Making the Web of Data Available Via Web Feature Services

Jim Jones, Werner Kuhn, Carsten Keßler and Simon Scheider

Abstract Interoperability is the main challenge on the way to efficiently find and access spatial data on the web. Significant contributions regarding interoperability have been made by the Open Geospatial Consortium (OGC), where web service standards to publish and download spatial data have been established. The OGCs GeoSPARQL specification targets spatial data on the Web as Linked Open Data (LOD) by providing a comprehensive vocabulary for annotation and querying. While OGC web service standards are widely implemented in Geographic Information Systems (GIS) and offer a seamless service infrastructure, the LOD approach offers structured techniques to interlink and semantically describe spatial information. It is currently not possible to use LOD as a data source for OGC web services. In this chapter we make a suggestion for technically linking OGC web services and LOD as a data source, and we explore and discuss its benefits. We describe and test an adapter that enables access to geographic LOD datasets from within OGC Web Feature Service (WFS), enabling most current GIS to access the Web of Data. We discuss performance tests by comparing the proposed adapter to a reference WFS implementation.

J. Jones · S. Scheider
Institute for Geoinformatics, University of Münster, Münster, Germany
e-mail: jim.jones@uni-muenster.de

S. Scheider
e-mail: simon.scheider@uni-muenster.de

W. Kuhn
Center for Spatial Studies University of California, Santa Barbara, USA
e-mail: kuhn@geog.ucsb.edu

C. Keßler (✉)
CARSI, Department of Geography, Hunter College, City University of New York,
New York, USA
e-mail: carsten.kessler@hunter.cuny.edu

J. Huerta et al. (eds.), *Connecting a Digital Europe Through Location and Place*,
Lecture Notes in Geoinformation and Cartography, DOI: 10.1007/978-3-319-03611-3_20,

1 Introduction

Linked Open Data (LOD) is an approach for creating typed links between data from different sources in the Web. These typed links are based on objects, which have their meaning explicitly defined by terms in shared LOD vocabularies (Heath and Bizer 2011). With the advent of LOD vocabularies, these objects and their links can be built in a machine-readable way, enabling computers to perform queries and reasoning on datasets. The LOD approach is based on the Linked Data Principles,[1] which define essential steps for publishing data in the Web and for making it part of a single global dataset (Bizer et al. 2009). These principles help to enable interoperability and discoverability of datasets, creating a rich network of information. Due to these characteristics, LOD has become a key solution when it comes to efficiently publishing data on the Web.[2]

The LOD cloud is growing very rapidly, and some of its most important central hubs contain vast amounts of geographic information. The DBPedia initiative,[3] for example, systematically extracts information from Wikipedia,[4] publishes it as LOD and links it to other datasets (Auer et al. 2007). Part of this information is a geocoordinate for every localizable phenomenon described in Wikipedia. Successful efforts on implementing geographic LOD have also been carried out by government agencies, such as the Ordnance Survey of Great Britain,[5] which contributes significantly to the growth of the Web of geographic LOD based datasets (Goodwin et al. 2008).

Despite the benefits and efforts around LOD and also its inarguably increasing acceptance, the specific requirements of publishing geographic information on the Web have been addressed by standardized web services so far. An example is the Web Feature Service (WFS), a standard for providing geographic features on the Web, widely implemented in most Geographic Information Systems (GIS), but not supporting LOD. Despite their difference, both techniques, LOD and geographic web services, have their specific benefits and shortcomings for publishing and accessing geographic information on the Web. It has been argued before that combining both worlds has a great potential for boosting accessibility and interoperability of geographic information (Janowicz et al. 2010). For example, making Linked Open Data available in a geo service standard will turn all geo-service compatible GIS tools, whether they consist of simple desktop clients or distributed service implementations, into powerful geographic analysis tools of the LOD cloud. This combines the strengths of spatial data manipulation in a GIS with the potential of accessing datasets that are interlinked in the Web of Data.

This chapter addresses one of the open challenges for reaching this goal. We propose a way to efficiently access geographic LOD datasets via WFS. The main

[1] http://www.w3.org/DesignIssues/LinkedData.html

[2] http://lod-cloud.net

[3] http://dbpedia.org/About

[4] http://www.wikipedia.org

[5] http://www.ordnancesurvey.co.uk

idea is to use current Geographic Information Service standards and re-implement them in order to consume geographic LOD datasets published on the Web. The remainder of the chapter is structured as follows: Sect. 2 gives an overview of Linked Geographic Data, showing how it is described in different vocabularies. Section 3 describes the Web Feature Service standard, and explores its capabilities through its standard operations. Section 4 outlines the requirements and introduces our solution. Section 5 evaluates the performance of our implementation against the WFS reference implementation. Section 6 reviews related work, followed by conclusions and an outlook on future work in Sect. 7.

2 Linked Geographic Data

LOD datasets are described using the Resource Description Framework[6] (RDF), specified by the World Wide Web Consortium (Brickley and Guha 2004). RDF is a technology for describing resources and their interrelations in subject-predicate-object form. These so-called RDF Triples are commonly stored using an optimized storage and retrieval technology called Triple Store. Most Triple Stores organize RDF Triples in sub-sets called *Named Graphs*.[7] Named Graphs aggregate data, so that, for example, RDF Triples from distinct sources can be easily identified.

There have been several efforts to use LOD with geographic data. Suggestions include vocabularies for describing geographic data, together with storage and query techniques (Battle and Kolas 2011). Among the existing vocabularies for describing geographic LOD datasets is the Basic Geo Vocabulary[8] (WGS84 lat/long), which provides a namespace for describing geographic entities by coordinates pairs. This vocabulary is thus limited to points using WGS84 as a geodetic reference datum. Listing 1 shows an example using the WGS84 Vocabulary.

Listing 1 An example of a feature described with the WGS84 lat/long Vocabulary.

```
@prefix wgs84_pos: <www.w3.org/2003/01/geo/wgs84_pos#>.
@prefix my: <http://ifgi.lod4wfs.de/resource/>.
@prefix gn: <http://www.geonames.org/ontology#>.

my:GEOMETRY_1 a gn:Feature ;
       wgs84_pos:lat  "1.71389" ;
       wgs84_pos:long "69.3857" .
```

[6] http://www.w3.org/RDF/

[7] http://www.w3.org/TR/rdf11-concepts/#section-dataset

[8] http://www.w3.org/2003/01/geo/

An alternative to describe geographic LOD is the GeoSPARQL Vocabulary,[9] defined by the Open Geospatial Consortium[10] (OGC). It offers not only classes and properties for describing geographic LOD, but also spatial relations for querying geographic datasets (e.g. intersects, touches, overlaps, etc.). Listing 2 shows an example of a geographic LOD dataset using the GeoSPARQL Vocabulary, with the same point as in Listing 1. Geometries are defined by the class `Geometry` and the coordinates can be encoded in an RDF literal of type Well Know Text (WKT) using a single RDF property, namely `asWKT`.

Listing 2 An example of a feature described with the GeoSPARQL Vocabulary.

```
@prefix geo: <http://www.opengis.net/ont/geosparql/1.0#>.
@prefix my:  <http://ifgi.lod4wfs.de/resource/>.
@prefix sf:  <http://www.opengis.net/ont/sf#>.

my:GEOMETRY_1 a geo:Geometry ;
      geo:asWKT "POINT (-69.3857 1.71389)"^^sf:wktLiteral .
```

Due to the use of WKT literals, which correspond to OGC simple features (Herring 2011), GeoSPARQL enables an efficient way to describe many different kinds of geometry (e.g. polygons, lines, points, multipoint, etc.). Another important aspect of the GeoSPARQL vocabulary is the flexibility regarding coordinate reference systems. The latter are encoded as a literal type. This enables the use of many different coordinate reference systems by adding their corresponding URI to the WKT literal (see Listing 3). If no specific reference system is provided in the WKT literal, the WGS84 Longitude-Latitude[11] reference system is assumed by default.

Listing 3 An example of a feature described with the GeoSPARQL Vocabulary stating a specific Coordinate Reference System.

```
@prefix geo: <http://www.opengis.net/ont/geosparql/1.0#>.
@prefix my: <http://ifgi.lod4wfs.de/resource/>.
@prefix sf: <http://www.opengis.net/ont/sf#>.

my:GEOMETRY_1 a geo:Geometry ;
       geo:asWKT "<http://www.opengis.net/def/crs/EPSG/0/4326>
       POINT (-69.3857 1.71389)"^^sf:wktLiteral .
```

GeoSPARQL also offers the possibility to use the Geography Markup Language (GML) to encode geometries. In this case, the data type (`GMLLiteral`), property (`asGML`) and the URL for the geometry type (e.g. `http://www.opengis.net/def/gml/ Polygon`) have to be changed accordingly.

[9] http://www.opengis.net/doc/IS/geosparql/1.0

[10] http://www.opengeospatial.org/

[11] http://www.opengis.net/def/crs/OGC/1.3/CRS84

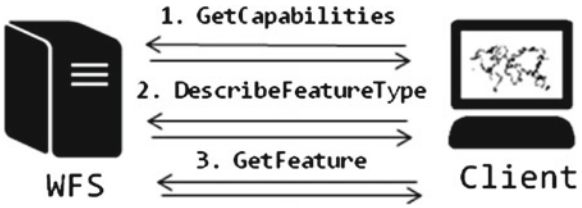

Fig. 1 Web feature service standard operations overview

3 Web Feature Service

The Web Feature Service[12] (WFS) is a platform-independent web service standard for vector-based geographic feature requests on the Web, defined by the OGC. A feature contains one or many geometries, optionally with attribute values. Its communication interface is established by HTTP requests encoded as key-value pairs, to which the server responds with XML documents. The standard operations of WFS are based on the `GetCapabilities`, `DescribeFeatureType` and `GetFeature` requests, as shown in Fig. 1.

3.1 GetCapabilities Request

The `GetCapabilities` request lists the WFS versions that the server can work with, the geometries available on the WFS server, together with their metadata (e.g. title, maintainers, abstract, provider's contact information, spatial reference system, etc.). It also informs the client which encodings are available for delivering the requested geometries (e.g. GML, GML2, JSON, CSV, etc.). Finally, the XML-based Capabilities Document also indicates which spatial functions are supported for each feature type. Listing 4 shows an example of how a GetCapabilities request can be sent to a WFS server.

Listing 4 GetCapabilities Request Example.

```
http://[SERVER_ADDRESS]/wfs?SERVICE=WFS&REQUEST=GetCapabilities
```

3.2 DescribeFeatureType Request

As shown in Fig. 1, the next step after receiving the Capabilities Document from the WFS server is to perform the `DescribeFeatureType` request. This request, as

[12] http://www.opengeospatial.org/standards/wfs

shown in Listing 5, enables the client to select a feature—previously listed in the Capabilities Document—and specify in which WFS encoding version it should be delivered. The response of this request is an XML document containing all fields of the requested features attribute table and their data types.

Listing 5 DescribeFeatureType Request Example.

```
http://[SERVER_ADDRESS]/wfs?SERVICE=WFS&VERSION=1.0.0&
REQUEST=DescribeFeatureType&TYPENAME=FEATURE_ID&SRSNAME=EPSG:4326
```

3.3 GetFeature Request

The last step to obtain features from a WFS is to perform the GetFeature operation. In this operation the client asks for a feature in a specific WFS encoding version, as shown in Listing 6. Finally, the client receives an XML document containing the feature and its attribute table.

Listing 6 GetFeature Request Example.

```
http://[SERVER_ADDRESS]/wfs?SERVICE=WFS&VERSION=1.0.0&
REQUEST=GetFeature&TYPENAME=FEATURE_ID&SRSNAME=EPSG:4326
```

Although the `DescribeFeatureType` and `GetFeature` requests syntactically only differ in the `REQUEST` parameter, they play different roles in the Web Feature Service standard, namely request information about a certain feature and retrieve the feature itself, respectively.

Another implementation of WFS—the Web Feature Service Transaction (WFS-T)—allows creating, deleting and updating features, but these functionalities are currently not addressed in this work. The WFS characteristics of: a) providing a platform-independent layer for querying geographic features requests on the Web, b) the capability of attaching attributes to the geographic features, and c) being a standard widely used as a vector data source, make WFS one of the most suitable standards for this work.

4 Linked Open Data for Web Feature Services (LOD4WFS Adapter)

Linked Open Data offers a structured approach to describe and interlink raw data on the Web, and the Web Feature Service standard offers a standardized and widely used way to deliver geographic features through web services. The union of these two technologies could increase the accessibility of geographic LOD datasets

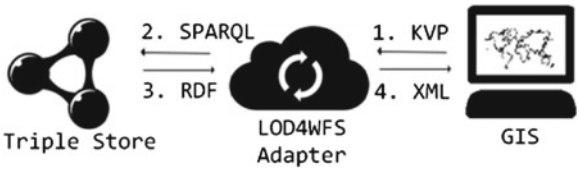

Fig. 2 LOD4WFS adapter overview

significantly. However, there is currently no common way for them to communicate. Filling this gap between LOD and WFS will allow current GIS to access geographic LOD datasets, thus enabling users to exploit the interactive tools of GIS to visualize and analyse them. Having LOD as a data source can also open new functionalities for WFS, namely the possibility of integrating different data sources, which is currently not supported by conventional WFS implementations that usually host their data sources in geographic databases or Shapefiles. This would enable, for instance, having access to the municipalities of a certain country from server A and having its river basins from server B in a single request. From this scenario emerged the idea of creating an adapter to enable access from WFS to LOD. Figure 2 gives an overview of how such an *LOD4WFS Adapter*[13] would enable access from GIS clients to geographic LOD datasets via WFS.

The adapter implements a service, compliant to the OGC WFS specification, which listens to WFS requests and converts these requests into the SPARQL Query Language for RDF.[14] After the SPARQL Query is processed, the LOD4WFS Adapter receives the RDF[15] result set from the Triple Store, encodes it as a WFS XML document, and returns it to the client (e.g. a GIS). This approach enables current GIS to transparently have access to geographic LOD datasets, using their implementation of WFS, without any adaptation whatsoever being necessary. In order to reach a higher number of GIS, the currently most common implementation of WFS has been adopted for the LOD4WFS Adapter, namely the OGC Web Feature Service Implementation Specification 1.0.0 (Vretanos 2002). The LOD4WFS Adapter enables access to geographic LOD datasets in two different ways, which we will call *Standard Data Access* and *Federated Data Accesses* in the following.

4.1 Standard Data Access

The Standard Data Access feature was designed in order to enable access to all geographic LOD datasets contained in a triple store. This feature basically works as an explorer, looking for geographic LOD datasets from a certain Triple Store and making them available via WFS. Due to the possibility of describing different

[13] https://github.com/jimjonesbr/lod4wfs
[14] http://www.w3.org/TR/rdf-sparql-query/
[15] http://www.w3.org/RDF/

types of geometries (polygons, lines, points) and many different coordinate reference systems, which are characteristic requirements of a WFS, we chose the GeoSPARQL Vocabulary as an input requirement for the Standard Data Access feature. Listing 7 shows how geometries and their related attributes are expected to be structured. The geometries are encoded as WKT literals and the attributes of features are linked to the instance of the `geo:Geometry` class via RDF Schema[16] and Dublin Core Metadata Element Set[17] vocabularies. However, there are no constraints on which vocabularies or properties may be used for describing attributes.

Listing 7 LOD dataset example: Turtle RDF encoding of a dataset, including ID and description.

```
@prefix geo:  <http://www.opengis.net/ont/geosparql/1.0#>.
@prefix my:   <http://ifgi.lod4wfs.de/resource/>.
@prefix sf:   <http://www.opengis.net/ont/sf#>.
@prefix dc:   <http://purl.org/dc/elements/1.1/>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#>.

my:FEATURE_RECIFE a geo:Feature ;
   rdf:ID "2611606"^^xsd:integer ;
   dc:description "Recife"^^xsd:string ;
                  geo:hasGeometry my:GEOMETRY_REFICE .

my:GEOMETRY_RECIFE a geo:Geometry ;
   geo:asWKT "<http://www.opengis.net/def/crs/EPSG/ 0/4326> POLYGON ((
        -35.0148559599999984 -8.0564907399999992,
        -34.9939074400000010 -8.0493884799999993,
        ...
        -35.0148559599999984 -8.0564907399999992)) "^^sf:wktLiteral .
```

4.1.1 Required Metadata

In order to make the datasets discoverable via the Standard Data Access feature, additional metadata must be added to the datasets. We make use of Named Graphs for this purpose. Every Named Graph in the LOD data source must contain only objects of the same feature type. This approach facilitates the discoverability of Features, speeding up queries that list the Features available in the triple store. In case a Named Graph contains multiple feature types, the features can be split into different layers using the Federated Data Access (see Sect. 4.2). Finally, each Named Graph needs to be described by certain RDF properties, namely `abstract`, `title` and `subject` from the Dublin Core Terms Vocabulary.[18] This information helps the adapter to classify all Features available in a Triple Store, so that they can be further on discovered by the WFS client through the WFS Capabilities Document (see Listing 8). Alternatively, the LOD4WFS Adapter could also use a query based on other RDF types to construct the Capabilities Document.

[16] http://www.w3.org/TR/rdf-schema/

[17] http://dublincore.org/documents/dces/

[18] http://dublincore.org/documents/dcmi-terms/

Listing 8 Named Graph Example.

```
@prefix dct: <http://purl.org/dc/terms/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

<http://ifgi.lod4wfs.de/graph/municipalities>  dct:title "Brazilian
        Municipalities"^^xsd:string ;
        dct:abstract "Municipalities of the Brazilian Federal
        States."^^xsd:string ;
        dct:subject "municipalities boundaries"^^xsd:string .
```

It is important to emphasize that these RDF properties are used simply as a proof of concept for the proposed adapter, therefore other vocabularies and properties could be used instead.

4.2 Federated Data Access

The Federated Data Access feature offers the possibility of accessing geographic LOD datasets based on predefined SPARQL Queries. Differently than the Standard Data Access, the Federated Data Access feature is able to execute SPARQL Queries to multiple SPARQL Endpoints, thus enabling WFS features to be composed of data from different sources. As a proof of concept of what can be achieved, Listing 9 shows an example of a federated query, combining data from DBpedia and Ordnance Survey of Great Britain. The SPARQL Query is executed against the Ordnance Survey's SPARQL Endpoint,[19] retrieving the GSS Code[20] and geographic coordinates from districts of Great Britain—the coordinates are provided by the Ordnance Survey using the WGS84 lat/long Vocabulary, but this example converts them to WKT literals using the function CONCAT. Afterwards, the retrieved entries are filtered by matching the districts' names with DBpedia entries from the east of England, which are written in English language. The result of this SPARQL Query can be further on listed as a single WFS feature via the LOD4WFS Adapter, thereby providing a level of interoperability between datasets that is currently unachievable by any implementation of WFS, whether using Shapefiles or geographic databases.

Listing 9 Federated Data Access – SPARQL Query Example.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/resource/>
PREFIX wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX os: <http://data.ordnancesurvey.co.uk/ontology/admingeo/>

SELECT ?abstract ?name ?gss(CONCAT("POINT(", xsd:string(?long), " ",
```

[19] http://data.ordnancesurvey.co.uk/datasets/os-linked-data/explorer/sparql

[20] http://data.ordnancesurvey.co.uk/ontology/admingeo/gssCode

```
        xsd:string(?lat), ")") AS ?wkt)
WHERE
       {?subject rdfs:label ?name .
        ?subject wgs84:lat ?lat .
        ?subject wgs84:long ?long .
        ?subject os:gssCode ?gss .
        ?subject a os:District
        SERVICE <http://dbpedia.org/sparql/> {
                ?entry rdfs:label ?place .
                ?entry dbpo:abstract ?abstract .
                ?entry dbpo:isPartOf dbp:East_of_England
                FILTER langMatches(lang(?place), "EN")
                FILTER langMatches(lang(?abstract), "EN")
                FILTER ( STR(?place) = ?name )
                }
}
```

The LOD4WFS Adapter provides a web interface that allows users to write, validate and store SPARQL Queries (see Sect. 4.3.2).

4.3 LOD4WFS Software Architecture

The LOD4WFS Adapter, which was entirely developed in the Java programming language, is divided into 6 main system modules: *WFS Interface*, *Web Interface*, *Request Validator*, *Query Manager*, *Connection Manager* and *RDF2WFS Converter*. Figure 3 shows an overview of the application modules.

4.3.1 Web Interface

The Web Interface is responsible for receiving HTTP requests and translating them to the WFS Interface. It also provides access to a web-based system for maintaining SPARQL Queries created via Federated Data Access and changing the system's settings (e.g. default SPARQL Endpoint). This interface was developed using the Java-based HTTP server Jetty,[21] enabling the application to be deployed without the need of an external servlet container.

4.3.2 WFS Interface

The WFS Interface implements a listener for the standard operations defined in the OGC WFS Specification, namely `GetCapabilities`, `DescribeFeatureType` and `GetFeature`. Its main goal is to create an agnostic communication layer that enables any WFS client implementation to send requests and receive query results.

[21] http://www.eclipse.org/jetty/

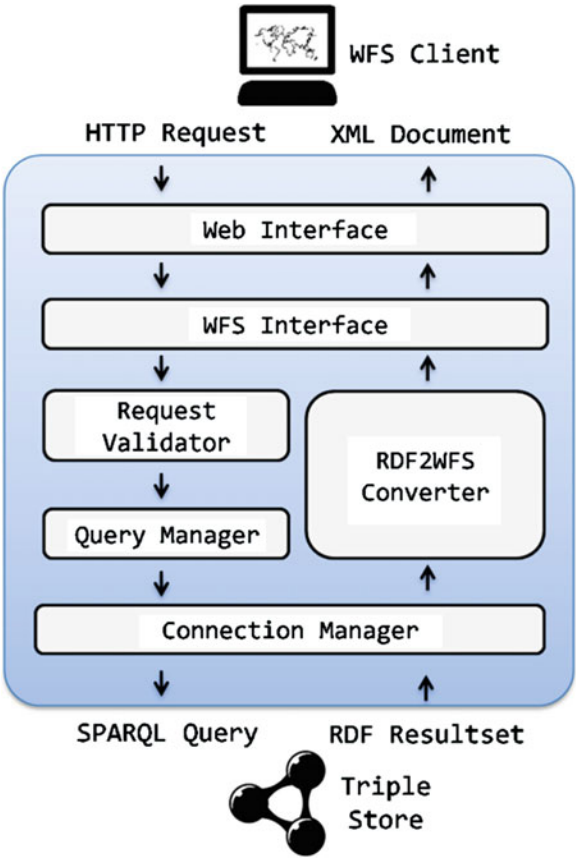

Fig. 3 LOD4WFS modules

Table 1 Validated WFS operations

Operation	Values
SERVICE	WFS by default
REQUEST	GetCapabilities, DescribeFeatureType or GetFeature
SRSNAME	Spatial reference system of a feature available in the system, e.g. EPSG:4326
TYPENAME	ID of a feature available in the system, provided by at the capabilities document
VERSION	1.0.0 by default

4.3.3 Request Validator

This module is responsible for validating the HTTP request received by the WFS Interface, making sure all operations sent by the WFS client are properly fulfilled. Table 1 shows the operations implemented by the Request Validator.

In case of invalid or unknown requests are sent (e.g. non-existing feature or wrong version), an exception report is delivered, according to the Web Feature Service Implementation Specification.

4.3.4 Query Manager

Once the requests have been approved by the Request Validator, they must be translated and processed. The Query Manager is responsible for parsing requests sent by the WFS client and for translating them into SPARQL queries. It is also responsible for mapping each feature to its data access technique (*Standard Data Access* or *Federated Data Access*), which have their requests translated differently. The requests are translated as follows:

GetCapabilities

Standard Data Access–Selects all named graphs (Containers of Features) from the triple store, together with the geometry type of the containing Feature.
Federated Data Access–Lists all customized SPARQL Queries stored via the Web Interface.

DescribeFeatureType

Standard Data Access–Lists all properties attached to a selected Feature together with their range.
Federated Data Access–Lists the variables expected from the customized SPARQL Queries.

GetFeature

Standard Data Access–Selects all geometries of a selected Feature together with the values of their related properties.
Federated Data Access–Executes the customized SPARQL Query of the requested feature to its predefined SPARQL Endpoint.

4.3.5 Connection Manager

The Connection Manager module is responsible for establishing communication from the LOD4WFS Adapter to Triple Stores. Its main goal is to execute SPARQL queries, previously composed by the Query Manager, and forwards its results to the

RDF2WFS Converter for further processing. It is based on the Apache Jena API[22] for building Semantic Web applications.

4.3.6 RDF2WFS Converter

Once the SPARQL Query has been processed and its results are returned to the system, the RDF2WFS module converts it to standard WFS documents. Depending on the request performed by the WFS client (`GetCapabilities`, `DescribeFeatureType` or `GetFeature`) it creates an XML document with the SPARQL Query result and delivers it back to the WFS client.

5 Solution Evaluation

In order to evaluate the performance of the proposed adapter, this section presents tests to compare it to the reference implementation of OGC WFS, namely the software server for geospatial data GeoServer.[23] The test compares the server response time for the `GetFeature` request in both LOD4WFS and GeoServer WFS implementations. Its main goal is to measure the time each of the services takes to process a GetFeature request, perform the query on the storage management system and send the XML document back to the client. For setting up GeoServer, the database PostgreSQL,[24] with its spatial extension PostGIS,[25] was chosen as feature storage for the WFS (Scenario A). For the LOD4WFS Adapter, three different Triple Stores were tested, namely Parliament,[26] Fuseki[27] and OWLIM-Lite[28] (Scenario B). The `GetFeature` requests were performed using the command line tool cURL.[29] The standard installations of all software involved in the tests were kept. Figure 4 shows an overview of how the test environment is structured.

5.1 Test Environment

All tests were performed using a virtual machine as specified in Tables 2 and 3.

[22] http://jena.apache.org/
[23] http://geoserver.org/display/GEOS/Welcome
[24] http://www.postgresql.org/
[25] http://postgis.net/
[26] http://parliament.semwebcentral.org/
[27] http://jena.apache.org/documentation/serving_data/
[28] http://www.ontotext.com/owlim
[29] http://curl.haxx.se/

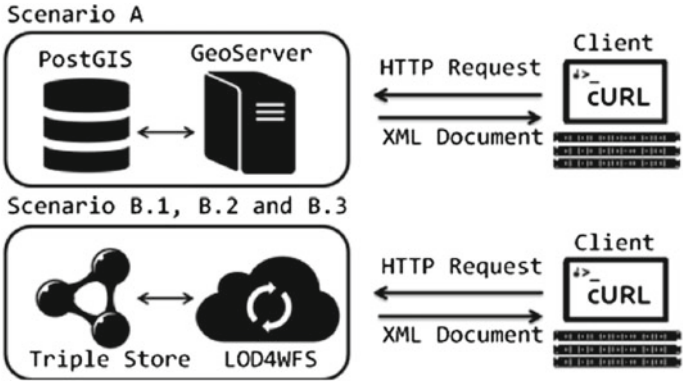

Fig. 4 Test environment overview

Table 2 Hardware environment

Processor	Intel(R) Xeon(R) CPU E5530 @ 2.40 GHz, dual core
Network card	82545EM Gigabit ethernet controller (copper) Capacity: 1 GB/s
Memory	Clock: 66 MHz 8 GB

Table 3 Software environment

Software	Version
Operating system	Ubuntu server Linux 3.2.0-58-generic (amd64) Version 12.04 LTS File system: ext4
Apache tomcat	6.0.35
GeoServer[a]	2.4.4
PostgreSQL	9.1
PostGIS	1.5.3
OWLIM-Lite[a]	4.0
Java runtime	Sun microsystems Inc.: 1.6.0_27 (OpenJDK 64-Bit server VM)
cURL	7.29.0

[a] Embedded at OpenRDF Sesame 2.7.0 and hosted with Apache Tomcat

Table 4 Test datasets

Brazilian municipalities dataset	
Number of geometries	5799
Dataset size	11.2 MB
Amazon rivers dataset	
Number of geometries	18690
Dataset size	45 MB
Amazon vegetation dataset	
Number of geometries	39082
Dataset size	173.2 MB

Table 5 Datasets overview for scenario A

Dataset	Table records	Table size[a]
Brazilian municipalities	5799	16 MB
Amazon rivers	18690	55 MB
Amazon vegetation	39083	183 MB

[a] Including indexes

5.2 Test Datasets

The datasets used for the tests (see Table 4) were created by the Brazilian Institute of Geography and Statistics[30] (IBGE). They all contain polygon geometries and are available in Shapefile format.[31]

To test *Scenario A*, the dataset was stored in the PostgreSQL database and further on added to the GeoServer as a data source for feature layers (see Table 5). This was necessary to enable access to the features through the GeoServer WFS interface.

In order to use the same dataset for *Scenario B*, the dataset had to be converted to LOD, fulfilling the characteristics previously discussed in Sect. 4.1. For this purpose, a script (*shp2rdf*) in the R programming language[32] was developed for reading Shapefiles and creating an LOD dataset in N-Triples syntax (Beckett 2014). The script uses the rgdal[33] and rgeos[34] packages.

After the conversion, the same RDF N-Triples files (see Table 6) were loaded into the Parliament (*Scenario B.1*), Fuseki (*Scenario B.2*) and OWLIM-Lite (*Scenario B.3*) Triple Stores. The datasets in all test scenarios could also be successfully

[30] http://ibge.gov.br/

[31] ftp://geoftp.ibge.gov.br/mapas_interativos/

[32] http://www.r-project.org/

[33] http://cran.r-project.org/web/packages/rgdal/rgdal.pdf

[34] http://cran.r-project.org/web/packages/rgeos/rgeos.pdf

Table 6 Datasets overview for scenario B

Dataset	Total triples	File size
Brazilian municipalities	86988	28.7 MB
Amazon rivers	359206	113.8 MB
Amazon vegetation	703497	416.1 MB

downloaded and displayed using the WFS clients of GIS QGIS[35] and ArcMap.[36] The converted datasets can be found at the following SPARQL Endpoint.[37]

5.3 Test Procedure

The loaded datasets were queried via HTTP `GetFeature` requests using cURL. The `GetFeature` request was performed 10 times in each test scenario for each dataset, afterwards the arithmetic mean value of the time elapsed was calculated. To avoid the network speed to affect the test results, the download speed was limited to 500 kilobytes per second, so that all test scenarios have the same download performance. Listing 10 shows an example of how the requests per cURL were sent to the test server. Table 7 summarizes the tests performed in each test scenario.

Listing 10 Sample HTTP Request Sent via cURL.

```
$ curl --limit-rate 500k 'http://[SERVER_ADDRESS:PORT]/wfs?SERVICE=
        WFS&VERSION=1.0.0&REQUEST=GetFeature&TYPENAME=FEATURE_ID'
        -o feature.xml;$
```

5.4 Results and Discussion

The results demonstrated a non-substantial efficiency difference between the test scenarios. Querying the Brazilian municipalities dataset, all tested scenarios showed a similar response time, having *Scenario A* as the most efficient one, being 1.33 % faster than the second fastest scenario, namely *Scenario B.1* (see Table 7-I). The efficiency difference querying this dataset was limited to the milli-second scale, though (see Fig. 5).

The tests querying the Amazon rivers dataset showed again a similar performance between the test scenarios using triple stores. Among them, Scenario B.3 had a slightly better performance than *Scenario B.1* and *B.2*. *Scenario A* had again the best

[35] http://www.qgis.org/

[36] http://esri.de/products/arcgis/about/arcmap.html

[37] http://data.uni-muenster.de/open-rdf/repositories/lod4wfs

Table 7 Performance of `GetFeature` requests

Test scenario	Avg. Time (mm:ss.ms)	Standard deviation
I. *Brazilian municipalities dataset*		
A – (GeoServer WFS with PostgreSQL)	00:38.217	0.1916
B.1 – (LOD4WFS with Parliament)	00:38.731	0.0961
B.2 – (LOD4WFS with Fuseki)	00:38.877	0.1283
B.3 – (LOD4WFS with OWLIM-Lite)	00:38.857	0.0762
II. *Amazon rivers dataset*		
A – (GeoServer WFS with PostgreSQL)	02:36.542	0.0802
B.1 – (LOD4WFS with Parliament)	02:38.110	0.1181
B.2 – (LOD4WFS with Fuseki)	02:38.150	0.2795
B.3 – (LOD4WFS with OWLIM-Lite)	02:38.076	0.0872
III. *Amazon vegetation dataset*		
A – (GeoServer WFS with PostgreSQL)	08:35.681	0.0642
B.1 – (LOD4WFS with Parliament)	08:44.013	0.2447
B.2 – (LOD4WFS with Fuseki)	08:44.771	0.1037
B.3 – (LOD4WFS with OWLIM-Lite)	08:39.079	0.0868

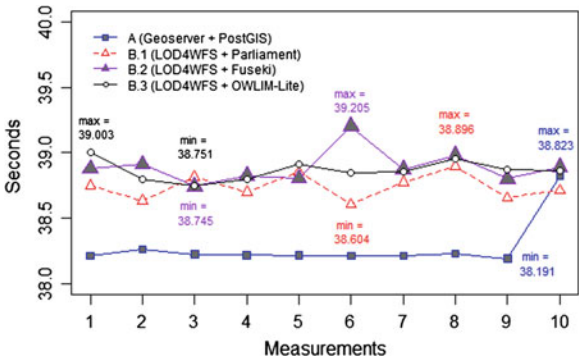

Fig. 5 Performance comparison for the Brazilian municipalities dataset

performance among all test scenarios (see Table 7-II), being 0.97 % faster than the second fastest test scenario, namely *Scenario B.3*.

Tests querying the Amazon vegetation dataset showed a bigger performance difference between the test scenarios involving triple stores (see Table 7-III). *Scenario B.3* demonstrated to have a more efficient response time than *Scenarios B1* and *B.2* when querying bigger datasets, being 0.94 % faster than the second fastest triple store based test scenario, namely *Scenario B.1*. Among all test scenarios, *Scenario A* demonstrated again a better performance than all others test scenarios (see Fig. 7), being 0.65 % faster than *Scenario B.3*.

Though the test results showed no expressive difference between the test scenarios, it demonstrated that the combination of GeoServer with the relational database PostgreSQL still provides a slightly faster platform for enabling access to geographic

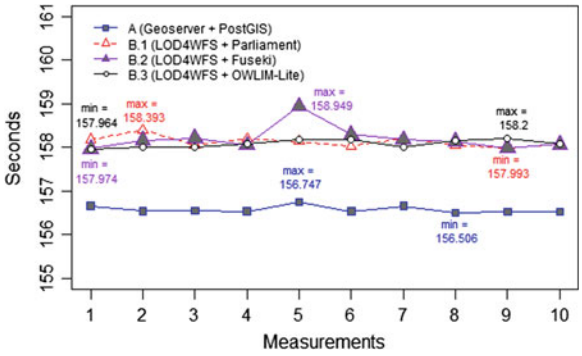

Fig. 6 Performance comparison for the Brazilian rivers dataset

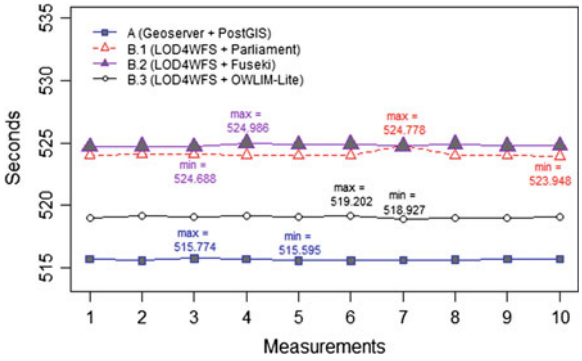

Fig. 7 Performance comparison for the Brazilian vegetation dataset

vector data. The results showed also, considering the given test environment, that the efficiency difference between the LOD4WFS approach and GeoServer with PostgreSQL gets smaller when bigger datasets are requested. The approach proposed by the LOD4WFS relies on the respective triple store's efficiency, which has been shown to be slower than a relational database in our test scenarios. However, the main point we want to stress in this work is the great benefit of having LOD datasets as data source for WFS. This approach provides not only an innovative and competitive way for serving data to current web service standards, but also offers the possibility of combining multiple data sources and creating new datasets on demand (see Sect. 4.2), which is currently not provided by any WFS implementation.

It is also important to mention that the results presented in these tests represent the performance of specific system versions in a single-user environment (see Sect. 5.1), therefore reproducing the tests with other releases will inevitably lead to different results.

6 Related Work

Significant efforts have been made to introduce and enhance the usage of semantics (Kuhn 2005) in geospatial information and web services. Among them are the works on geographical Linked Data (Goodwin et al. 2008), Semantic Geospatial Web services (Roman and Klien 2007), semantic enablement for spatial data infrastructures (Janowicz et al. 2010), structured alignment methods to geospatial ontologies (Cruz and Sunna 2008), semantic-based automatic composition of geospatial Web service chains (Yue et al. 2007) and a framework for semantic knowledge transformation of geospatial data (Zhao et al. 2009). The technological challenges and benefits of adding a spatial dimension to the Web of Data have been also discussed by Auer et al. (2009), where spatial data was systematically extracted from the collaborative project OpenStreetMap[38] and converted to RDF. Efforts on yielding geographic information in OGC web services and embedding them as LOD have been conducted by Roth (2011) with the Geographic Feature Pipes.

Other authors have suggested to use the OGC WFS standard as an interface for providing access to semantic data; Staub (2007) and Donaubauer et al. (2007) have proposed an extension of the existing WFS standard to create a model-driven interface. These works, however, require modification of the OGC WFS standard. In contrast, we use the WFS standard as it is specified by OGC, so that current GIS can access it without any modification.

7 Conclusions and Future Work

This chapter presents an alternative way of accessing geographic LOD datasets from current GIS. We have explored the possibility of using the OGC WFS standard as an intermediate layer between geographic LOD datasets and GIS. We developed an application (LOD4WFS Adapter) that acts as a service for: 1) listening to WFS requests and translating them to SPARQL Queries; and 2) transforming the RDF result set into WFS standard documents. Performance tests of the LOD4WFS Adapter against the reference implementation of OGC WFS (GeoServer) were conducted. The test environment involved three different triple stores and a relational database. The preliminary tests showed that our LOD4WFS Adapter can compete with the reference implementation for WFS services, while providing significantly larger flexibility in accessing and integrating data sources on the Web.

This chapter demonstrates that using LOD as data source for WFS is perfectly feasible and has a great potential. It combines the benefits of a widely used web service standard with the interoperability offered by LOD. This improves accessibility of geographical information on the Web of Data for GIS. Future work includes:

First, the implementation of WFS spatial operations. This would allow the LOD4WFS Adapter to translate supported WFS spatial operations (e.g. contains,

[38] http://www.openstreetmap.org/

intersects) to SPARQL using the Geographic Query Language for RDF (Geo SPARQL). Currently only a few Triple Stores implement GeoSPARQL (e.g. Parliament, Oracle Spatial RDF Semantic Graph,[39] Strabon[40]). This situation may improve once standard Triple Stores will adopt GeoSPARQL and corresponding OGC standards for spatial queries.

The second enhancement is the transaction operation (WFS-T). Currently, the LOD4WFS Adapter implements only requests of geographic information, and does not allow any data manipulation. Implementing the operations defined by WFS-T would enable WFS clients not only to query geographic LOD datasets, but also to insert, edit and delete existing features. The third enhancement we intend is the possibility of accessing geographic LOD datasets encoded as GML and other common formats, e.g. GeoRSS,[41] or GeoJSON.[42] Currently, only WKT is supported.

Finally, we intend to perform more detailed comparisons of the LOD4WFS Adapter and conventional WFS implementations. In order to achieve this, we plan to perform stress tests and to evaluate the application behavior in both single and multi-user environments using different operating systems.[43]

Acknowledgments This work is funded by the German Research Foundation (DFG) through the Linked Data for eScience Services (LIFE) Project, KU 1368/11-1.

References

Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives Z (2007) Dbpedia: a nucleus for a web of open data. In: The semantic web. Springer, Heidelberg, p 722–735

Auer S, Lehmann J, Hellmann S (2009) LinkedGeoData: adding a spatial dimension to the web of data. In: The semantic web—ISWC 2009. Springer, Heidelberg, p 731–746

Battle R, Kolas D (2011) Geosparql: enabling a geospatial semantic web. Semantic Web J 3(4):355–370

Beckett D (2014) N-Triples. A line-based syntax for an RDF graph. W3C proposed recommendation. http://www.w3.org/TR/n-triples/

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

Brickley D, Guha RV (2004) RDF vocabulary description language 1.0: RDF schema. W3C recommendation. http://www.w3.org/TR/rdf-schema/

Cruz IF, Sunna W (2008) Structural alignment methods with applications to geospatial ontologies. Trans GIS 12(6):683–711

Donaubauer A, Straub F, Schilcher M (2007) mdWFS: a concept of web-enabling semantic transformation. In: Proceedings of the 10th AGILE conference on geographic information science

[39] http://www.oracle.com/technetwork/database/options/spatialandgraph/overview/rdfsemantic-graph-1902016.html

[40] http://www.strabon.di.uoa.gr/

[41] http://www.georss.org/

[42] http://geojson.org/

[43] http://lodum.de/life

Goodwin J, Dolbear C, Hart G (2008) Geographical linked data: the administrative geography of great britain on the semantic web. Trans GIS 12(1):19–30
Heath T, Bizer C (2011) Linked data: evolving the web into a global data space. Synth lect semantic web theor technol 1(1):1–136
Herring JR (2011) Simple feature access—part 1: common architecture. OpenGIS implementation standard for geographic information, OGC 06–103r4. http://portal.opengeospatial.org/files/?artifact_id=18241
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
Kuhn W (2005) Geospatial semantics: why, of what, and how. J Data Semant III. Lecture notes in computer science, vol 3534, pp 1–24
Roman D, Klien E (2007) Swing-a semantic framework for geospatial services. In: The geospatial web. Springer, Heidelberg, p 229–234
Roth M (2011) Geographic feature pipes. Diploma thesis, Institute for Geoinformatics, University of Münster, Germany
Staub P (2007) A model-driven web feature service for enhanced semantic interoperability. OSGeo J 3(1)
Vretanos PA (2002) Web feature service implementation specification. OpenGIS implementation standard for geographic information, OGC 02–058. http://portal.opengeospatial.org/files/?artifact_id=7176
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
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