and $t2$, and restricting the spatial dimension to cells contained in the municipality geometry. The coverage temperature values are aggregated using functions average, minimum and maximum values, and combined with the name and the abstract of the municipality to generate the final result, i.e., aggregated meteorological data for each municipality.

Fig. 3(b) shows part of a possible query plan for the above GeoSPARQL query. The output of a SPARQL query, and also of a SPARQL operation of its algebra, is a sequence of solution mappings, where each solution mapping is a function that associates a variable of the query with an RDF term, i.e, either an RDF literal, a resource identifier (IRI) or a blank node. Operators that may appear as leaf nodes in query plans generate solution mappings from the input RDF graphs. Thus, for example, the BGP operator at the bottom left part of the Fig. 3(b), generates a sequence of solution mappings for variables $(?mun, ?mname, ?mgeo, ?mdbpedia)$, applying a list of triple patterns to the input RDF graph. On the other hand, all the other operators transform input solution mapping sequences into output sequences.

```
PREFIX
geo:
http://www.opengis.net/ont/geosparql#
PREFIX
geof:
<http://www.opengis.net/def/
function/geosparql/>
PREFIX
dbo:<http://dbpedia.org/ontology/>
PREFIX
feat:<http://www.mydomain.com/
features/>
PREFIX cov:<http://www.mydomain.com
/coverages/>

SELECT ?mname ?abs ?time
       (avg(?temp) as ?avgtemp)
       (min(?temp) as ?mintemp)
       (max(?temp) as ?maxtemp)
WHERE
 {?mun a feat:municipality;
      feat:hasName ?mname;
      geo:hasGeometry ?mgeo;
      feat:hasDBPedia ?mdbpedia.
 SERVICE <http://dbpedia.org/sparql>
  {?mdbpedia dbo:abstract ?abs.}
  ?s a cov:Meteo;
     cov:hasLocation ?sloc;
     cov:Temp ?temp;
     cov:hasTime ?time.
 FILTER (geof:contains(?mgeo,?sloc)
     && ?time >= t1 && ?time <= t2)
 }
GROUP BY ?mname ?abs ?time
```

(a) GeoSPARQL Query


GROUP
(?mname ?abs ?time)
?.0
?.1
?.2
avg
min
max
?temp
?temp
?temp
FILTER
geof:contains
?mgeo
?sloc
JOIN
WCPS
FILTER
&&
>=
<=
?time
t1
?time
t2
JOIN
BGP
(?mun rdf:type feat:municipality)
(?mun feat:hasName ?mname)
(?mun geo:hasGeometry ?mgeo)
(?mun feat:hasDBPedia ?mdbpedia)
SERVICE
<http://dbpedia.org/sparql>
BGP
(?mdbpedia dbo:abstract ?abs)
BGP
(?s a cov:Meteo)
(?s cov:hasLocation ?sloc)
(?s cov:Temp ?temp)
(?s cov:hasTime ?time)

(b) SPARQL Optimized Query Plan

Fig. 3. Example of a GeoSPARQL query and its corresponding SPARQL query plan.

Thus, for example, the first JOIN operator at the bottom left part of Fig. 3(b) combines the above sequence of solution mappings with the sequence generated for variables $(?mdbpedia, ?abs)$ by operators BGP and SERVICE. The FILTER operator selects the input solution mappings on which a given condition holds. Thus, for example, the filter operator at the bottom right part of the query plan of the figure restrict the solution mappings of variables $(?s, ?sloc, ?temp, ?time)$ to those where variable $?time$ has values between $t1$ and $t2$. Finally, the GROUP operator at the top of the Fig. 3(b), generates mappings for variables $(?name, ?abs, ?time, ?.0, ?.1, ?.2)$, where values of variables $(?.0, ?.1, ?.2)$ are computed by aggregating (through functions avg, min and max) values of the variable $?temp$ obtained from mappings with identical values for variables $(?name, ?abs, ?time)$.

5.2. WCPS operator

To enable the delegation of raster coverage querying on WCPS services (Baumann, 2009, 2010), a new leaf SPARQL operator was defined and implemented, namely, the *WCPS* operator. Broadly, this operator allows the execution of raster coverage processing statements on a WCPS service to generate solution mappings for variables representing coverage dimensions and coverage fields. Clearly, delegating part of the query in the data source moves the processing closer to the data, which is one of the basic principles to follow to achieve efficient query processing.

The WCPS coverage processing language (Baumann, 2010) is powerful enough to support the evaluation of different types of filters and aggregations on input coverages. The current implementation of the WCPS operator restricts to filtering, and therefore the description below restricts to WCPS coverage processing language expressions for filtering over dimensions and fields.

Dimension filtering is performed in WCPS with slicing and trimming operations. Thus, for example, the following expression retrieves a coverage of temperature values from coverage *Meteo* in Fig. 2(a), by trimming the spatial dimensions and by slicing the temporal dimension. The result is encoded in CSV format.

```
For $m in (Meteo)
Return encode($m.Temp[x(-3,2),y(1,4),time(t2)],
"CSV").
```

Spatial, temporal and spatio-temporal filter expressions are used to generate appropriate WCPS slicing and trimming expressions. Spatial and temporal coordinates used at SPARQL level must be adapted to the spatial and temporal resolution of the data source. Thus, a condition of the form "$geof: Equals(?sloc, p)$", where $?sloc$ is a variable referencing the spatial dimension of the coverage and p a point literal, must retrieve data despite the fact that p coordinates might not match exactly the coordinates of the centroid of any cell of the coverage. Besides, subsequent post-processing of the WCPS results might be needed in some cases. For example, the evaluation of a predicate "$geof: Contains(p, ?sloc)$", where p is a polygon literal, requires a spatial trimming over the minimum bounding rectangle (mbr) of p, done by the WCPS expression, and a post-processing that discards cells that lie inside the mbr but are not contained in p.

Filtering with conditions different from those of trimming and slicing over dimensions is not possible in WCPS, as it is also not possible to filter over coverage fields. However, it is possible to generate a special value (Not a Number—NaN) in cells where the condition does not hold, by using a "switch" statement. Those NaN cells are discarded by the WCPS operator after querying the WCPS service. Thus, for example, the following WCPS expression obtains a coverage with data only at areas with temperatures between 10 and 15.

```
For $m in (Meteo)
Return encode(switch
                 case $>= 10 and $m.Temp <=15
                    return $m.Temp
                 default return nan, "CSV")
```

5.3. Query optimization

The SPARQL query optimizer of ARQ transforms the initial query plan produced by the SPARQL query parser into another query plan that is expected to achieve a better performance. An example of a

Table 1
Notation used in the WCPS tagging algorithm pseudocode.

Notation	Description
$WCPSTagger(P)$	Returns a tagged version of query plan tree P.
$BGPTagger(P)$	P must be a BGP node. It creates a tagged query tree where triples related to a single coverage are isolated into a single BGP.
P	Query plan tree.
N	BGP leaf node.
$isbgp(P)$	Returns true if P is a BGP node.
$isFilter(P)$	Returns true if P is a Filter node.
$child(P)$	It returns the single child node of node P.
$setChild(P1, P2)$	It sets node $P2$ as the single child of node $P1$.
$setTag(P, t)$	It sets text t as the tag for node P.
$tag(P)$	It obtains the tag of node P.
$children(P)$	It returns the collection of child nodes of P.
$append(C, e)$	It appends element e to collection C.
$setChildren(P, C)$	It sets the collection of nodes C as the children of node P.
$createBGPS(N, C, s)$	It processes a BGP node N to create: (i) a collection C of BGP nodes, one for each coverage c referenced in a triple pattern of the form (?a rdf:type c) and (ii) a single BGP node s with all the remainder triple patterns of the form (?a rdf:type b).
$classifyTriples(N, C, s)$	It processes a BGP node N to classify its triple patterns. If a triple pattern is related to only one coverage, then it is added to the relevant BGP node in C, otherwise it is added to s.
$createJoin(C, s)$	It creates a query plan that joins the results of all the BGP nodes in collection C with the BGP node in BGP node s, using SPARQL SEQUENCE operations.

general heuristic applied during the optimization is to move the Filter operations as much as possible down in the query plan tree. As an example, the condition "$?time >= t1 \&\& ?time <= t2$" of the GeoSPARQL of Fig. 3(a) is placed after optimization in a FILTER operator immediately on top of the BGP operator of the bottom right part of Fig. 3(b). This way, the data corresponding to time instants out of the range [t1,t2] does not have to be processed by the JOIN operator, which leads to a more efficient query plan.

Once the above optimizations, already supported by ARQ, are applied, a new optimization step was added to enable the use of the WCPS operator at the appropriate place. To achieve this, maximum subtrees, starting from leaf nodes, must be identified, whose functionality may be replaced by a WCPS processing statement and, therefore, they may be replaced by a WCPS operator. In the current implementation, those maximum subtrees include only a BGP operator, with an optional FILTER on top of it.

Algorithm 1 WCPS tagging algorithm

```
1: function WCPSTagger($P$)
2:     if $isbgp(P)$ then return $BGPTagger(P)$
3:     else if $isFilter(P)$ then
4:         $TaggedChild = WCPSTagger(child(P))$
5:         $setChild(P, TaggedChild)$
6:         $setTag(P, tag(TaggedChild))$
7:         return $P$
8:     else
9:         $C = children(P)$
10:        for all $c \in C$ do
11:            $append(pchildren, WCPSTagger(c))$
12:        end for
13:        $setChildren(P, pchildren)$
14:        $setTag(P, \text{"SPARQL"})$
15:        return $P$
16:    end if
17: end function
18: function BGPTagger($N$)
19:     $createBGPS(N, cbgps, sbgp)$
20:     $classifyTriples(N, cbgps, sbgp)$
21:     return $createJoin(cbgps, sbgp)$
22: end function
```

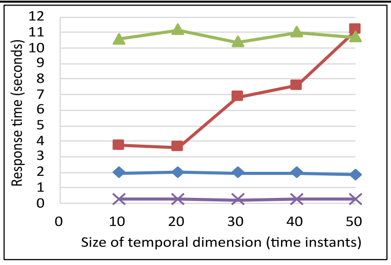
First, each node of the query plan tree is tagged with a string. Leaf BGP nodes that may be translated to a WCPS request are tagged with the name of the relevant raster coverage. If a BGP operator combines triple patterns corresponding to a raster coverage, with triple patterns corresponding to either other coverages or other data sources, then it has to be split to be tagged appropriately. The result of the generated BGPs has to be combined with a SEQUENCE operator, which is a special type of n-ary JOIN. The algorithm for this node tagging process is illustrated by the pseudocode of Algorithm 1, whose notation is described in Table 1. Function *WCPSTagger* recursively tags the nodes of the query plan tree as follows: (i) if the node is a leaf BGP then function *BGPTagger* is used to process it (line 2); (ii) if the node is a Filter (lines 3–7), then its child is recursively tagged using the same function and the Filter is tagged with the same tag of its child; (iii) if the node is of any other operator (lines 9–14), then all the children are tagged recursively, and the node is tagged as "SPARQL". Function *BGPTagger* processes BGP nodes as follows. First (line 17), the list of BGP triple patterns is processed to create a list of coverage BGPs ($cbgps$), i.e., BGPs corresponding to a single coverage, and a SPARQL BGP ($sbgp$), containing all the triple patterns that do not correspond to any coverage. The former are tagged with the name of the corresponding coverage and the latter are tagged as "SPARQL". The triple patterns with predicates *rdf:type* and the existing coverage to RDF mappings (see Section 4) are used to determine when a new coverage BGP has to be created. In a second step (line 18), all the remainder triple patterns are classified to be added to one of the created BGPs. Finally, all the created BGPs (in case more than one is created) are combined using a SEQUENCE operator (line 19), which is tagged as "SPARQL". At the end of this tagging process, the operator nodes that have to be used to generate a WCPS operator are tagged with the name of the corresponding coverage. As an example, the FILTER and BGP operators at the bottom right part of Fig. 3(b) are tagged with the name of coverage "Meteo".

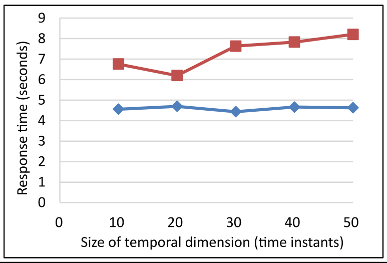
Once the query plan tree has been tagged, it has to be processed to insert WCPS operators at the required places. In particular, each BGP operator (or sequence of BGP and FILTER operators) tagged with the name of a coverage must be replaced by a WCPS operator which contains a WCPS processing statement that obtains the required raster coverage data. As an example, the FILTER and BGP operators that are the bottom right part of Fig. 3(b) will be replaced by a WCPS operator with the following processing statement:

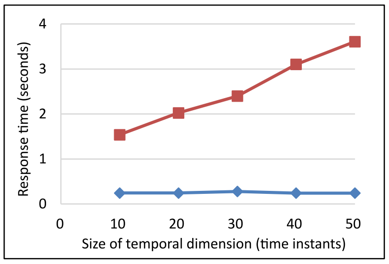
```
For $c in (Meteo)
Return encode($c.Temp[time(t1,t2)],"CSV").
```

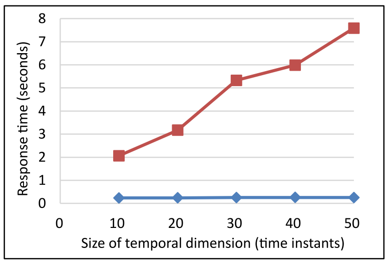
Additional optimizations considered as part of future work would require the implementation of specific spatial join and aggregation operators that leverage the underlying processing capabilities of vector and raster subsystems. As an example, in the query of Fig. 3, specific spatial join and aggregation operations could be used to combine the vector data of the municipalities (left part of the tree) and the raster coverage (right part of the tree), leveraging the spatial filtering and aggregation capabilities of the WCPS engine.

Query Type	SPARQL Expression	WCPS Expression	Response Time GeoLD ◆ STRABON ■ GeoSPARQL+ (Text) ▲ GeoSPARQL+ (Binary Hex.) ✕

Time Point Query	PREFIX cov:<http://www.mydomain.com/coverages/>
	SELECT ?temp
	WHERE
	{?s a cov:Meteo; cov:hasLocation ?sloc; cov:Temp ?temp;
	cov:hasTime ?time. FILTER (?time=t)}
		FOR c IN (Meteo)
		RETURN
		encode(c[E(*:*),N(*:*),
		ansi(t)],"csv")
			

Time Range Query	PREFIX cov:<http://www.mydomain.com/coverages/>
	SELECT ?temp
	WHERE
	{?s a cov:Meteo; cov:hasLocation ?sloc; cov:Temp ?temp;
	cov:hasTime ?time. FILTER (?time>=ts && ?time<=te)}
		FOR c IN (Meteo)
		RETURN
		encode(c[E(*:*),N(*:*),
		ansi(ts:te)],"csv")
			

Space Point Query	PREFIX cov:<http://www.mydomain.com/coverages/>
	PREFIX
	geof:<http://www.opengis.net/def/function/geosparql/>
	SELECT ?temp
	WHERE
	{?s a cov:Meteo; cov:hasLocation ?sloc; cov:Temp ?temp;
	cov:hasTime ?time. FILTER (geof:Equals(?sloc, p))}
		FOR c IN (Meteo)
		RETURN
		encode(c[E(px),N(py),
		ansi(*:*)],"csv")
			

Space Range Query	PREFIX cov:<http://www.mydomain.com/coverages/>
	PREFIX
	geof:<http://www.opengis.net/def/function/geosparql/>
	SELECT ?temp
	WHERE
	{?s a cov:Meteo; cov:hasLocation ?sloc; cov:Temp ?temp;
	cov:hasTime ?time. FILTER (geof:Constains(p, ?sloc))}
		FOR c IN (Meteo)
		RETURN
		encode(c[E(minpx:maxpx),
		N(minpy:maxpy),
		ansi(*:*)],"csv")
			

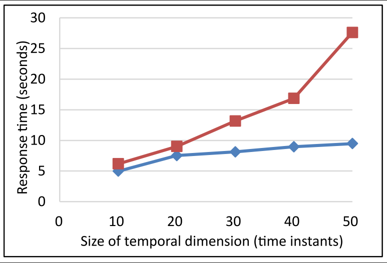
Field Range Query	PREFIX cov:<http://www.mydomain.com/coverages/>
	SELECT ?temp
	WHERE
	{?s a cov:Meteo; cov:hasLocation ?sloc; cov:Temp ?temp;
	cov:hasTime ?time. FILTER (?temp>=t1 && ?temp <= t2)}
		FOR c IN (Meteo)
		RETURN
		encode(
		SWITCH
		CASE
		c[E(*:*),N(*:*),
		ansi(*:*)]>= t1
		AND c[E(*:*),N(*:*),
		ansi(*:*)]<=t2 RETURN (c)
		DEFAULT RETURN nan,"csv")
			

Fig. 4. Performance evaluation scaling the temporal dimension.

6. Performance evaluation

To demonstrate the clear benefits in terms of performance that brings to GeoLD the delegation of raster data access in specialized technologies, an evaluation was undertaken that compared the response times of four different SPARQL implementations, namely, STRABON (Kyzirakos et al., 2012), SciSPARQL (Andrejev and Risch, 2012; Andrejev et al., 2013), GeoSPARQL+ (Homburg et al., 2020) and GeoLD.

All the software was installed in a 32 bit virtual machine with Ubuntu 16.04, since SciSPARQL was only available for 32 bit architectures. The hardware specification used by the virtual machine was the following: 4 processors Intel core i7 3.40 GHz and 8 GB of RAM. In general, five different types of queries were performed over all the datasets:

Space Point Query: It retrieves the time series of all the cell values (temperature values in our experiment) whose cells intersect the query spatial point, i.e., it performs a slicing over the spatial dimension of the coverage.

Time Point Query: It retrieves the spatial raster coverage valid at a query time instant, i.e., it performs a slicing over the temporal dimension of the coverage.

Space Range Query: It obtains the spatio-temporal coverage whose spatial extension is restricted to the query rectangle, i.e., it performs a trimming over the spatial dimension.

Time Range Query: It obtains the spatio-temporal coverage whose temporal extension is restricted to the query time period, i.e., it performs a trimming over the temporal dimension.

Field Range Query: It obtains the triples (space point, time instant, value) obtained from the input coverage such that the value (temperature value in our experiment) lies inside a given query range.

All the queries were evaluated with a cold cache system status, to avoid the undesirable effects in the response time of the buffer cache of the DBMS PostgreSQL used by STRABON, which is not present in none of the other solutions.

Two different evaluation experiments were done aiming at evaluating the scalability of the approach when either temporal or spatial dimensions of the datasets grow. The spatio-temporal coverages used to test scalability over the time dimension were generated from historical numeric weather predictions of MeteoGalicia,[5] the meteorological agency of the Spanish region of Galicia. Five coverages with time dimensions ranging from 10 to 50 time instants and space dimension of 117 × 142 pixels were used, whose overall size ranged from 664.k to 3.3M data elements (spatio-temporal cells). Those coverages were loaded in rasdaman to be used by GeoLD and also transformed to RDF to be loaded in the different SPARQL engines.

An overview of the achieved results is shown in Fig. 4. For each of the above query types, the charts provide the SPARQL syntax, the WCPS query generated by GeoLD for the WCPS operator, and the response times of the evaluated solutions for each database size (in number of time instants). All the queries could be implemented in GeoLD, STRABON and SciSPARQL. On the other hand, only time point queries, as defined above, could be implemented in the current proof of concept implementation of GeoSPARQL+. This was mainly due to the lack of appropriate implementation of raster data type functions to extract cell values from tiles. Spatial and temporal point and range queries, with fixed query selectivity (fixed temporal query period and spatial rectangle size) are solved in constant time (with respect to the database size) by GeoLD, as it is shown in the figure. This is due to the direct access to the data elements through array dimensions provided by the rasdaman raster storage structures, which is invoked by the GeoLD WCPS operator. Furthermore, as it was expected, field range queries increase with the database size. Response times of SciSPARQL are two orders of magnitude worse than those of GeoLD. As an example, for a database size of 20 time instants, response times range from 110 to 23 s, depending on the query type. For this reason, those results are not shown in the Figure. Regarding GeoSPARQL+, for time point queries, two different encodings were tested for the output raster coverage tiles. The use of a text encoding of the result tiles gives a performance clearly worse than GeoLD, and even worse than STRABON for small coverage sizes. On the other hand, using a more compact raster representation (well-known-binary hexadecimal in our experiment) enables a much more efficient encoding of the result. This shows the importance of the incorporation of specific efficient compact encodings to be used when the result of the query is a coverage. The design and implementation of such compact raster representations is part of future work in GeoLD. All the other query types could not be implemented, however, the evaluation of a query that retrieves the time instants of raster tiles that intersect a given query point showed response times of around 4 s, which is worse than GeoLD and also worse than STRABON, despite not having to access the raster tiles to obtain the cell values. Finally, regarding STRABON, the spatial relational DBMS-based data storage layer enables the use of specific indexing structures, such as R-Trees for 2D spatial dimensions and B+-trees for 1D dimensions. In spite of this, as it is also shown in the figure, these structures do not enable achieving a performance with raster data similar to that of a specialized raster engine.

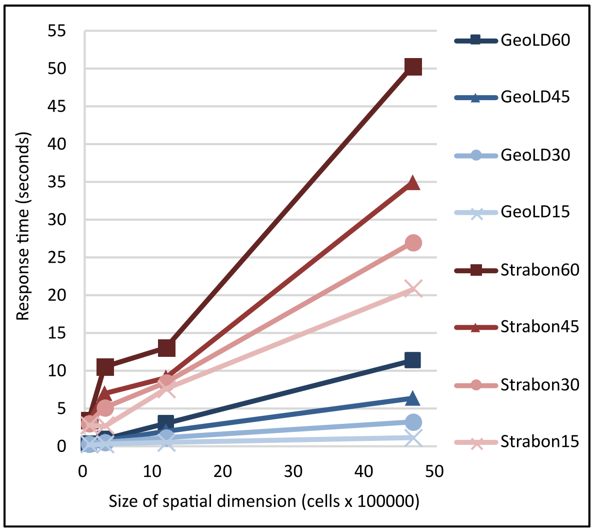


Fig. 5. Performance evaluation of spatial range queries scaling the spatial dimension.

The scalability over the space dimension was tested using coverages generated from a Digital Elevation Model (DEM) of Galicia with 100 metres of spatial resolution. In particular, four datasets were used with resolutions of 800, 400, 400 and 100 m, and with sizes (in number of data elements) ranging from around 72 k (800 m resolution) to 4.6 M (100 m resolution). Various space range queries using query squares of increasing sizes (ranging from 15 km to 60 km of side length) were evaluated on GeoLD and STRABON. The results are shown in Fig. 5. Clearly, GeoLD outperforms STRABON in all cases, due to the better performance provided by rasdaman for raster data access.

7. Conclusion

The design and first GeoSPARQL query engine for scientific raster array data, called GeoLD, was briefly described. The system analyses optimized SPARQL algebra trees, generated by state of the art technologies, and identifies maximum parts that may be evaluated by a standard WCPS service. To achieve this, Coverage to RDF mapping solutions (both direct mapping and C2RML) were defined, which are based on the well-known W3C standard used for relational data. A first prototype of the system was implemented as an extension of the ARQ SPARQL engine of Apache Jena. The Rasdaman array database was used to record the raster data and its WCPS implementation Petascope was used to access it through the web. GeoLD enables accessing scientific raster data with conventional linked data technologies, opening the possibility to incorporate scientific datasets to conventional applications with little effort. As it was initially expected, both from these authors intuition and other authors asserts, the solution outperforms an already existing GeoSPARQL implementation, which is based on relational technology. Besides, it does not require specific array extensions, as SciSPARQL and GeoSPARQL+ do. Currently, GeoLD outperforms also SciSPARQL and GeoSPARQL+ in the data access stage, due to their lack of specific raster data storage and access technologies. GeoSPARQL+ however, shows the importance of providing specific raster data encodings to be used when raster coverages are retrieved. Future work is related to (i) completing the generation of WCPS queries for other SPARQL operations (beyond BGP and FILTER), for example aggregates, (ii) the efficient implementation of spatial JOIN operations between vector

[5] https://www.meteogalicia.gal/modelos.

and raster datasets and (iii) the incorporation of mechanisms to use standard efficient encodings for raster coverages (such as GeoTIFF and NetCDF) in query results.

CRediT authorship contribution statement

Shahed Bassam Almobydeen: Study of the state of the art, design and implementation of the solution, Preparation of the first draft of the paper. **José R.R. Viqueira:** Supervision of the state of the art study, design of the solution and of its evaluation, Revision of the paper. **Manuel Lama:** Design of the solution and of its evaluation, Revision of the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The code is available through CiTIUS GitLab at the following URL: https://gitlab.citius.usc.es/shahed.almobydeen/extended-jena-arq

Acknowledgements

The work of Shahed Bassam Almobydeen was partially funded by European Union under the Erasmus Mundus Peace II mobility program. The work of José R.R. Viqueira was partially funded by Xunta de Galicia, Spain under the Project ED431B 2021/16, by the TRAFAIR EU project 2017-EU-IA-0167, co-financed by the Connecting Europe Facility, by the EU RADAR-ON-RAIA project (0461_RADAR_ON_RAIA_1_E), co-financed by the European Regional Development Fund (ERDF) through the Iterreg V-A Spain-Portugal program (POCTEP) 2014–2020 and by project MAGIST-ELA PID2019-105221RB-C42, funded by Spanish Ministry of Economy and Competitiveness, Spain. The work of Manuel Lama was partially funded by the Spanish Ministry for Science, Innovation and Universities under the project TIN2017-84796-C2-1-R.

References

Aiordăchioaie, A., Baumann, P., 2010. Petascope: An open-source implementation of the OGC WCS Geo service standards suite. In: International Conference on Scientific and Statistical Database Management. Springer, pp. 160–168.

Albertoni, R., Browning, D., Cox, S., Beltran, A.G., Perego, A., Winstanley, P., 2020. Data Catalog Vocabulary (DCAT) - Version 2. W3C Recommendation 04 February 2020. https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/. Online; accessed July-2020.

Andrejev, A., Risch, T., 2012. Scientific SPARQL: semantic web queries over scientific data. In: 2012 IEEE 28th International Conference on Data Engineering Workshops. IEEE, pp. 5–10.

Andrejev, A., Toor, S., Hellander, A., Holmgren, S., Risch, T., 2013. Scientific analysis by queries in extended SPARQL over a scalable e-science data store. In: 2013 IEEE 9th International Conference on E-Science. IEEE, pp. 98–106.

Andrés, S., Arvor, D., Mougenot, I., Libourel, T., Durieux, L., 2017. Ontology-based classification of remote sensing images using spectral rules. Comput. Geosci. 102, 158–166. http://dx.doi.org/10.1016/j.cageo.2017.02.018.

Arenas, M., Bertails, A., Prud'hommeaux, E., Sequeda, J., 2012. A Direct Mapping of Relational Data to RDF. W3C Recommendation 27 September 2012. https://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/. Online; accessed July-2020.

Athanasis, N., Kalabokidis, K., Vaitis, M., Soulakellis, N., 2009. Towards a semantics-based approach in the development of geographic portals. Comput. Geosci. 35 (2), 301–308. http://dx.doi.org/10.1016/j.cageo.2008.01.014.

Baumann, P., 2009. Web Coverage Processing Service (WCPS) Language Interface Standard. https://www.ogc.org/standards/wcps. Online; accessed July-2020.

Baumann, P., 2010. The OGC web coverage processing service (WCPS) standard. Geoinformatica (14), 447–479. http://dx.doi.org/10.1007/s10707-009-0087-2.

Baumann, P., Dehmel, A., Furtado, P., Ritsch, R., Widmann, N., 1998. The multidimensional database system RasDaMan. In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. pp. 575–577.

Belhajjamey, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J., 2013. PROV-O: The PROV Ontology. W3C Recommendation 30 April 2013. https://www.w3.org/TR/2013/REC-prov-o-20130430/. Online; accessed July-2020.

Bereta, K., Xiao, G., Koubarakis, M., 2019. Ontop-spatial: Ontop of geospatial databases. J. Web Semant. 58, 100514. http://dx.doi.org/10.1016/j.websem.2019.100514, URL: https://www.sciencedirect.com/science/article/pii/S1570826819300447.

Bizer, C., Heath, T., Berners-Lee, T., 2009. Linked data - the story so far. Int. J. Semant. Web Inf. Syst. 5 (3), 1–22. http://dx.doi.org/10.4018/jswis.2009081901.

Buccella, A., Cechich, A., Fillottrani, P., 2009. Ontology-driven geographic information integration: A survey of current approaches. Comput. Geosci. 35 (4), 710–723. http://dx.doi.org/10.1016/j.cageo.2008.02.033, Geoscience Knowledge Representation in Cyberinfrastructure.

Cyganiak, R., Reynolds, D., 2014. The RDF Data Cube Vocabulary. W3C Recommendation 16 January 2014. https://www.w3.org/TR/vocab-data-cube/. Online; accessed July-2020.

Das, S., Sundara, S., Cyganiak, R., 2012. R2RML: RDB to RDF Mapping Language. W3C Recommendation 27 September 2012. https://www.w3.org/TR/2012/REC-r2rml-20120927/. Online; accessed July-2020.

Debattista, J., Dekkers, M., Guéret, C., Lee, D., Mihindukulasooriya, N., Zaveri, A., 2016. Data on the Web Best Practices: Data Quality Vocabulary. W3C Working Group Note 15 December 2016. https://www.w3.org/TR/2016/NOTE-vocab-dqv-20161215/. Online; accessed July-2020.

Essawy, B.T., Goodall, J.L., Xu, H., Gil, Y., 2017. Evaluation of the OntoSoft ontology for describing metadata for legacy hydrologic modeling software. Environ. Model. Softw. 92, 317–329. http://dx.doi.org/10.1016/j.envsoft.2017.01.024.

Foley, R., 2009. Integrated spatial data infrastructure. In: Kitchin, R., Thrift, N. (Eds.), International Encyclopedia of Human Geography. Elsevier, Oxford, pp. 507–511. http://dx.doi.org/10.1016/B978-008044910-4.00039-0, URL: http://www.sciencedirect.com/science/article/pii/B9780080449104000390.

Gahegan, M., Luo, J., Weaver, S.D., Pike, W., Banchuen, T., 2009. Connecting GEON: Making sense of the myriad resources, researchers and concepts that comprise a geoscience cyberinfrastructure. Comput. Geosci. 35 (4), 836–854. http://dx.doi.org/10.1016/j.cageo.2008.09.006.

Graybeal, J., Isenor, A.W., Rueda, C., 2012. Semantic mediation of vocabularies for ocean observing systems. Comput. Geosci. 40, 120–131. http://dx.doi.org/10.1016/j.cageo.2011.08.002.

Green, J., Dolbear, C., Hart, G., Goodwin, J., Engelbrecht, P., 2008. Creating a semantic integration system using spatial data. In: Proceedings of the 2007 International Conference on Posters and Demonstrations-Volume 401. CEUR-WS. org, pp. 70–71.

Harris, S., Seaborne, A., 2013. SPARQL 1.1 Query Language. W3C Recommendation 21 March 2013. https://www.w3.org/TR/2013/REC-sparql11-query-20130321/. Online; accessed July-2020.

Herring, J., 2011. OpenGIS Implementation Standard for Geographic Information - Simple Feature Access - Part 1: Common Architecture. https://www.ogc.org/standards/sfa. Online; accessed July-2020.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S., 2012. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation 11 December 2012. https://www.w3.org/TR/2012/REC-owl2-primer-20121211/. Online; accessed July-2020.

Homburg, T., Staab, S., Janke, D., 2020. GeoSPARQL+: Syntax, semantics and system for integrated querying of graph, raster and vector data. In: Pan, J.Z., Tamma, V., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L. (Eds.), The Semantic Web – ISWC 2020. Springer International Publishing, Cham, pp. 258–275.

ISO, 2014. ISO 19115-1:2014. Geographic Information — Metadata — Part 1: Fundamentals. https://www.iso.org/standard/53798.html. Online; accessed July-2020.

Janowicz, K., Schade, S., Bröring, A., Keßler, C., Maué, P., Stasch, C., 2010. Semantic enablement for spatial data infrastructures. Trans. GIS 14 (2), 111–129. http://dx.doi.org/10.1111/j.1467-9671.2010.01186.x.

Jiang, L., Yue, P., Kuhn, W., Zhang, C., Yu, C., Guo, X., 2018. Advancing interoperability of geospatial data provenance on the web: Gap analysis and strategies. Comput. Geosci. 117, 21–31. http://dx.doi.org/10.1016/j.cageo.2018.05.001.

Koubarakis, M., Datcu, M., Kontoes, C., Di Giammatteo, U., Manegold, S., Klien, E., 2012. TELEIOS: A database-powered virtual earth observatory. Proc. VLDB Endow. 5 (12), 2010–2013. http://dx.doi.org/10.14778/2367502.2367560.

Kyzirakos, K., Karpathiotakis, M., Koubarakis, M., 2012. Strabon: a semantic geospatial DBMS. In: International Semantic Web Conference. Springer, pp. 295–311.

Kyzirakos, K., Vlachopoulos, I., Savva, D., Manegold, S., Koubarakis, M., 2014. GeoTriples: A tool for publishing geospatial data as RDF graphs using R2RML mappings. In: Proceedings of the 2014 International Conference on Posters & Demonstrations Track - Volume 1272. In: ISWC-PD'14, CEUR-WS.org, Aachen, DEU, pp. 393–396.

Lutz, M., Sprado, J., Klien, E., Schubert, C., Christ, I., 2009. Overcoming semantic heterogeneity in spatial data infrastructures. Comput. Geosci. 35 (4), 739–752. http://dx.doi.org/10.1016/j.cageo.2007.09.017.

Ma, X., Carranza, E.J.M., Wu, C., van der Meer, F.D., Liu, G., 2011. A SKOS-based multilingual thesaurus of geological time scale for interoperability of online geological maps. Comput. Geosci. 37 (10), 1602–1615. http://dx.doi.org/10.1016/j.cageo.2011.02.011.

Ma, X., Wu, C., Carranza, E.J.M., Schetselaar, E.M., van der Meer, F.D., Liu, G., Wang, X., Zhang, X., 2010. Development of a controlled vocabulary for semantic interoperability of mineral exploration geodata for mining projects. Comput. Geosci. 36 (12), 1512–1522. http://dx.doi.org/10.1016/j.cageo.2010.05.014.

Ma, X., Zheng, J.G., Goldstein, J.C., Zednik, S., Fu, L., Duggan, B., Aulenbach, S.M., West, P., Tilmes, C., Fox, P., 2014. Ontology engineering in provenance enablement for the National Climate Assessment. Environ. Model. Softw. 61, 191–205. http://dx.doi.org/10.1016/j.envsoft.2014.08.002.

Narock, T., Wimmer, H., 2017. Linked data scientometrics in semantic e-science. Comput. Geosci. 100, 87–93. http://dx.doi.org/10.1016/j.cageo.2016.12.008.

Nativi, S., Mazzetti, P., Santoro, M., Papeschi, F., Craglia, M., Ochiai, O., 2015. Big data challenges in building the global earth observation system of systems. Environ. Model. Softw. 68, 1–26. http://dx.doi.org/10.1016/j.envsoft.2015.01.017.

Nebert, D., Voges, U., Bigagli, L., 2016. OGC Catalogue Services 3.0 - General Model. http://docs.opengeospatial.org/is/12-168r6/12-168r6.html. Online; accessed July-2020.

OGC, 2018. OGC Web Coverage Service (WCS) 2.1 Interface Standard - Core. http://docs.opengeospatial.org/is/17-089r1/17-089r1.html. Online; accessed July-2020.

Patroumpas, K., Giannopoulos, G., Athanasiou, S., 2014. Towards GeoSpatial semantic data management: strengths, weaknesses, and challenges ahead. In: Huang, Y., Schneider, M., Gertz, M., Krumm, J., Sankaranarayanan, J. (Eds.), Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014. ACM, pp. 301–310. http://dx.doi.org/10.1145/2666310.2666410.

Perry, M., Herring, J., 2012. OGC GeoSPARQL - A Geographic Query Language for RDF Data. https://www.ogc.org/standards/geosparql. Online; accessed July-2020.

Prud'hommeaux, E., Buil-Aranda, C., 2013. SPARQL 1.1 Federated Query. W3C Recommendation 21 March 2013. https://www.w3.org/TR/2013/REC-sparql11-federated-query-20130321/. Online; accessed July-2020.

Raskin, R.G., Pan, M.J., 2005. Knowledge representation in the semantic web for Earth and environmental terminology (SWEET). Comput. Geosci. 31 (9), 1119–1125. http://dx.doi.org/10.1016/j.cageo.2004.12.004.

Regueiro, M.A., Viqueira, J.R., Stasch, C., Taboada, J.A., 2017. Semantic mediation of observation datasets through Sensor Observation Services. Future Gener. Comput. Syst. 67, 47–56. http://dx.doi.org/10.1016/j.future.2016.08.013.

Regueiro, M.A., Viqueira, J.R., Taboada, J.A., Cotos, J.M., 2015. Virtual integration of sensor observation data. Comput. Geosci. 81, 12–19. http://dx.doi.org/10.1016/j.cageo.2015.04.006.

Schreiber, G., Raimond, Y., 2014. RDF 1.1 Primer. W3C Working Group Note 24 June 2014. https://www.w3.org/TR/rdf11-primer/. Online; accessed July-2020.

Stock, K., Stojanovic, T., Reitsma, F., Ou, Y., Bishr, M., Ortmann, J., Robertson, A., 2012. To ontologise or not to ontologise: An information model for a geospatial knowledge infrastructure. Comput. Geosci. 45, 98–108. http://dx.doi.org/10.1016/j.cageo.2011.10.021.

Viqueira, J.R.R., Villarroya, S., Mera, D., Taboada, J.A., 2020. Smart environmental data infrastructures: Bridging the gap between earth sciences and citizens. Appl. Sci. 10 (3), http://dx.doi.org/10.3390/app10030856.

Vretanos, P., 2010. OpenGIS Web Feature Service 2.0 Interface Standard. https://www.ogc.org/standards/wfs. Also ISO 19142. Online; accessed July-2020.

Wang, C., Ma, X., Chen, J., 2018. Ontology-driven data integration and visualization for exploring regional geologic time and paleontological information. Comput. Geosci. 115, 12–19. http://dx.doi.org/10.1016/j.cageo.2018.03.004.

Yue, P., Di, L., Yang, W., Yu, G., Zhao, P., 2007. Semantics-based automatic composition of geospatial Web service chains. Comput. Geosci. 33 (5), 649–665. http://dx.doi.org/10.1016/j.cageo.2006.09.003.

Zhao, P., Di, L., Yu, G., Yue, P., Wei, Y., Yang, W., 2009. Semantic Web-based geospatial knowledge transformation. Comput. Geosci. 35 (4), 798–808. http://dx.doi.org/10.1016/j.cageo.2008.03.013.