Article

GeoSPARQL 1.1: Motivations, Details and Applications of the Decadal Update to the Most Important Geospatial LOD Standard †

Nicholas J. Car [1,2,*,‡] and Timo Homburg [3,‡]

1 SURROUND Australia Pty Ltd., Canberra 2600, Australia
2 College of Engineering and Computer Science, Australian National University, Canberra 2600, Australia
3 i3mainz–Institute for Spatial Information & Surveying Technology, Mainz University of Applied Sciences, 55128 Mainz, Germany; timo.homburg@hs-mainz.de
* Correspondence: nicholas.car@anu.edu.au; Tel.: +61-477-560-177
† This paper is an extended version of our paper published in GeoLD Workshop at ESWC 2021.
‡ These authors contributed equally to this work.

Abstract: In 2012, the Open Geospatial Consortium published GeoSPARQL defining "an RDF/OWL ontology for [spatial] information", "SPARQL extension functions" for performing spatial operations on RDF data and "RIF rules" defining entailments to be drawn from graph pattern matching. In the 8+ years since its publication, GeoSPARQL has become the most important spatial Semantic Web standard, as judged by references to it in other Semantic Web standards and its wide use for Semantic Web data. An update to GeoSPARQL was proposed in 2019 to deliver a version 1.1 with a charter to: handle outstanding change requests and source new ones from the user community and to "better present" the standard, that is to better link all the standard's parts and better document and exemplify elements. Expected updates included new geometry representations, alignments to other ontologies, handling of new spatial referencing systems, and new artifact presentation. This paper describes motivating change requests and actual resultant updates in the candidate version 1.1 of the standard alongside reference implementations and usage examples. We also describe the theory behind particular updates, initial implementations of many parts of the standard, and our expectations for GeoSPARQL 1.1's use.

Keywords: GeoSPARQL; GeoSPARQL 1.1; spatial; geospatial; Semantic Web; RDF; OWL; OGC; Open Geospatial Consortium; standard; ontology

check for updates

Citation: Car, N.J.; Homburg, T. GeoSPARQL 1.1: Motivations, Details and Applications of the Decadal Update to the Most Important Geospatial LOD Standard. *ISPRS Int. J. Geo-Inf.* **2022**, *11*, 117. https://doi.org/10.3390/ijgi11020117

Academic Editors: Rob Brennan, Brian Davis, Armin Haller and Beyza Yaman

Received: 7 November 2021
Accepted: 7 January 2022
Published: 7 February 2022

1. Introduction and Motivation

The GeoSPARQL standard, issued in 2012 by the Open Geospatial Consortium (OGC) (https://www.ogc.org, accessed on 30 October 2021) is one of the most popular *Semantic Web* standards for spatial data. References to GeoSPARQL in other well-known standards, such as DCAT2 [1] and CIDOC-CRM [2,3] suggests it is popular, as do the incoming links in Linked Open Vocabularies (https://lov.linkeddata.es/dataset/lov/vocabs/gsp, accessed on 30 October 2021) and the list of implementors that includes most of the popular triplestore vendors, a list of which has been compiled here: https://github.com/opengeospatial/ogc-geosparql/issues/59, accessed on 30 October 2021. The original release–GeoSPARQL 1.0 [4]–contained:

- A *specification*: document:
  - The main GeoSPARQL document defining, in human-readable terms and with code snippets, most elements of the standard including ontology elements, geospatial functions that may be performed on Resource Description Format (RDF) [5] data via SPARQL [6,7] queries, entailment rules in the Rules Interchange Format (RIF) [8] for RDF reasoning and requirements and abstract tests for testing ontology data and function implementations.

- An RDF/OWL [9] *schema*:
  - The GeoSPARQL ontology—Semantic Web data model—in an RDF file.
- An RDF *vocabulary*:
  - The simple features vocabulary for "defining SimpleFeature geometry types" taken from [10] in RDF/OWL terms, also in an RDF file.

In this form, GeoSPARQL 1.0 has been used widely; however, requests for updates to it have been received by the OGC. In this publication, an extension of the workshop paper [11], we discuss the motivations behind updating GeoSPARQL 1.0 in the remainder of this section, content of the planned GeoSPARQL 1.1 release (see the GeoSPARQL working repository for the latest candidate release form of GeoSAPRQL 1.1: https://github.com/opengeospatial/ogc-geosparql, accessed on 30 October 2021) in Section 2 and reference implementations and use cases in Sections 3 and 4. Finally, in Section 5, we provide an outlook to further feature requests which are likely to be tackled in future GeoSPARQL releases or replacements.

As the GeoSPARQL 1.1 standard, at the time of publication of [11] had still been in active development, we point to changes which have occurred since. These changes involve the removal of the class `geo:SpatialMeasure` and the inclusion of several supplementary materials (alignments, SHACL shapes and JSON-LD contexts) on which we elaborate extensively in this publication.

The OGC and World Wide Web Consortium's (W3C) *Spatial Data On The Web Working Group* (SDWWG) published a list of Best Practice points (see https://w3c.github.io/sdw/bp/, accessed on 30 October 2021 for an accessible version of the points online and [12] for the corresponding academic publication) for rating web-based spatial data. Through the use of that rating system and other work, the SDWWG noted that: "A best practice for returning geometries in a specific requested CRS has not yet emerged". Given the scope of GeoSPARQL 1.0, this statement indicates that GeoSPARQL 1.0 had not then 'solved' web-based spatial data publishing. The group also informally captured specific suggested updates for GeoSPARQL (https://www.w3.org/2015/spatial/wiki/Further_development_of_GeoSPARQL, accessed on 30 October 2021), however no updates to GeoSPARQL itself were then made.

The authors note that in the 3+ years since that statement's publication, GeoSPARQL 1.0 has become far more widely supported by Semantic Web databases (so-called "triplestores") and other Semantic Web applications, as evidenced by frequent attempts to benchmark geospatial-aware triplestores for GeoSPARQL compliance and performance [13–16]. Some further notes on GeoSPARQL support is provided in Section 5.1.

In 2019, the OGC reconstituted the *GeoSPARQL Standards Working Group* (SWG) to update GeoSPARQL. The motivation for work within the area of GeoSPARQL, that of spatial *Semantic Web* data more generally, and some specific fault fixes and proposed extensions to GeoSPARQL 1.0 are captured in an OGC White Paper [17]. Some, but not all, of the SDWWG's issues raised were taken up by the SDW, for example, the *Best Practices* [12] aspiration that "A possible way forward is an update for the GeoSPARQL spatial ontology. This would provide an agreed spatial ontology, i.e., a bridge or common ground between geographical and non-geographical spatial data...". This issue is specifically addressed in GeoSPARQL 1.1's extensions for scalar spatial data.

Other *Best Practices* issues raised such as "it makes sense to publish different geometric representations of a spatial object that can be used for different purposes" are partly addressed; GeoSPARQL 1.1 indeed indicates how to use multiple geometric representations of geometries in relation to a single feature, but concepts such as defining *roles* for geometries, with respect to features, have not been implemented.

The SWG's Charter–their final scope of work–is also published by the OGC [18] and this guided the SWG's activities. Specific actions of the SWG and their staging are explained through the use of a publicly-available, online, task tracking system within the SWG's working online code repository (https://github.com/opengeospatial/ogc-geosparql/projects/1, accessed on 30 October 2021).

At a high-level, proposed updates to GeoSPARQL by both the SDWWG and the SWG may be categorised as one of the following:

- New geometry serialisations:
  - GeoJSON, KML and other now-popular formats missing from GeoSPARQL 1.0.
  - The possibility to convert between literal formats in-query.
- New and specialised ontology classes and properties:
  - More nuanced spatial data representation and alignment with other systems.
- More spatial functions:
  - Implementing functions well-known in non-Semantic Web spatial systems.
- Scalar spatial properties:
  - Area, volume, etc., alongside geometries.
- Better handling of Spatial (Coordinate) Reference Systems (SRS)
  - Allowing for automated coordinate serialisation conversions.
- Internet protocol-based selection of different geometries for features.

Some of these proposed updates were predicted in GeoSPARQL 1.0, with the *Future Work* section listing several of the points above. The SWG's *Charter*, anticipating that the more obvious updates such as new geometry serialisations would certainly be implemented, listed the following extra areas of investigation that emerged from SWG proponent's discussions:

- Revising "upper ontology" GeoSPARQL structure–how its classes relate to fundamental concepts in ontology;
- Alignments to other ontologies, perhaps *W3C Time Ontology in OWL* [19];
- Catering for very different SRSes, such as Discrete Global Grid Systems.

Specifically ruled out of scope in the *Charter* was any investigation of property graphs. Recent (last several years) discussions in the OGC and elsewhere about property graphs motivated their consideration. However, the SWG proponents felt that while property graphs might be important for future *Semantic Web* spatial data systems, there was more than enough work scoped for initial SWG work (several revisions of the standard) to initially exclude this area of investigation.

After initial meetings, the SWG determined to make multiple versioned releases of GeoSPARQL updates with different goals:

- 1.1 : Extensions that are fully compatible with GeoSPARQL 1.0;
- 1.2: Fully or mostly compatible extensions but which are larger additions to the standard's conceptual coverage;
- 2.0: Future GeoSPARQL likely incompatible with GeoSPARQL 1.0.

The reason for expecting a future, incompatible, GeoSPARQL 2.0 is that early SWG attendees thought spatio-temporal relations and fundamental ontology elements in GeoSPARQL either could or should be remodeled, which might break the current, familiar, feature/geometry class relations. Details of these potential changes have not been fully expounded at the time of this paper, however initial SWG attendees' intuition is that a future GeoSPARQL 2.0, or perhaps a renamed GeoSPARQL replacement, might generalise spatial concepts and move away from only, or primarily, *geo*spatial, or perhaps focus not just on feature/geometry relations but look to generalised mechanisms for describing dimensions of features of which *geometry* is just one of many, and *temporality* might be another. See Section 5 for further details.

Originally unexpected, an area of updates to standard presentation was considered by the SWG. Motivation from conceptual work within the W3C and OGC for the presentation of multi-part standards and the desire by the OGC's "Naming Authority". (the Naming Authority (https://www.ogc.org/projects/groups/ogcnasc, accessed on 30 October 2021) has the remit to "…ensure an orderly process for assigning URIs for OGC resources, such as OGC documents, standards, XML namespaces, ontologies" and also acts as a process evangelist, promoting, as it sees, better standards publication practice) to publish standards more systematically and in more machine-readable forms, as well as programs of work such as the OGC's "Test Bed 17: Model Driven-Standards" (The Testbed 17 work package "Model Driven Standards" (https://portal.ogc.org/files/?artifact_id=95726#ModelDrivenStandards, accessed on 30 October 2021) focused on generating documents from models but also partly developed test implementations of formally-defined, multi-part, standards, such as GeoSPARQL 1.1), this has resulted in the profile declaration explained in the next section.

2. Updates in GeoSPARQL 1.1

In 2021, the GeoSPARQL SWG addressed many of the GeoSPARQL change requests in the 1.1 release. All of the changes reported here are now visible in the current working draft of the GeoSPARQL 1.1 standard, the specification document, and the other standard parts [20]. It is expected that the candidate standard will remain essentially unchanged through to final publication, barring perhaps minor updates due to wider implementation feedback. This section lists work completed only (see Section 5.1 for further notes on final GeoSPARQL 1.1 work) and will point out new features, possibilities, and applications of the elements included in the GeoSPARQL 1.1 update. At first, we discuss the new structure of the standard's specification (Section 2.1), describe relevant extensions to the ontology model and query language in Sections 2.4 and 2.9.

2.1. Profile Declaration

One of the first SWG actions was to link GeoSPARQL 1.0 elements through a *profile* declaration, where a *profile* is a formally-defined variant of a *standard*. *profile* and *standard*, as well as relations between them and their parts, are defined by *The Profiles Vocabulary* [21].

The initial motivation for this was twofold: the SWG's recognition that GeoSPARQL 1.0 consisted of multiple parts, not all of which were easy to discover, so some GeoSPARQL users were unaware of GeoSPARQL resources, some of which have been accidentally duplicated or partly re-implemented, and the desire for machine-readable forms of as many of the standard's parts as possible.

Descriptions of multi-part standards using the *Profiles Vocabulary* are anticipated by the OGC as being their future *best practice* method for standards delivery (This is ascertained through personal communication with the OGC's Naming Authority staff, one of whom was a co-editor of *The Profiles Vocabulary*).

As the elements of GeoSPARQL 1.1 have been created, they too have been described using the Profile Vocabulary, and GeoSPARQL 1.0 has been indicated as being a profile of, that is, a subset of, GeoSPARQL 1.1 since all GeoSPARQL 1.0 elements are present in GeoSPARQL 1.1.

The profile declaration for GeoSPARQL 1.0 and all of its parts will be published alongside those of GeoSPARQL 1.1, currently expected in early 2022. Currently, all draft 1.0 and 1.1 resources are available in the SWG's online code repository (https://github.com/opengeospatial/ogc-geosparql, accessed on 30 October 2021).

The 1.1 releases' profile resources, described using roles given in the *Profiles Vocabulary* are:

1. A profile declaration
    - The definition of the profile links to the things it profiles and a listing of its parts
    - Nn human (HTML) and machine (RDF) readable forms
2. A specification resource
    - As per GeoSPARQL 1.0, the normative document of the GeoSPARQL standard
    - Contains requirements and conformance classes
    - Presented as a document in human-readable form (a PDF) but also containing normative code (*schema*) snippets and function definition tables and examples

3. An RDF/OWL model *schema* resource
   - The GeoSPARQL 1.1 ontology, in both RDF and HTML forms
4. Several *vocabulary* resources
   - Mainly derived from the *schema*
   - Presented in human- and machine-readable forms of the Simple Knowledge Organization System (SKOS) taxonomy model [22]
   - There are vocabularies for Functions, Rules, Conformance Classes in addition to GeoSPARQL 1.0's Simple Features definitions
5. JSON-LD 'context' *mappings*
   - Mappings between local names and fully qualified ontology identifiers for the GeoSPARQL 1.1 ontology and also the Simple Features definitions vocabulary
6. A *validation* resource
   - A series of Shapes Constraint Language (SHACL) [23] shapes for RDF data validation.

All elements of the GeoSPARQL 1.1 profile are listed and linked to in the *profile declaration*'s current draft, online location:

- GeoSPARQL 1.1 draft profile declaration

When finally published, this resource will be available at its namespace IRI location: http://www.opengis.net/def/geosparql, accessed on 30 October 2021.

2.2. Ontology Extensions

GeoSPARQL 1.1 extends the GeoSPARQL ontology by adding multiple new properties and three collection classes. Initially, the SWG proposed a `SpatialMeasure` class to represent scalar spatial measurements too, but this was ultimately not added in favour of a series of 'size' properties only. See Figure 1 for an overview of the current, which is a redrawing of GeoSPARQL 1.1's ontology overview diagram.

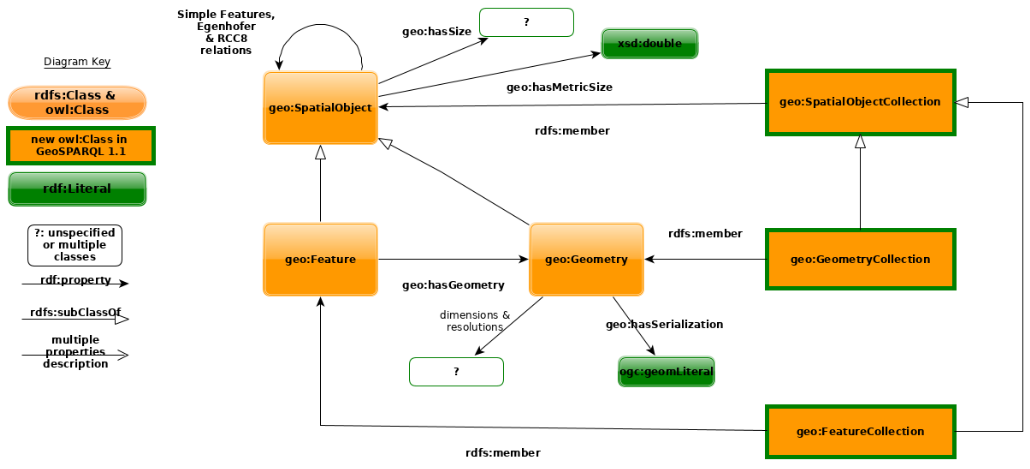


Figure 1. GeoSPARQL 1.1 ontology overview diagram including classes new properties. After GeoSPARQL 1.1's own overview diagram.

2.2.1. Scalar Spatial Properties

Scalar properties of spatial objects, such as a volume, length, can now be indicated in GeoSPARQL 1.1 with either a 'metric' property– one that indicates a literal value in meters–or a non-metric property that may indicate a measurement value and a unit of measure. The metric/non-metric property pairs are all sub properties of a generic 'size' property and have the domain of `geo:SpatialObject` meaning they can be used with either `geo:Feature` or `geo:Geometry` instances. Properties pairs defined are:

- `geo:hasSize` & `geo:hasMetricSize`–generic property
- `geo:hasLength` & `geo:hasMetricLength`
- `geo:hasPerimeterLength` & `geo:hasMetricPerimeterLength`
- `geo:hasArea`, & `geo:hasMetricArea`
- `geo:hasVolume` &
- `geo:hasMetricVolume`

The dual definition of metric and non-metric properties is aimed at allowing both simple and more detailed use. The metric properties are informally preferred for use. The SWG considered their inclusion only; however, the non-metric options were ultimately included to allow for the use of historic units for which no conversion to the metric system is known. Listing 1 shows two metric/non-metric pairs in use for the area and perimeter length, and Figure 2 includes scalar spatial measure examples alongside other ontology implementation examples.

Inclusion of scalar spatial measurements within GeoSPARQL itself addresses concerns raised by the user community (see the SWG's *Charter*) that scalar spatial use with GeoSPARQL was occurring but that it was un-standardised or unguided and thus not necessarily interoperable. The particular pattern of non-metric scalar spatial measure chosen emulates common patterns for such measurements in ontologies such as the W3C's *SOSA* [24,25].

Listing 1. Scalar spatial properties for the Australian federal electoral district of Brisbane.

```
PREFIX ex: <http://example.com/thing/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX qudt: <http://qudt.org/schema/qudt/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX unit: <http://qudt.org/vocab/unit/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

ex:brisbane
    a geo:Feature ;
    rdfs:label "Brisbane Electorate" ;

    geo:hasMetricArea "57486676"^^xsd:double ;
    geo:hasArea [
        qudt:value "5748.6676"^^xsd:float ;
        qudt:unit unit:HA ; # hectare
    ] ;

    geo:hasMetricPerimeterLength "43832"^^xsd:double ;
    geo:hasPerimeterLength [
        qudt:numericValue "27.235942"^^xsd:float ;
        qudt:unit unit:MI ; # mile
    ];
.
```

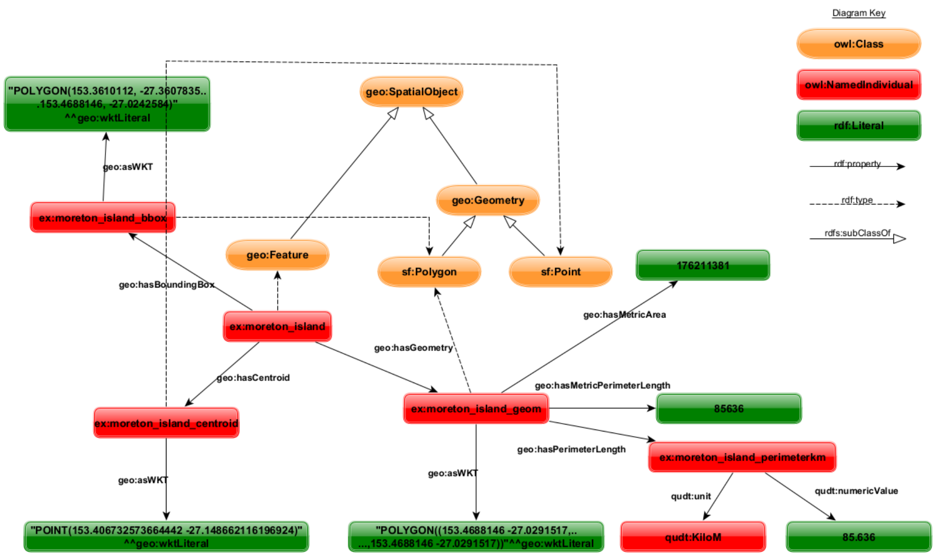


Figure 2. Excerpt of the GeoSPARQL 1.1 ontology including one example feature.

2.2.2. New Geometry Properties

GeoSPARQL 1.1 introduced more sub properties of `geo:hasGeometry`. Where GeoSPARQL 1.0 defined only a single property of it, `geo:hasDefaultGeometry`, GeoSPARQL 1.1 defines `geo:hasCentroid` to indicate geometries with the role of centroid, and other, similar properties, such as geo:hasBoundingBox. Figure 3 shows multiple geometries for a single feature and Listing 2 shows a representation of them in RDF with these new properties used.

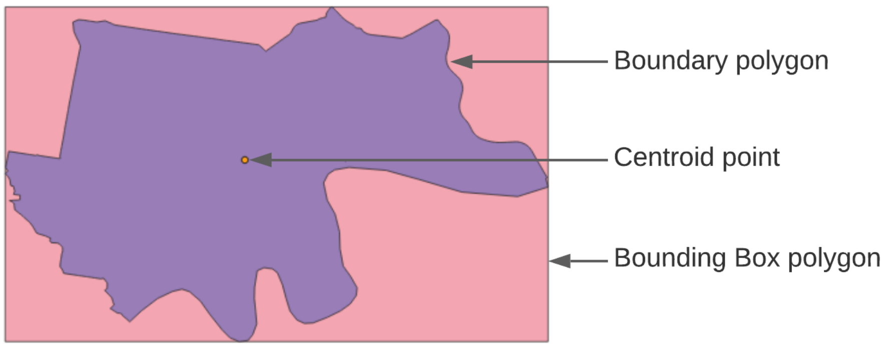


Figure 3. Geometries of the Australian federal electoral district of Brisbane. GeoSPARQL 1.0 contained only the property `geo:hasGeometry` to indicate a `geo:Geometry` instance for a `geo:Feature` instance. GeoSPARQL 1.1 contains the specialised properties of `geo:hasCentroid` & `geo:hasBoundingBox` which can indicate geometries with particular roles. See Listing 2 for an RDF representation of this figure's elements.

The SWG recognised that there could be many more subproperties of `geo:hasGeometry` added to GeoSPARQL 1.1 than the few they did ultimately add. However, no inclusion/exclusion logic was determined by the group nor properties deliberately excluded. One path for future exploration in this area mooted but not implemented for GeoSPARQL 1.1 was the concept of geometry roles–the nature of a geometry's role with respect to a feature–and the SWG discussed creating a geometry roles vocabulary. This is noted again in Section 5.

Listing 2. A possible RDF representation of the elements in Figure 3.

```
PREFIX ex: <http://example.com/thing/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

ex:brisbane
    a geo:Feature ;
    rdfs:label "Brisbane Federal Electorate" ;
    geo:hasGeometry [
        a geo:Geometry ;
        geo:asWKT """POLYGON ((
                153.099932 -27.445258,
                ... ,
                153.099932 -27.445258
            ))"""^^geo:wktLiteral ;
    ] ;
    geo:hasCentroid [
        a geo:Geometry ;
        geo:asWKT """POINT (
                153.030431 -27.438943
            )"""^^geo:wktLiteral ;
    ] ;
    geo:hasBoundingBox [
        a geo:Geometry ;
        geo:asWKT """POLYGON ((
                152.975299 -27.480651,
                153.099932 -27.480651,
                153.099932 -27.404039,
                152.975299 -27.404039,
                152.975299 -27.480651
            ))"""^^geo:wktLiteral ;
    ] ;
.
```

2.2.3. Topological Relations

GeoSPARQL 1.1 does not define any new topological relations or properties for them as compared to GeoSPARQL 1.0 (cf. Table A1), however, the new specification version does include much more supporting material for users of these relations and GeoSPARQL in general, in particular examples of real-world geometry data represented in RDF according to GeoSPARQL 1.1. Figure 4 and its associated Listing, Listing 3, show the sorts of examples given in the GeoSPARQL 1.1 specification document (Annex C) as well as the GeoSPARQL repository of extended examples (https://github.com/opengeospatial/ogc-geosparql/tree/master/1.1/examples, accessed on 30 October 2021).

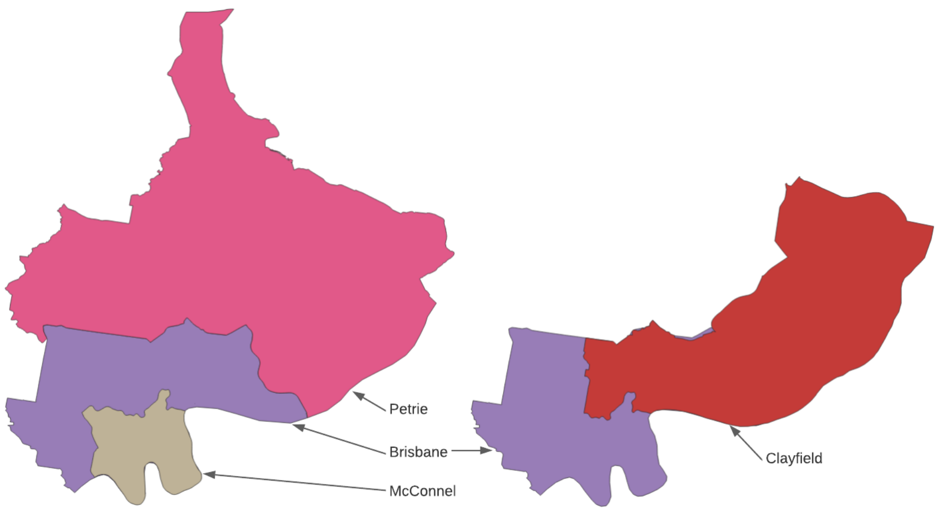


Figure 4. Geometries of the Australian federal electoral districts of Petrie and Brisbane and the state electorate of McConnel (**left**) and Brisbane again and the state electorate of Clayfield (**right**). Brisbane is also the same as in Figure 3. Topological spatial relations that may be declared either between these geometries directly or between the features that these are geometries for. Possible relations for these are: Brisbane `geo:sfContains` McConnel, McConnel `geo:sfWithin` Brisbane, Brisbane `geo:sfTouches` Petrie and McConnel `geo:sfDisjoint` Petrie. See Listing 3 for an RDF representation of this figure's elements.

2.2.4. Support for Collections

GeoSPARQL 1.1 introduces support for collections of Features (geo:FeatureCollection) and Geometries (geo:GeometryCollection) which are both subclasses of the more general class geo:SpatialObjectCollection. This allows for the grouping of features and geometries by specific attributes, and defined object collections in data are also required by standards such as the OGC's *Features* API [26].

While the GIS community commonly organises geospatial features in the form of collections, including in software such as ArcGIS (https://www.arcgis.com/, accessed on 30 October 2021) or QGIS (https://www.qgis.org/, accessed on 30 October 2021), before the inclusion of specific collection classes in GeoSPARQL 1.1., GeoSPARQL data users had to implement virtual collections from the results of SPARQL queries for compatibility with many of their tools. This required more work on behalf of tool maintainers for tools such as the SPARQLing Unicorn QGIS plugin (https://github.com/sparqlunicorn/sparqlunicornGoesGIS, accessed on 30 October 2021).

The SWG recognised that there is little semantic difference between virtual and defined collections from an OWL modelling point of view. However, they also recognised that other users of Semantic Web data might find defined collections much easier to use, hence their ultimate inclusion. This follows recent Semantic Web standards practice, for example, the creation of an extension to the W3C's *SOSA* ontology for observation collections [27].

Listing 3. A possible RDF representation of the elements in Figure 4 in the JSON-LD format which imports the GeoSPARQL 1.1 profile's JSON-LD context resource which allows for simpler JSON data representation through the use of locally-defined names. Supplementary context statements are also included for the example's namespace and supporting namesaces–RDFS.

```
{
    "@context": [
        "https://raw.githubusercontent.com/opengeospatial/
        ogc-geosparql/master/1.1/contexts/geo-context.json",
        {
            "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
            "label": "rdfs:label",
            "ex": "http://example.com/thing/"
        }
    ],
    "@graph": [
        {
            "@id": "ex:brisbane",
            "@type": "Feature",
            "sfContains": {"@id": "ex:mcconnel"},
            "sfTouches": {"@id": "ex:petrie"},
            "sfOverlaps": {"@id": "ex:clayfield"},
            "label": "Brisbane Federal Electorate"
        },
        {
            "@id": "ex:mcconnel",
            "@type": "Feature",
            "sfDisjoint": {"@id": "ex:petrie"},
            "sfWithin": {"@id": "ex:brisbane"},
            "label": "McConnel State Electorate"
        },
        {
            "@id": "ex:petrie",
            "@type": "Feature",
            "sfDisjoint": {"@id": "ex:mcconnel"},
            "sfTouches": {"@id": "ex:brisbane"},
            "label": "Petrie Federal Electorate"
        },
        {
            "@id": "ex:clayfield",
            "@type": "Feature",
            "sfOverlaps": {"@id": "ex:brisbane"},
            "label": "Clayfield State Electorate"
        }
    ]
}
```

It needs to be remarked that the support of geometry collections in GeoSPARQL 1.1 does not replace the previously defined class sf:GeometryCollection defined in the GeoSPARQL 1.0 Simple Features vocabulary. The geo:GeometryCollection class can be used to group geometries which have been defined natively in RDF. The sf:GeometryCollection class is used to define a GeometryCollection within a geometry literal, e.g., WKT Listing 4, which is therefore not modeled in RDF. Listing 4 exemplifies a GeometryCollection within a WKT geometry literal.

Listing 4. GeometryCollection in WKT.

```
GEOMETRYCOLLECTION (
    POINT (40 10),
    LINESTRING (10 10, 20 20, 10 40),
    POLYGON ((40 40, 20 45, 45 30, 40 40))
)
```

However, GeoSPARQL 1.1 improves the access to geometry collections defined in literal types by defining new SPARQL extension functions:

- geof:geometryN—allows the retrieval of the nth geometry inside a `sf:GeometryCollection` instance;
- geof:numGeometries—allows the retrieval of the number of geometries contained in a `sf:GeometryCollection` instance.

2.3. Ontology Alignments

GeoSPARQL may be the most widely used ontology to represent spatial data, but there are many other vocabularies in use which try to encode geospatial locations. Many of these ontologies, such as W3CGeo, Geonames or Wikidata representations of geolocations may only be used to encode point geometries. However, some vocabularies such as NeoGeo or schema.org, also allow for the representation of more complex geometries, for example in Well-Known-Text literals. Alignments with GeoSPARQL 1.1 and 15 other vocabularies containing spatial elements are presented in an Annex of the upcoming standard. In addition to written statements and tables of alignments, alignments have also been presented as RDF files for direct reuse in RDF databases. Listing 5 shows two individual alignments–from schema.org and from the Provenance Ontology–to exemplify their presentation in the standard: a direct class mapping and a property for which there is no mapping mapping.

Listing 5. Two example mappings: a class mapping with schema.org and a non-mapping with a Provenance Ontology property.

```
1.schema.org (sdo)

sdo:GeospatialGeometry owl:equivalentClass geo:SpatialObject~.

Since GeospatialGeometry is the domain of SimpleFeature-like
properties and a superclass of~GeoShape

2. Provenance~Ontology

prov:atLocation: no~mapping

The PROV property indicates a prov:Location, so perhaps a
geo:Feature, but~GeoSPARQL has no property to indicate a
geo:Feature
```

2.4. New Geometry Literal Types

Three new geometry serializations are introduced in GeoSPARQL 1.1:

1. **GeoJSON** (Geo-JavaScript Object Notation) [28].
2. **KML** (Keyhole Markup Language) [29].
3. **DGGS** (Discrete Global Grid System) [30].

2.4.1. GeoJSON & KML

GeoJSON & KML have been much anticipated and were requested by the SDWWG and many users of GeoSPARQL, due to those formats' popularity. The DGGS format is more forward-looking in that it is not driven by user demand but by predicted demand.

An example GeoJSON geometry, a point, representing the location of the centroid of the Brisbane electoral district from Figure 3 is given in Listing 6, Listing 7 shows the same geometry in KML for comparison, and that geometry's inclusion in GeoSPARQL 1.1. RDF is given in Listing 8.

Listing 6. GeoJSON point geometry.

```
{
    "type":"Point",
    "coordinates":[153.030431, -27.438943]
}
```

Listing 7. KML point geometry.

```
<Point>
      <coordinates>153.030431, -27.438943</coordinates>
</Point>
```

In GeoSPARQL 1.1, geometry data formats that allow for feature information, such as GeoJSON, are limited to representing geometries only, to avoid possibly conflicting literal and RDF information. This is in keeping with GeoSPARQL 1.0, which imposed the same restriction on GML data. In the example in Figure 3, GeoJSON is used to indicate the geometry type—`"type":"Point"`—as well as give the geometry's coordinates, but not feature annotations, such as name.

Keyhole Markup Language (KML) is a well-known, XML-based, GML-like geometry format. Its use in GeoSPARQL 1.1 is straightforward.

Listing 8. A representation of the centroid point of the Brisbane electoral district from Figure 3 in RDF (Turtle serialization) using the GeoJSON literal data from Listing 6.

```
PREFIX ex: <http://example.com/thing/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

ex:brisbane
    a geo:Feature ;
    rdfs:label "Brisbane Electorate Centroid" ;
    geo:hasCentroid [
        a geo:Geometry ;
        geo:asGeoJSON
            """{
                "type":"Point",
                "coordinates":[153.030431, -27.438943]
            }"""^^geo:geoJSONLiteral ;
    ] ;
.
```

2.4.2. DGGS Literals

Discrete Global Grid System (DGGS) descriptions of spatial objects can be represented in GeoSPARQL 1.1 using literals too, and their inclusion in GeoSPARQL 1.1 took far more consideration than either GeoJSON or KML. Listing 9 gives an *AusPIX* (https://w3id.org/dggs/auspix, accessed on 30 October 2021) DGGS representation of the area of the Brisbane electoral district from Figure 3.

Listing 9. A DGGS geometry serialization example in RDF (Turtle) of the area of the feature in Figure 3. The generic predicate for indicating a DGGS serialization—`geo:asDGGS`—is used and the specific DGGS—*AusPIX*—is indicated via use of a custom datatype `ex:auspixLiteral`.

```
PREFIX ex: <http://example.com/thing/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

ex:brisbane
   a geo:Feature ;
   rdfs:label "Brisbane Electorate Centroid" ;
   geo:hasGeometry [
      a geo:Geometry ;
      geo:asDGGS
         """<https://w3id.org/dggs/auspix>
         CELLLIST ((
            R8338506 R8338507
            R8338508 R8338516
            R8338530 R8338531
            R8338532 R8338534
            R8338540
         ))"""^^ex:auspixLiteral ;
   ] ;
.
```

GeoSPARQL 1.1 does not provide for specific DGGS literals, for example *AusPIX*, directly but only for an abstract DGGS literal with the `geo:dggsLiteral` datatype and the relevant `geo:asDGGS` geometry property, hence the use of the example namespace `http://example.com/thing/` in Listing 9. This is because the DGGSes that currently exist, of which *AusPIX* is just one, have vastly different representation formats. Users of GeoSPARQL 1.1's `geo:asDGGS` geometry property are expected to indicate the particular DGGS being used by implementations of custom literal datatype properties, as per Listing 9.

To assist with understanding what DGGS data is, Figure 5 shows the data from Listing 9 as well as finer DGGS approximations of the Brisbane electoral district's boundary polygon. DGGS such as *AusPIX* represent areas, points, lines and other geometric shapes with collections of 'cells' of different sizes at different 'levels'. There is no theoretical limit as to how many levels, and thus fine cells *AusPIX* may use, thus too, no theoretical resolution limit.

2.4.3. Appropriateness of DGGS Data as Geometry Literals

The appropriateness of the addition of GeoJSON and KML to GeoSPARQL 1.1 as new geometry literal formats was uncontroversial, with the single consideration of substance being the exclusion of information other than geometry information from GeoJSON, as indicated above.

The inclusion of a placeholder or abstract property and datatype for DGGS literals was quite controversial due to differing perceptions of what DGGS data represents. The SWG members do not all agree that DGGS data represent geometries or features, and there is no straightforward theoretical case to be made for either due to DGGS' novelty and mechanisms that are vastly different from traditional spatial data systems.

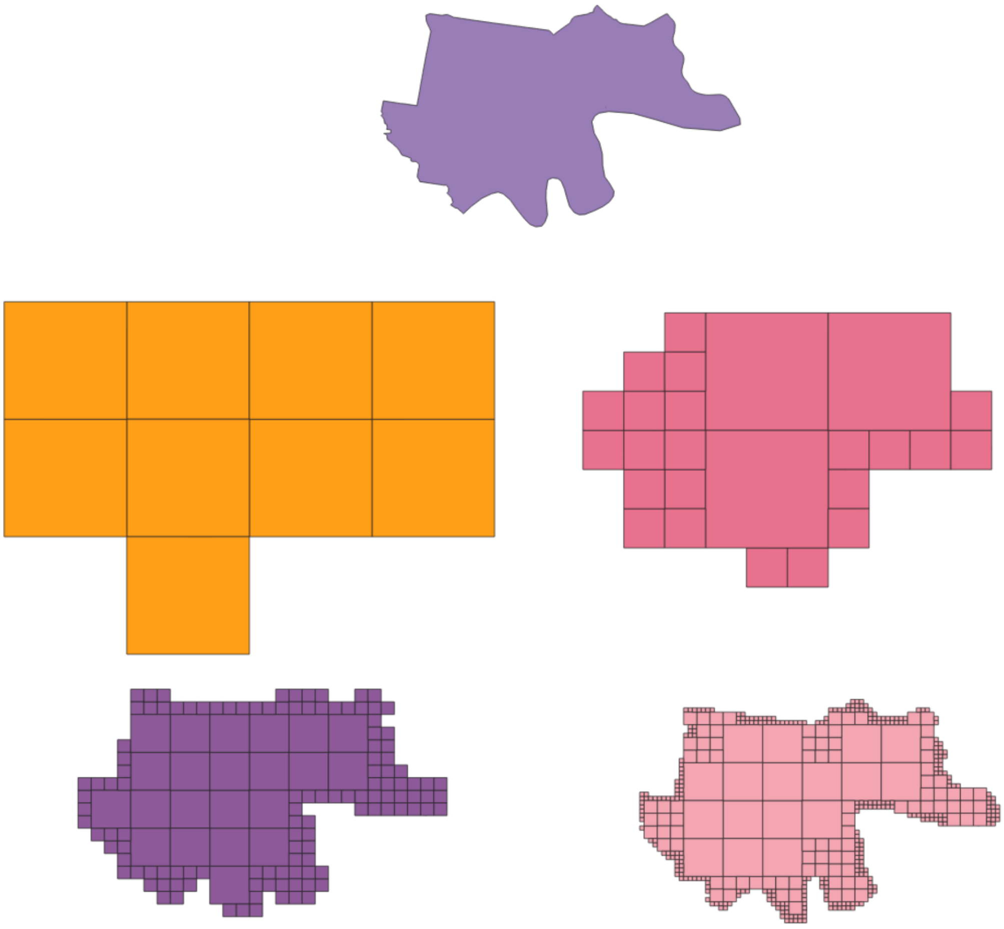


Figure 5. *AusPIX* Discrete Global Grid System representations of the Brisbane electoral district area geometry. The original polygon is represented at the top with the four lower images showing *AusPIX* DGGS cells at higher and higher resolutions, so-called levels 7, 8, 9 and 10.

The SWG decided that the criteria for geometry representation in GeoSPARQL 1.1 were that the literals had to be able to act as geometry objects in order to satisfy at least some major proportion of the GeoSPARQL functions. Essentially, anything that could be used for spatial relations calculation and spatial aggregates could be treated, at least by GeoSPARQL 1.1, as a geometry literal.

During the first half of 2021, within the lifetime of the SWG, a software library was produced that implemented all the `Simple Features` functions, except for `geof:sfCrosses` (see Section 3.1 for the implementation description). While not all *Simple Features* functions were implemented for, nor were functions for the other families of spatial relations, the SWG determined that, given the implementation evidence so far, *AusPIX* DGGS data could indeed act as geometry literals, from GeoSPARQL's point of view. Thus, DGGS literals could satisfactorily act as geometries, at least from GeoSPARQL's point of view. The SWG's discussion on this matter is captured in their online code repository's Issue 112.

2.5. New Geometry Conversion Functions

GeoSPARQL 1.1 provides the conversion functions geof:asWKT, geof:asGML, geof:asKML, geof:asGeoJSON and geof:asDGGS to convert between different literal types while considering the literal types capabilities. For example, a conversion of a WKT literal in the EPSG:25832 coordinate system to either KML or GeoJSON will result in the geometry being converted to the World Geodetic System 1984, as this is the only coordinate reference system valid in the two respective literal types. Hence, a conversion from GeoJSON or KML to WKT will stay in the World Geodetic System 1984 coordinate reference system. To still be able to convert geometries to other coordinate reference systems, GeoSPARQL 1.1 adds the *geof:transform* function, which converts a geometry literal to another coordinate reference system if the literal formats allow such a transformation.

Another addition is a projection function, which allows changing dimensionalities of geometries is likely to be included (https://github.com/opengeospatial/ogc-geosparql/issues/231, accessed on 30 October 2021).

With these additions, future triplestore implementations become very flexible in providing a variety of geometry conversions even without being dependent on another intermediate web service for coordinate/format conversions.

2.6. Spatial Aggregate Functions

While spatial aggregation functions are the norm in many non-semantic geospatial databases such as PostGIS or Oracle Spatial, at the time of defining the GeoSPARQL 1.0 standard, aggregation functions had not yet been introduced into the SPARQL standard, but have been with SPARQL 1.1 [6]. Spatial aggregation functions similar to traditional (relational database) aggregation functions such as AVG, MAX, or MIN allow aggregated results of geometry queries, for example, to create the union of a set of selected serialised geometries. While calculating these aggregates is certainly possible outside of a semantic database, and thus GeoSPARQL, the inclusion of the functions provides distinct advantages:

1. No client-side library is needed to create an aggregated geometry result.
2. Fewer/more appropriate results are returned, for example a union result.
3. Federated SPARQL queries can aggregate results from multiple endpoints.

In addition to *geof:union*, *geof:envelope* and *geof:convexHull* defined in GeoSPARQL 1.0 for use within SPARQL `FILTER` operations, 1.1 defines *geof:aggUnion* as well as *geof:aggBoundingCircle*, *geof:aggCentroid*, *geof:aggConcatLines*—concatenating a set of overlapping linestrings—and *geosf:aggConcaveHull* that can return aggregated results. Listing 10 shows one new function in use.

Functions to retrieve min/max values of geometries' coordinates are added: *geof:minX* & *geof:maxX*, *geof:minY* & *geof:maxY* and *geof:minZ* & *geof:maxZ*.

Listing 10. Aggregation Function example SPARQL query.

```
# returns circle bounding all geometries of Feature <x>
SELECT (geof:aggBoundingCircle(?geo) AS ?circ)
WHERE {<x> geo:hasGeometry/geo:asWKT ?geo .}
```

2.7. Comparison of Query Capabilities

It makes sense to determine how the new version GeoSPARQL 1.1 compares to other, also non-semantic query languages dealing with geospatial representations. We discuss the comparable query languages CQL [31], Simple Features for SQL specification, and the PostGIS implementation extensions as the most typical representatives of spatial data query languages besides GeoSPARQL.

2.7.1. Common Query Language CQL

The Common Query Language CQL was developed for usage with OGC Web Services and provides filter capabilities for feature collection or coverage data. The CQL query language is also part of the currently in-development OGC API Features standard [31], the successor specification to OGC web services. OGC API Features proposes using the Common Query Language (CQL) for filtering but is also open to other query language implementations such as GeoSPARQL. When comparing the filter capabilities of CQL to GeoSPARQL, one can observe that the two query languages provide comparable spatial functionality (cf. Table 1). However, the CQL proposed supports spatiotemporal operators, which may be an addition to GeoSPARQL to be further explored in its continuous development process.

Table 1. Examples of equivalences between CQL and GeoSPARQL for a literal value, query parameters, and comparison predicates in FILTER expressions.

Category	CQL Expression	GeoSPARQL Expression
Query Parameter	limit = 5	LIMIT 5
Literal Value	"A string"	"A string"^^xsd:string
Comparison predicate	name IS NOT NULL	EXISTS {?item my:name ?name}
Spatial Operators	CONTAINS(geometry1,geometry2)	FILTER(geof:sfContains (?geometry1,?geometry2))

To exemplify the relationship between CQL and the GeoSPARQL query language, the SWG will release a Best Practice document (https://opengeospatial.github.io/ogc-geosparql/bestpractice/bestpractice_cql.html, accessed on 30 October 2021), which includes detailed descriptions of the equivalences between the two query languages.

2.7.2. Simple Features for SQL

Another interesting comparison can be made between the query capabilities of GeoSPARQL and the Simple Features for SQL specification [10]. To date, we can conclude that feature parity with the Simple feature specification has been reached with regards to the expression of geospatial relations using properties and using filter functions. Features still missing in the GeoSPARQL 1.1 standard are functions that specifically address attributes of particular geometry types. In that way, it is impossible to access specific points of LineStrings (such as start and end points) and their closedness in-query and the access of polygonal rings and single coordinates of points using query functions. These functions are planned to be implemented in an upcoming, possibly minor update to GeoSPARQL 1.1. We include a comparison table between SQL and the GeoSPARQL specification to highlight the differences in detail.

2.7.3. PostGIS Query Capabilities

In relational spatial databases, one may observe more types of spatial functions as compared to the Simple Features for SQL specification. In addition, more types of geospatial representations can be processed. Finally, one can look at comparable implementations in relational database systems, such as PostGIS. In particular, this is true for representing coverage data, i.e., rasters in PostGIS. The database features comparing raster data to vector data representations, raster algebra operations on raster, and further raster-specific modifications. These query capabilities are currently far beyond the capabilities of GeoSPARQL 1.1 and might be tackled in further development.

2.8. Shacl Shapes for Graph Validation

Since the adoption of SHACL [23] as the recommended way to represent constraints on RDF graphs by W3C, the semantic web community has created shapes for various knowledge domains. These shapes fulfill different purposes, from checking the graph structure for consistency to checking the contents of the graph. Included with GeoSPARQL 1.1, there are SHACL shapes that validate the graph structure as defined in GeoSPARQL 1.1 in the following aspects:

1. *Encouragement of a unified Geometry instance structure:* GeoSPARQL 1.1 encourages geo:Geometry instances to only link to one serialisation. The intention behind this rule is that not all Geometry serialisations that GeoSPARQL 1.1 supports are 1:1 convertible.

Users are still free to use more than one serialisation attached to a Geometry but should be warned about the fact that serialisations may not be 100% equivalent. A simple example of this non-equivalence can be seen when a geometry is associated with a WKT literal in a non CRS84 coordinate reference system and a GeoJSON literal. Because of a limitation of the GeoJSON literal to only accept one coordinate reference system, the literal values of these to literals cannot be equivalent.

2. *Rudimentary checks of literal contents*: Geometry literal contents are checked for plausibility. These checks do not contain the parsing of geometry literals and its validation but aim to check whether the contents of the geometry literal seem to be correct according to its literal type.
3. *Correct usage of GeoSPARQL classes*: Several SHACL shapes test the proper usage of GeoSPARQL classes. In particular, SpatialObjectCollections are expected to have at least one member relation, and geo:Feature instances are expected to be associated to at least one geo:Geometry instance, whereas each Geometry instance is expected to relate to at least one Geometry serialisation.
4. *Geometry property consistency:* Further SHACL shapes test for the consistency of values and cardinality of properties of a geo:Feature or geo:Geometry. For example, one SHACL shape tests the consistency of dimensionality properties of a geo:Geometry.

For the scope of the standard, GeoSPARQL 1.1 stops at the definition of the aforementioned SHACL shapes. However, the SWG has identified the need for various research communities to define checks of geometry relations and further consistency checks of Geometry contents and their relations to each other. Independently of the efforts of the SWG [32] introduced GeoSHACL for exactly this purpose and the SWG intends to found a community group for the collection of further useful validation shapes. Shapes will be created in a new Github Repository (https://github.com/opengeospatial/ogc-geosparql-shapes, accessed on 30 October 2021).

We illustrate the usefulness of the defined GeoSPARQL 1.1 SHACL shapes using a minimum example in Listing 11.

Listing 11. SHACL shape validation case: A geometry with a wrong literal type, two serializations, a missing connection to its corresponding feature and a non-connected FeatureCollection.

```
ex:electorateCollection rdf:type geo:FeatureCollection .
ex:brisbane
    a geo:Feature ;
    rdfs:label "Brisbane Electorate Centroid" ;
.
ex:brisbane_geom
    a geo:Geometry ;
    geo:asGeoJSON
        """{
            "type":"Point",
            "coordinates":[153.030431, -27.438943]
        }"""^^xsd:string ;
    geo:asWKT """POINT(153.030431 -27.438943)"""^^geo:wktLiteral ;
    geo:asKML """POINT(153.030431 -27.438943)"""^^geo:kmlLiteral ;
.
```

The example shows possible common mistakes in GeoSPARQL graphs with respect to the aspects mentioned previously. Each mistake cannot be detected using OWL reasoning approaches. While an empty `geo:FeatureCollection` and a non-connected feature instance are not necessarily wrong instances in a graph, we deem these as useless; hence they produce violations in a SHACL validation. Compared to this, using the wrong literal type (xsd:string) or a wrong literal content (geo:kmlLiteral for WKT content) is a clear error in applying the standard specification. The last issue, representing two serialisations connected to a geometry, is something noteworthy. Either the author of the given data is aware that geometries in different serialisations are not necessarily equivalent as precisions and CRS conversions might differ, or the author should consider creating different instances of geometry that can represent different aspects of geospatial data quality.

2.9. JSON-LD Contexts

GeoSPARQL 1.1 provides JSON-LD [33] contexts for the Simple Features and GeoSPARQL vocabularies allowing for the publication of simpler representations of GeoSPARQL data. Listing 3 provides example data as context-less JSON, which can still be interpreted as RDF through its linking to the GeoSPARQL 1.1 JSON-LD context resource. Context-less JSON is simpler than the more verbose JSON-LD for systems to produce and for developers to understand. Presentation of a JSON-LD context for GeoSPARQL is also aimed at assisting implementors of APIs that wish to present both Linked Data API and OGC API Features [26] interfaces: the context-less JSON will be far easier to incorporate into both required specifications' outputs. Tests of such systems are in development (see https://github.com/surroundaustralia/ogcldapi, accessed on 30 October 2021) for a combination of a Linked Data and OGC API framework and (http://floods.surroundaustralia.com, accessed on 30 October 2021) for an instance of its use. This system plans to move its RDF payloads to context-less JSON).

2.10. Requirements and Conformance Class Vocabulary

As the OGC mandates, all GeoSPARQL requirements and conformance classes are described using a URI. However, in the GeoSPARQL 1.1 specification, these unique identifiers are also modeled in RDF as part of the standard's profile specification. This allows the combination of said vocabularies with different other RDF resources.

To date, we can see the usefulness of this design in two cases.

2.10.1. Compliance Benchmarking

Good practices of standards of any kind are that they are first defined and then implemented in reference implementations. To test whether the reference implementation and all following implementations fulfil the criteria that the given standard sets, compliance benchmarking can be used. [13] created the first compliance benchmark for GeoSPARQL 1.0 using the HOBBIT benchmarking platform [34]. Once an execution of the GeoSPARQL compliance benchmark is finished, it may produce a benchmark result in RDF (https://github.com/hobbit-project/platform/issues/531, accessed on 30 October 2021). Since the definition of the GeoSPARQL 1.1 requirements and conformance class vocabulary, these results can now be related to the actual definitions of GeoSPARQL requirements in RDF as shown in Figure 6.

The provision of benchmark results connected to requirements of a standard in RDF makes these results accessible as a machine-readable resource.

2.10.2. Partial Data Conformance Claims

In addition to systems claiming to implement GeoSPARQL functions, data may claim conformance to GeoSPARQL's ontology. Such claims may be tested both with RDF reasoners and also with the use of the SHACL validators that GeoSPARQL 1.1 provides (see Section 2.8). Since GeoSPARQL contains many parts, useful data may be created that is conformant to only part of GeoSPARQL, and the vocabulary of conformance classes allows for the indication of conformance of data to parts of GeoSPARQL, not just the whole.

Additionally, particular GeoSPARQL user communities might create more constrained SHACL validators to ensure that GeoSPARQL data conforms to a particular implementation pattern from a set of possible patterns. One example is the pattern whereby each `geo:Feature` is only associated with at most one `geo:Geometry`. In this case, a community could define additional conformance classes, like those in GeoSPARQL, and indicate data conformance to them too. Listing 12 shows a custom validator enforcing a restriction on GeoSPARQL 1.1 Features: that they must have one and only one GeoJSON geometry serialization.

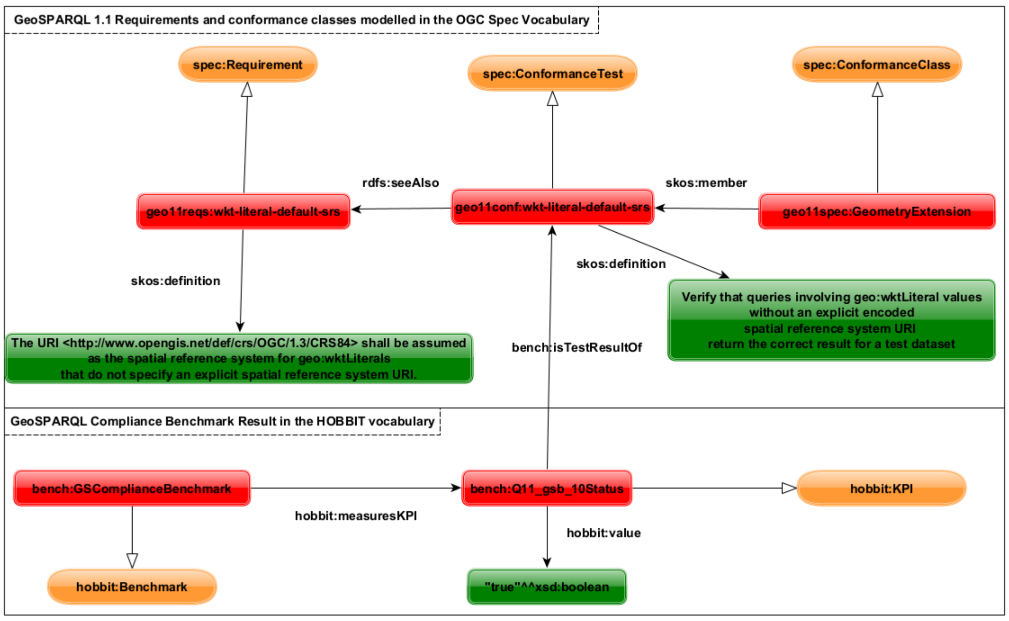

GeoSPARQL 1.1 Requirements and conformance classes modelled in the OGC Spec Vocabulary
spec:Requirement
spec:ConformanceTest
spec:ConformanceClass
rdfs:seeAlso
skos:member
geo11reqs:wkt-literal-default-srs
geo11conf:wkt-literal-default-srs
geo11spec:GeometryExtension
skos:definition
skos:definition
Verify that queries involving geo:wktLiteral values without an explicit encoded spatial reference system URI return the correct result for a test dataset
The URI <http://www.opengis.net/def/crs/OGC/1.3/CRS84> shall be assumed as the spatial reference system for geo:wktLiterals that do not specify an explicit spatial reference system URI.
bench:isTestResultOf
GeoSPARQL Compliance Benchmark Result in the HOBBIT vocabulary
bench:GSComplianceBenchmark
bench:Q11_gsb_10Status
hobbit:KPI
hobbit:measuresKPI
hobbit:value
hobbit:Benchmark
"true"^^xsd:boolean

Figure 6. Requirements vocabulary applied to benchmark results of the GeoSPARQL compliance benchmark applied in the HOBBIT platform. Benchmark results can be mapped to RDF concepts representing requirements and conformance classes of the to-be-tested specification.

Listing 12. An custom GeoSPARQL Feature validator ensuring all Features have only one GeoJSON geometry representation.

```
<hasGeometry-hasSerialization-specialized>
    a sh:NodeShape ;
    sh:targetObjectsOf geo:hasGeometry ;
    sh:property [
        a sh:PropertyShape ;
        sh:path geo:asGeoJSON ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:message "Each node with an incoming geo:hasGeometry,
                    or a specialization of it, must have one and
                    only one outgoing relation of geo:asGeoJSON"@en ;
    ] ;
.
```

3. Reference Implementations

This section describes reference implementations that implement the GeoSPARQL 1.1 specification either entirely or to a certain extent.

3.1. RDFLib DGGS

The *rHEALPix DGGS Simple Feature functions* Python package [35] is a library of functions built in mid-2021 to demonstrate that DGGS geometries within the *rHealPix* DGGS family [36], of which *AusPIX* is a member, could be used in the calculation of *Simple Features* topological relations. Using these functions and RDF and SPARQL capability from the RDFlib (RDFlib is a widely-used, open source, Python programming language, RDF manipulation toolkit: https://github.com/RDFLib/rdflib, accessed on 30 October

2021) Python package, the *RDFlib GeoSPARQL Functions for DGGS* [37] was then built that exposes DGGS geometry-based *Simple Features* calculation functions to RDFlib's SPARQL interpreter via SPARQL extension functions. Listing 13 shows Python application code demonstrating the use of *RDFlib GeoSPARQL Functions for DGGS* with some *AusPIX* data.

At the time of writing, all *Simple Features* topological relations functions were implemented except for `sfCrosses`, but this is expected to be implemented soon: this appears to be held up by programming issues only, not theoretical issues. While only the *rHealPix* DGGS family has currently been catered for, there appears to be no theoretical reason why other DGGSes, such as H3 (https://eng.uber.com/h3/, accessed on 30 October 2021) could not also be catered for.

Listing 13. Python programming code showing the use of the *RDFlib GeoSPARQL Functions for DGGS*. After [37].

```
# import elements from RDFlib and this package (gsdggs)
from rdflib import Literal, Graph, Namespace, URIRef
from gsdggs import~DGGS

EX = Namespace("http://example.com/")
GEO = Namespace("http://www.opengis.net/ont/geosparql#")

# Define the DGGS Geometries, add them to an in-memory RDF
                                    graph
g = Graph()
g.add((
    URIRef('https://geom-a'),
    GEO.asDGGS,
    Literal('CELLLIST ((R0 R10 R13 R16 R30 R31 R32 R40))', EX.
                                    ausPixLiteral)))
g.add((
    URIRef('https://geom-b'),
    GEO.asDGGS,
    Literal('CELLLIST ((R06 R07 R30 R31))', EX.ausPixLiteral))
                                    )
g.add((
    URIRef('https://geom-c'),
    GEO.asDGGS,
    Literal('CELLLIST ((R11 R12 R14 R15))', EX.ausPixLiteral))
                                    )

# Query the in-memory graph
q = """
    PREFIX geo: <http://www.opengis.net/ont/geosparql#>
    PREFIX dggs: <https://placeholder.com/dggsfuncs/>

    SELECT ?a ?b
    WHERE {
        ?a geo:asDGGS ?a_geom .
        ?b geo:asDGGS ?b_geom~.

        FILTER dggs:sfWithin(?a_geom, ?b_geom)
    }"""
# Interate through and print results
for r in g.query(q):
    print(f"{r['a']} is within {r['b']}")
```

3.2. GeoSPARQL-Jena

The GeoSPARQL implementation of the Apache Jena software library GeoSPARQL-Jena [38] provides, according to recent benchmarks [13], the only complete implementation of the GeoSPARQL 1.0 specification. In addition, GeoSPARQL-Jena has been extended in a prototypical use case to support raster data in [39]. This implementation featured prototypical raster support in GeoSPARQL-Jena and aimed at implementing a variety of functions defined in the *Simple Features* implementation standard. Work is underway by members of the SWG to implement DGGS topological relations functions in Jena in a mirror implementation to that in RDFlib described above. A combination of the DGGS algorithms from the RDFlib implementation and the raster handling from [39] will likely provide the first implementation of an RDF library and accompanying triple store to provide full support of the GeoSPARQL 1.1 specification within Jena.

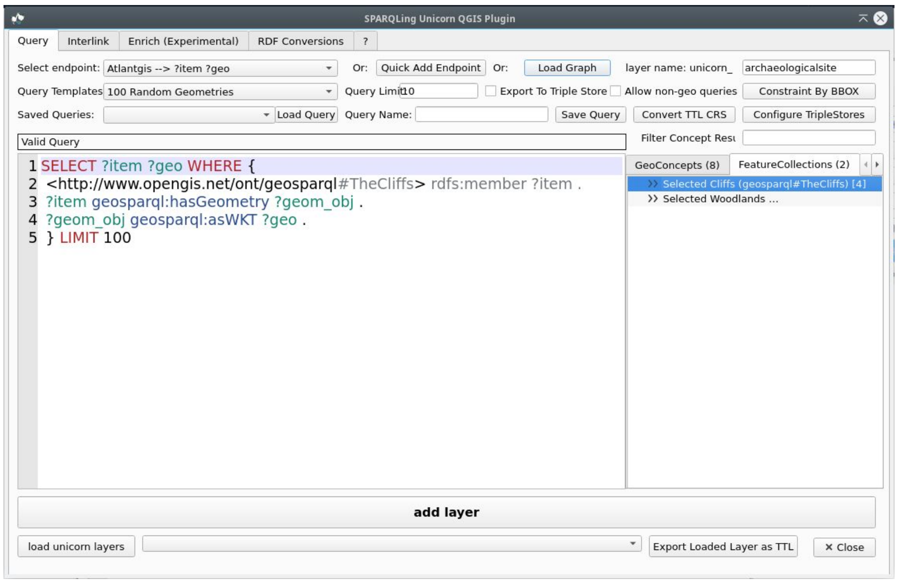
3.3. SPARQLing Unicorn QGIS Plugin

Another implementation already making use of the GeoSPARQL 1.1 specification is the SPARQLing Unicorn QGIS Plugin [40]. This plugin is, to the authors' knowledge, the only client library to make geospatial vector data accessible as vector layers in the popular, open source, QGIS desktop GIS software (https://qgis.org/en/site/, accessed on 30 October 2021). The plugin aims to provide three main functions: (1) Querying Linked Data; (2) preparing geodata for publication as Linked Data resources; and (3) the enrichment of geospatial data from Linked Data resources.

To extend the query capabilities of the plugin for GeoSPARQL 1.1, support for KML and GeoJSON literals has been added, as has support for the processing and review of geo:FeatureCollection and geo:GeometryCollection instances (cf. Figure 7) in a given triplestore.

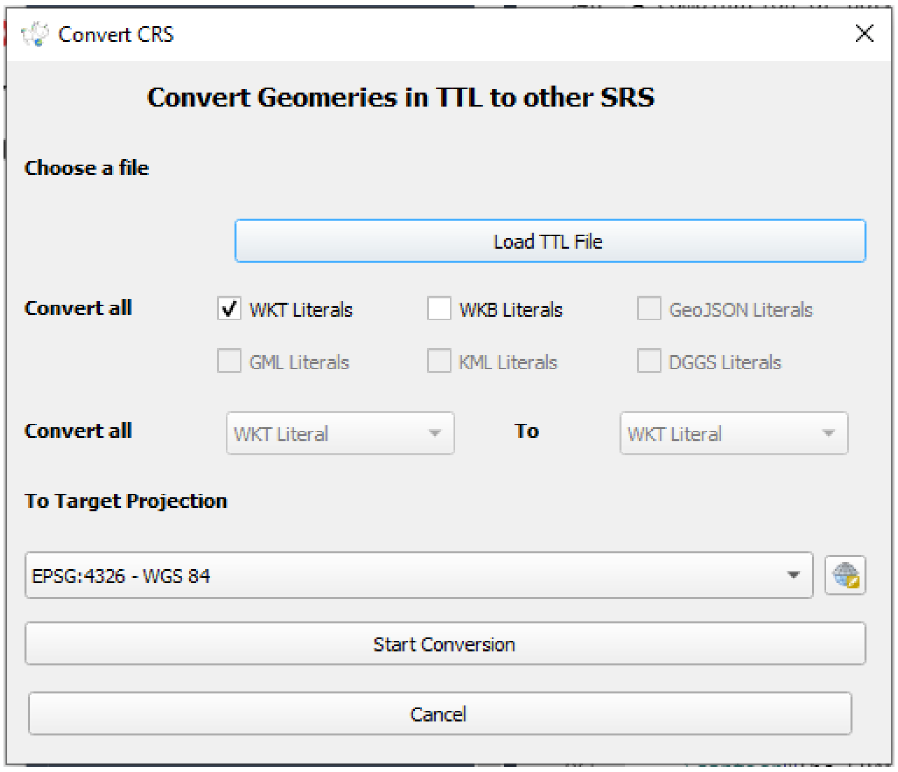
As an aspect of data processing and integration, the plugin implements the newly defined literal formats in its Linked Data conversion dialogue (cf. Figure 8). This dialogue allows converting geometry literals in Linked Data to other CRS supported by QGIS or to other geometry literal formats within the specifications of GeoSPARQL 1.1, as mentioned in Section 2.4.

With this implementation, the general public can discover, access, prepare and convert GeoSPARQL1.1-prepared data.



```
SELECT ?item ?geo WHERE {
 <http://www.opengis.net/ont/geosparql#TheCliffs> rdfs:member ?item .
 ?item geosparql:hasGeometry ?geom_obj .
 ?geom_obj geosparql:asWKT ?geo .
 } LIMIT 100
```

Figure 7. The SPARQL Unicorn QGIS Plugin extended with support for FeatureCollections and GeometryCollections as defined in GeoSPARQL 1.1.


Convert CRS

Convert Geomeries in TTL to other SRS

Choose a file

Load TTL File

Convert all ✓ WKT Literals ☐ WKB Literals ☐ GeoJSON Literals ☐ GML Literals ☐ KML Literals ☐ DGGS Literals

Convert all WKT Literal To WKT Literal

To Target Projection

EPSG:4326 - WGS 84

Start Conversion

Cancel

Figure 8. In-development conversion dialogue for geospatial Linked Data files aiming at full GeoSPARQL 1.1 literal conversion support.

4. Examples of Usage of GeoSPARQL 1.1

In this section, we illustrate the use of the new parts of GeoSPARQL 1.1.

4.1. Profile Declaration

The Australian government is currently conducting a project defining a "National Data Exchange Standard" (NDES) for biodiversity data, validators which are used in an online Application Programming Interface (API) system (The API is online in test form at http://ndesgateway.surroundaustralia.com/, accessed on 30 October 2021. This location will likely remain live until July 2022, at which point it will move to a long-term departmental web location). The system obtains the multiple validators required for use from the many standards that the NDES profiles, including GeoSPARQL, via Linked Data link-following methods which require that standards are made available online in machine-readable form (RDF), with RDF predicates linking to any resources with the role *validator* that they define and also that standards are linked together forming a dependency-based *profile hierarchy*. With such information online, the NDES API is able to recurse through the profile hierarchy, retrieve all defined *validation* resources and then compound them for use automatically, saving system update and maintenance time if/when standards update validators.

Even in the current draft form, the NDES system polls the GeoSPARQL 1.1 publication for validators. Polling has proved useful to retrieve the latest versions of the validators as the SWG has continued to develop them.

4.2. Use of New Geometry Formats

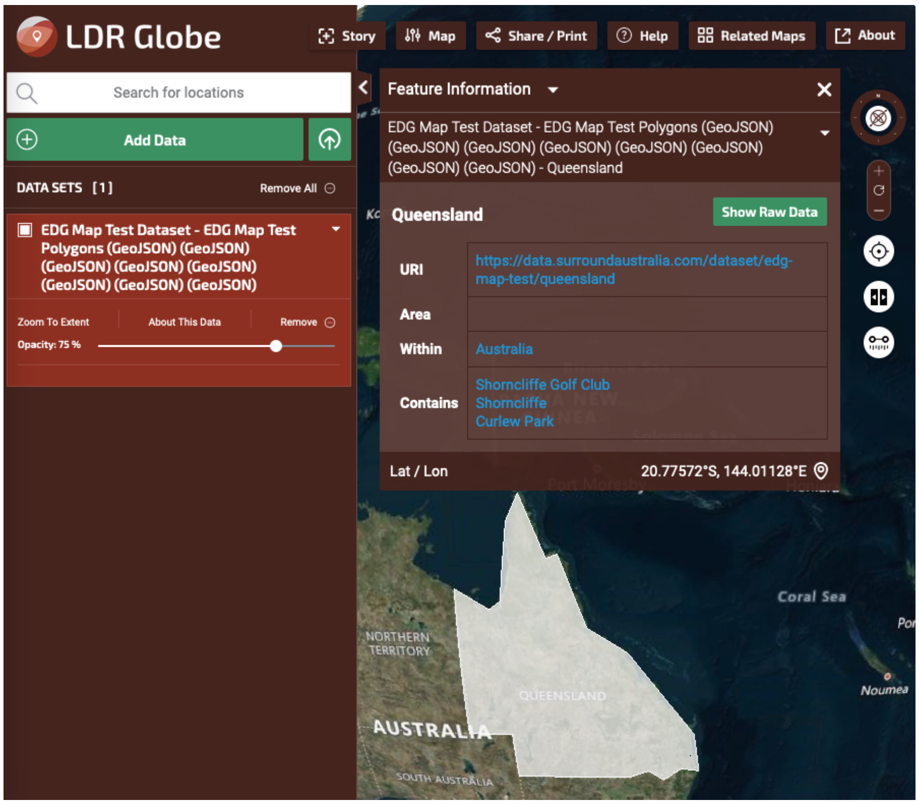
The Australian government's "National Map" (https://nationalmap.gov.au/, accessed on 30 October 2021) is a web-based globe that can display spatial data in a number of formats but none that GeoSPARQL supported until this 1.1 version. Now, with GeoJSON geometry literal support, triplestores can be used to store and filter spatial data, which the globe can now easily consume and display. A triplestore/Globe implementation is now operational (https://globe.surroundaustralia.com/, accessed on 30 October 2021) that reads data from a triplestore, within which the geometries of features are stored as GeoJSON. With geometries in that format, only very simple translations of a few RDF properties to JSON for `geo:Feature` instances are necessary for the Globe can display feature metadata, such as labels, as well as the geometries. Figure 9 shows an instance of the Globe listing test polygons for the Australian state of Queensland as well as three features it contains.

With the Globe's triplestore also containing *Simple Feature* relations, it is also now possible to show feature-to-feature links within the globe instance, as shown in Figure 9. These links are actionable, and the globe can refocus on features navigated to.

4.3. OGC API Features Backend

Traditional spatial data infrastructures are transitioning from being providers of spatial data via specified web services only to becoming spatial knowledge infrastructures [41,42]. These spatial knowledge infrastructures will provide spatial data in two ways, as illustrated in Figure 10. The first way is the provision of spatial data using geospatial web services. The second way is the provision of spatial data as Linked Data, the latter being enabled by ontologies such as GeoSPARQL. However, spatial web services are currently being reworked in the OGC API Features specification, which, as elaborated previously in Section 2.2.4, allows for Linked Data backends.

A live example of data delivered according to Linked Data principles, the OGC API Features specification, and also a SPARQL Protocol (https://www.w3.org/TR/sparql11-protocol/, accessed on 30 October 2021) service is Geoscience Australia's Floods API (http://floods.surroundaustralia.com/, accessed on 30 October 2021), a screenshot of which is shown in Figure 11. Through an application of the CQL to SPARQL mappings that the SWG is creating (see Section 2.7), the Floods API will also be able to respond to CQL queries.



EDG Map Test Dataset - EDG Map Test Polygons (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON)

EDG Map Test Dataset - EDG Map Test Polygons (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON) (GeoJSON) - Queensland

Queensland

URI	https://data.surroundaustralia.com/dataset/edg-map-test/queensland
Area	
Within	Australia
Contains	Shorncliffe Golf Club Shorncliffe Curlew Park

Lat / Lon 20.77572°S, 144.01128°E

Figure 9. A development version of Australia's "National Map" software showing data sourced from a GeoSPARQL 1.1 RDF dataset. The dialog box on the map interface shows features related to the 'Queensland' feature via *Simple Features* relations such as `geo:sfWithin` that the National Map software has previously not been able to show.

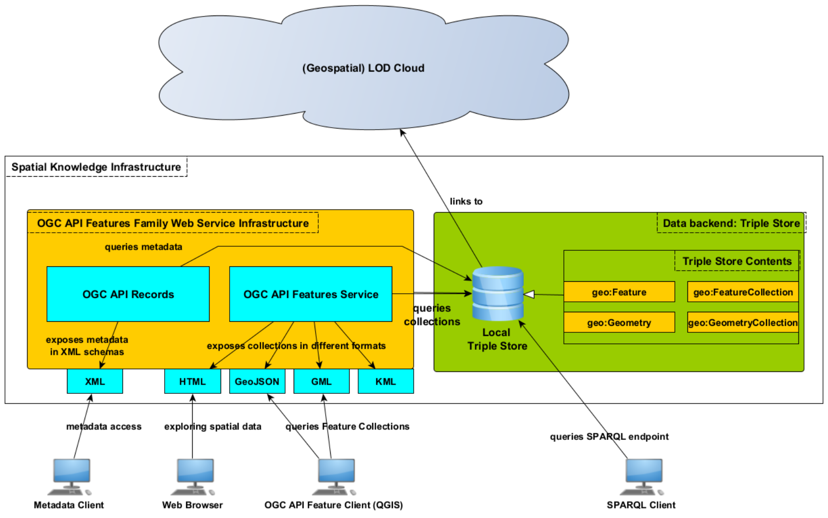
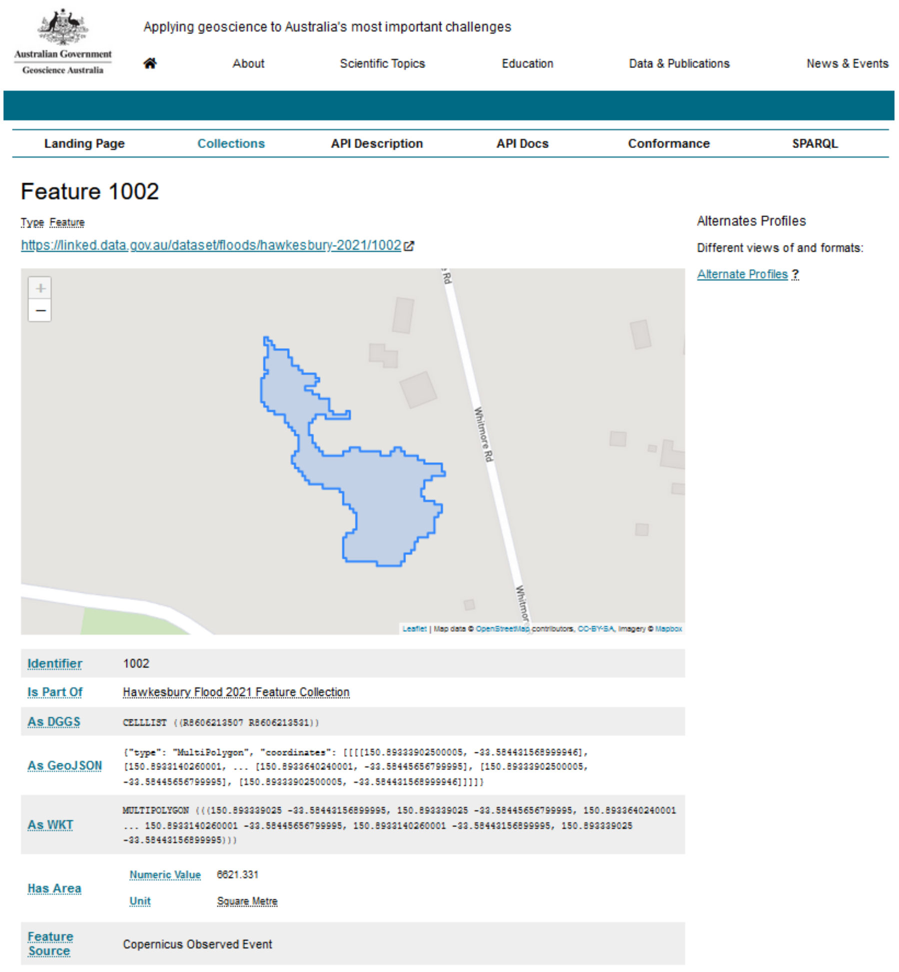


Figure 10. Excerpt of some components of a spatial knowledge infrastructure. A triplestore with GeoSPARQL 1.1. support is provided as a data backend for an OGC API Features powered services infrastructure used by traditional GIS clients.



Feature 1002

Type Feature

https://linked.data.gov.au/dataset/floods/hawkesbury-2021/1002

Identifier	1002
Is Part Of	Hawkesbury Flood 2021 Feature Collection
As DGGS	`CELLLIST ((R8606213507 R8606213531))`
As GeoJSON	`{"type": "MultiPolygon", "coordinates": [[[[150.89333902500005, -33.58443156899946], [150.8933140260001, ... [150.8933640240001, -33.58445656799995], [150.89333902500005, -33.58445656799995], [150.89333902500005, -33.58443156899946]]]]}`
As WKT	`MULTIPOLYGON (((150.893339025 -33.58443156899995, 150.893339025 -33.58445656799995, 150.8933640240001 ... 150.8933140260001 -33.58445656799995, 150.8933140260001 -33.58443156899995, 150.893339025 -33.58443156899995)))`
Has Area	Numeric Value 6621.331; Unit Square Metre
Feature Source	Copernicus Observed Event

Figure 11. Screenshot of a Linked Data API run by Geoscience Australia that delivers flooded area data online in HTML and RDF forms from a GeoSPARQL 1.1 backend. The API is also an OGC API Features-compatible API since the RDF is converted to (Geo)JSON as needed.

This API supplies information according to the required OGC API Features URL structures with very simple queries to a GeoSPARQL 1.1. backend. The simplicity is possible due to GeoSPARQL 1.1's modelling which allows for `geo:FeatureCollection`, `geo:Feature` and `geo:Geometry` instances, the first of which as introduced in GeoSPARQL 1.1 to specifically cater for APIs like the OGC API Features.

APIs such as this Floods API act similarly to previous generation spatial data APIs, such as the well-known Web Feature Service (WFS) (https://www.ogc.org/standards/wfs, accessed on 30 October 2021) from which OGC API Features derives but are also able to present much simpler human User Interfaces (web page), SPARQL endpoints and more data formats.

4.4. DGGS Application Example

DGGSes represent both vector and raster spatial data as collections of cell IDs. Since GeoSPARQL 1.1, the storage system for DGGS data may be a triplestore.

Data for the API in Figure 11 are initially recorded as rasters, and data for the *Australian Statistical Geographies Standard* (ASGS) (https://www.abs.gov.au/websitedbs/D3310114.nsf/home/Australian+Statistical+Geography+Standard+(ASGS), accessed on 30 October 2021) dataset is recorded in vector form. Both datasets are able to be stored using the Apache Jena TDB triplestore (https://jena.apache.org/documentation/tdb/index.html, accessed on 30 October 2021) as GeosPARQL 1.1 data with geometries in the *AusPIX* DGGS format.

The Floods & ASGS datasets are actually stored in the *Loc-I for Disaster Recovery* project's (https://ldr.surroundaustralia.com/, accessed on 30 October 2021) Data Platform with several other rasters and vector datasets, all of which can be cross-queried and presented as either features with geometries in vector form, features with geometries in raster form or as cells within regular grids with cell values of data or the IDs of their corresponding features.

The Loc-I platform does require up-front geometry data conversion to DGGS and feature information to GeoSPARQL RDF but then use of the spatial and non-spatial data is extremely simple–SPARQL queries–and all elements of the data can be used in one system.

5. Future Work

5.1. GeoSPARQL 1.1 Finalization

GeoSPARQL 1.1 is in the final stages of drafting as of November 2021. The new elements of the version, with one possible exception detailed below, have been finalised and the major remaining tasks are to:

- Send the new version to system implementors for wider review.
- Respond to implementors' feedback.
- Register the new IRIs within version 1.1 with the OGC Naming Authority.
- Initiate the mandatory OGC standard update notification period.

Given the simple nature of most of version 1.1's additions and the fact that SWG members have been able to create at least one implementation of almost all new ontology elements and functions, it is expected that the implementors' wider review will not result in major changes.

5.2. Work beyond GeoSPARQL 1.1

The original intention of the GeoSPARQL SWG, when formed in 2019, was to publish a 1.1 version of GeoSPARQL, and then possibly a 1.2 and a 2.0, as described in Section 1. While the scope of GeoSPARQL 1.1 has been in keeping with the original estimates for that version and there are still known un-tackled change requests that the SWG has tagged for a possible 1.2 version (see https://github.com/opengeospatial/ogc-geosparql/milestone/2, accessed on 30 October 2021), the SWG has entertained many thoughts about a more comprehensive tackling of spatial Semantic Web concerns that might require a different direction altogether, for example, non-earth geometries, comprehensive handling of rasters (see Issue 18) or a whole new ontological handling of Coordinate Reference Systems.

If there is a desire for much non-geo work, and if it extends beyond the SPARQL query language, it could be that neither the 'Geo-' or the '-SPARQL' parts of GeoSPARQL will sensibly apply to it, thus something other than a GeoSPARQL 1.2 or even a 2.0 may be required. The SWG will consult on this matter after GeoSPARQL 1.1's release.

The following subsections provide some more detail on specific GeoSPARQL 1.1+ potential directions.

5.3. Inclusion of Further Spatial Data Types

For now, GeoSPARQL 1.1 is a standard to describe 2D and 3D vector data and a single grid reference system. However, members of the SWG have already hinted at the need to represent mobile (e.g., 3D Meshes) and further immobile spatial objects and the inclusion of raster data query capabilities.

5.4. Geometry Roles

As noted in Section 2.2.1, new ontology elements were proposed for GeoSPARQL 1.1 to represent the *roles* of geometries with respect to features. SWG members saw new properties specialising `geo:hasGeometry`, such as `geo:hasCentroid` and `geo:hasBoundingBox`, as indicating geometries with particular roles and that GeoSPARQL could perhaps allow for an extensible way to indicate role by creating a vocabulary of them, which could be added to, rather than describing them in a number of properties that must remain fixed after a GeoSPARQL version's publication.

Timing and SWG participant effort did not allow for roles to be added in GeoSPARQL 1.1, and their addition may be sensible for a GeoSPARQL 1.2.

5.5. Interoperability with Buildings Data

There is a growing appetite for Semantic Web data within the building information modelling (BIM) communities, evidenced by the existence of working groups like the W3C Linked Building Data Community Group (https://www.w3.org/community/lbd/, accessed on 30 October 2021), and the existence of projects such as BRICK [43], IFC/Ontology mappings [44] and the "Building Topology Ontology" (https://w3id.org/bot, accessed on 30 October 2021). These all naturally have spatial components and some already use GeoSPARQL 1.0 (BRICK). It is also clear, evidenced by the fact that members of the Linked Building Data Community Group worked with members of the SWG to outline building-related use cases and some shortcomings of GeoSPARQL 1.0 for their purposes in a 'White Paper' in preparation for the formation of the SWG [17], that GeoSPARQL has long been considered important to that community. Unfortunately, not all of this community's needs were met in GeoSPARQL 1.1, so more could be done to support it, for example, the inclusion of other, specialised, topological relations for voids and non-void features; geometry roles (as per the section above); non-coordinate geometry characterisations, for example, cylinders; and sub-surface geometry handling.

5.6. Formalisation of Spatial Reference Systems

While it is currently possible to use spatial reference system definitions in literal descriptions, spatial reference system definitions have not been completely formalised using an ontology model as of today. This can be a problem in many ways. For example, a triplestore might store a geometry encoded in a coordinate reference system not registered in a well-known repository such as the EPSG Geodetic Parameter Dataset. In these cases, the hosting triplestore might provide custom support for this coordinate reference system, but this support ends once the geospatial data is queried in a federated query scenario. A future version of GeoSPARQL, or a successor to the standard, should be able to describe the contents of spatial reference systems so that the user can make informed decisions about the appropriateness of the spatial reference system assigned to a given geometry that federated queries may resolve even previously not known coordinate reference system definitions.

5.7. Linked Data Fragments Support

GeoSPARQL data collections can be very large, either regarding the number of features or geometries stored, the size of their geometry literals, or both. APIs wishing to deliver large numbers of GeoSPARQL Feature or geometry instances would benefit from the ability to deliver them in a streaming manner, and for this, the so-called "Linked Data Fragments" (LDF) [45] approach has been considered. It appears that the LDF approach if implemented to stream data from an API, would allow for client-side GeoSPARQL topological querying. An example scenario could be that an API user wishes to know if an API contains a feature that *overlaps* a locally held feature. If the API were to stream features from a `geo:FeatureCollection` or as a result of a filtering query, the user could incrementally test for an overlap and cease the streaming session when a match is found.

6. Conclusions

A staged schedule of updates to this essential *Semantic Web* spatial standard has been initiated with simple and strictly backward-compatible changes now in GeoSPARQL 1.1. Reference implementations for all of GeoSPARQL 1.1's new features have been made. Examples of all elements used can be indicated online. Features discussed for GeoSPARQL 1.2 include the formalisation of coordinate reference systems in RDF, the depiction of accuracies and level of detail, and the addition of further–possibly also binary–literal types. Work on GeoSPARQL 1.2 will start at the earliest in mid-2022. GeoSPARQL 2.0, as yet unspecified, is likely to introduce more substantial changes to the standard. Changes proposed for GeoSPARQL 2.0 include broadening the scope of GeoSPARQL to include further kinds of spatial data. To that end, full-featured support for 3D geometries and support for coverages are discussed on the level of data representations. These proposals are related to some growing interest in the semantic web community in representing further geospatial data related to building information [46] and coverage data [39]. More requirements might also be introduced once feedback has been received from the GeoSPARQL 1.1 and 1.2 releases.

Author Contributions: The two authors of this paper, N.J.C. and T.H. contributed equally to this paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Example data indicated in this paper is all already freely available online, linked to at its places of mention. All of the GeoSPARQL 1.1 official example data is stored in the version's repository, see https://github.com/opengeospatial/ogc-geosparql, accessed on 30 October 2021.

Acknowledgments: We thank the individual members of the GeoSPARQL Standards Working Group for the work to update GeoSPARQL for this version 1.1. It was a larger task than initially anticipated, and the SWG rose to that larger occasion. In particular, Joseph Abhayaratna (SWG chair), Matthew Perry (one of the original GeoSPARQL editors), Simon J. D. Cox, and Paul Cripps. We also thank non-SWG member contributors to the standard, particularly Frans Knibbe, Mathias Bonduel, and Robert Gibb. Finally, we wish to thank the Open Geospatial Consortium, both the organisation and their staff, for assisting with this important standards work.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
BIM	Building Information Modeling
CQL	Common Query Language
CRS	Coordinate Reference System

DGGS	Discrete Global Grid System
EPSG	European Petroleum Survey Group
GeoSPARQL	Geographic SPARQL Protocol Furthermore, RDF Query Language
GIS	Geographic Information System
GML	Geography Markup Language
HTML	Hypertext Markup Language
KML	Keyhole Markup Language
JSON	JavaScript Object Notation
LDF	Linked Data Fragments
NDES	National Data Exchange Standard
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
QGIS	Quantum GIS
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RIF	Rule Interchange Format
SDWWG	Spatial Data On The Web Working Group
SHACL	Shapes Constraint Language
SKOS	Simple Knowledge Organization System
SOSA	Semantic Sensor Network Ontology
SPARQL	SPARQL Protocol Furthermore, RDF Query Language
SRS	Spatial Reference System
SWG	Standard Working Group
W3C	World Wide Web Consortium
WKT	Well-Known Text
XML	Extensible Markup Language

Appendix A. GeoSPARQL 1.0 and 1.1 Topological Relations

GeoSPARQL 1.0 contains three types of topological relations. Simple Features Relations, Egenhofer Relations, and relations derived from the specification of the Region Connection Calculus (RCC8) [47].

Table A1. GeoSPARQL Topological Relations as defined in the GeoSPARQL 1.0 standard. The three columns reflect the order of standardization and not the order of relation between the given properties.

Simple Features	Egenhofer Relations	Region Connection Calculus (RCC8)
geo:sfEquals	geo:ehEquals	geo:rcc8eq
geo:sfDisjoint	geo:ehDisjoint	geo:rcc8dc
geo:sfIntersects	geo:ehMeet	geo:rcc8ec
geo:sfTouches	geo:ehOverlap	geo:rcc8po
geo:sfCrosses	geo:ehCovered	geo:rcc8tppi
geo:sfWithin	geo:ehCoveredBy	geo:rcc8tpp
geo:sfContains	geo:ehInside	geo:rcc8ntpp
geo:sfOverlaps	geo:ehContains	geo:rcc8ntppi

References

1. Cox, S.; Browning, D.; Beltran, A.G.; Albertoni, R.; Perego, A.; Winstanley, P. Data Catalog Vocabulary (DCAT)—Version 2. W3C Recommendation, W3C, 2020. Available online: https://www.w3.org/TR/vocab-dcat-2/ (accessed on 30 October 2021).
2. Doerr, M.; Hiebel, G.; Eide, Ø. *CRMgeo: Linking the CIDOC CRM to GeoSPARQL through a Spatiotemporal Refinement*; Technical Report; CIDOC-CRM Documentation Standards Working Group: Geneva, Switzerland, 2013. Available online: https://cidoc-crm.org/crmgeo/sites/default/files/Technical%20Report435-CRMgeo.pdf (accessed on 30 October 2021).
3. Hiebel, G.; Doerr, M.; Eide, Ø. CRMgeo: A spatiotemporal extension of CIDOC-CRM. *Int. J. Digit. Libr.* **2017**, *18*, 271–279. [CrossRef]
4. Perry, M.; Herring, J. *OGC GeoSPARQL—A Geographic Query Language for RDF Data*; OGC Implementation Standard; Open Geospatial Consortium: Rockville, MD, USA, 2012.
5. Cyganiak, R.; Wood, D.; Lanthaler, M. *RDF 1.1 Concepts and Abstract Syntax*; W3C Recommendation; World Wide Web Consortium: Cambridge, MA, USA, 2014.

6. W3C SPARQL Working Group. *SPARQL 1.1 Overview*; W3C Recommendation; World Wide Web Consortium: Cambridge, MA, USA, 2013. Available online: http://www.w3.org/TR/sparql11-overview/ (accessed on 30 October 2021).
7. Seaborne, A.; Harris, S. *SPARQL 1.1 Query Language*; W3C Recommendation; World Wide Web Consortium: Cambridge, MA, USA, 2013. Available online: http://www.w3.org/TR/sparql11-query/ (accessed on 30 October 2021).
8. Kifer, M.; Boley, H. *RIF Overview*, 2nd ed.; W3C Working Group Note; World Wide Web Consortium: Cambridge, MA, USA, 2013. Available online: https://www.w3.org/TR/rif-overview/ (accessed on 30 October 2021).
9. Motik, B.; Patel-Schneider, P.F.; Parsia, B. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*; W3C Recommendation; World Wide Web Consortium: Cambridge, MA, USA, 2009. Available online: https://www.w3.org/TR/owl2-syntax/ (accessed on 30 October 2021).
10. ISO 19125-1:2004. *Geographic Information—Simple Feature Access—Part 1: Common Architecture*; International Organization for Standardization: Geneva, Switzerland, 2004.
11. Car, N.J.; Homburg, T. *GeoSPARQL 1.1: An Almost Decadal Update to the Most Important Geospatial LOD Standard*; Yaman, B., Sherif, M.A., Ngonga Ngomo, A.C., Haller, A., Eds.; Geospatial Linked Data Workshop 2021; CEUR-WS: Hersonissos, Greece, 2021; Volume 2977, pp. 26–33.
12. Van den Brink, L.; Barnaghi, P.; Tandy, J.; Atemezing, G.; Atkinson, R.; Cochrane, B.; Fathy, Y.; Troncy, R. Best practices for publishing, retrieving, and using spatial data on the web. *Semant. Web* **2018**, *10*, 95–114. [CrossRef]
13. Jovanovik, M.; Homburg, T.; Spasić, M. A GeoSPARQL Compliance Benchmark. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 487. [CrossRef]
14. Troumpoukis, A.; Konstantopoulos, S.; Mouchakis, G.; Prokopaki-Kostopoulou, N.; Paris, C.; Bruzzone, L.; Pantazi, D.A.; Koubarakis, M. *GeoFedBench: A Benchmark for Federated GeoSPARQL Query Processors*; ISWC (Demos/Industry): Athens, Greece, 2020.
15. Ioannidis, T.; Garbis, G.; Kyzirakos, K.; Bereta, K.; Koubarakis, M. Evaluating geospatial RDF stores using the benchmark Geographica 2. *J. Data Semant.* **2021**, *10*, 1–40. [CrossRef]
16. Huang, W.; Raza, S.A.; Mirzov, O.; Harrie, L. Assessment and benchmarking of spatially enabled RDF stores for the next generation of spatial data infrastructure. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 310. [CrossRef]
17. Abhayaratna, J.; van den Brink, L.; Car, N.; Atkinson, R.; Homburg, T.; Knibbe, F.; Thiery, F. *OGC Benefits of Representing Spatial Data Using Semantic and Graph Technologies*; OGC White Paper; Open Geospatial Consortium: Rockville, MD, USA, 2020.
18. Abhayaratna, J.; van den Brink, L.; Car, N.; Homburg, T.; Knibbe, F. Ogc Swg Charter. In *OGC GeoSPARQL 2.0 SWG Charter*; Open Geospatial Consortium: Rockville, MD, USA, 2020.
19. Cox, S.; Little, C. *Time Ontology in OWL*; W3C Recommendation; World Wide Web Consortium: Cambridge, MA, USA, 2017.
20. Car, N.J.; Homburg, T.; Perry, M.; Herring, J.; Knibbe, F.; Cox, S.J.D.; Abhayaratna, J.; Bonduel, M. OGC GeoSPARQL—A Geographic Query Language for RDF Data. 2021. Available online: https://opengeospatial.github.io/ogc-geosparql/geosparql11/spec.html (accessed on 30 October 2021).
21. Atkinson, R.; Car, N.J. *The Profiles Vocabulary*; W3C Working Group Note; World Wide Web Consortium: Cambridge, MA, USA, 2020.
22. Miles, A.; Bechhofer, S. *SKOS Simple Knowledge Organization System Reference*; W3C Recommendation; World Wide Web Consortium: Cambridge, MA, USA, 2009.
23. Knublauch, H.; Kontokostas, D. *Shapes Constraint Language (SHACL)*; W3C Recommendation; W3C: Cambridge, MA, USA, 2017.
24. Haller, A.; Janowicz, K.; Cox, S.; Le Phuoc, D.; Taylor, K.; Lefrançois, M. *Semantic Sensor Network Ontology*; W3C Recommendation; World Wide Web Consortium: Cambridge, MA, USA, 2017.
25. Compton, M.; Barnaghi, P.; Bermudez, L.; Garcia-Castro, R.; Corcho, O.; Cox, S.; Graybeal, J.; Hauswirth, M.; Henson, C.; Herzog, A.; et al. The SSN ontology of the W3C semantic sensor network incubator group. *J. Web Semant.* **2012**, *17*, 25–32. [CrossRef]
26. Open Geospatial Consortium. *OGC API—Features—Part 1: Core*; Implementation Standard; Open Geospatial Consortium: Rockville, MD, USA, 2019.
27. Cox, S.J. *Extensions to the Semantic Sensor Network Ontology*; W3C Working Draft; World Wide Web Consortium: Cambridge, MA, USA, 2020.
28. Butler, H.; Daly, M.; Doyle, A.; Gillies, S.; Schaub, T.; Schaub, T. *The GeoJSON Format*; RFC 7946; Internet Engineering Taskforce: Reston, VA, USA, 2016. [CrossRef]
29. Nolan, D.; Lang, D.T. Keyhole Markup Language. In *XML and Web Technologies for Data Sciences with R*; Springer: New York, NY, USA, 2014; pp. 581–618. [CrossRef]
30. Sahr, K.; White, D.; Kimerling, A.J. Geodesic Discrete Global Grid Systems. In *Cartography and Geographic Information Science*; Taylor & Francis: London, UK, 2003; Volume 2, pp. 121–134.
31. Vretanos, P.A.; Portele, C. *OGC API—Features—Part 3: Filtering and the Common Query Language (CQL)*; Open Geospatial Consortium Standard Draft; Open Geospatial Consortium: Rockville, MD, USA, 2021.
32. Debruyne, C.; McGlinn, K. Reusable SHACL Constraint Components for Validating Geospatial Linked Data. *CEUR-WS* **2021**, *2977*, 59–66. Available online: http://ceur-ws.org/Vol-2977/paper8.pdf (accessed on 30 October 2021).
33. Champin, P.A.; Longley, D.; Kellogg, G. *JSON-LD 1.1*; W3C Recommendation; World Wide Web Consortium: Cambridge, MA, USA, 2020. Available online: https://www.w3.org/TR/2020/REC-json-ld11-20200716/ (accessed on 30 October 2021).
34. Röder, M.; Kuchelev, D.; Ngonga Ngomo, A.C. HOBBIT: A platform for benchmarking Big Linked Data. *Data Sci.* **2020**, *3*, 15–35. [CrossRef]

35. Habgood, D. Simple Feature Functions for rHEALPix DGGS. Python Software. 2021. Available online: https://pypi.org/project/rhealpix-sf/ (accessed on 30 October 2021).
36. Gibb, R.; Raichev, A.; Speth, M. *The rHEALPix Discrete Global Grid System*; Landcare Research New Zealand: Lincoln, New Zealand, 2016. [CrossRef]
37. Habgood, D. RDFlib GeoSPARQL Functions for DGGS. 2021. Available online: https://pypi.org/project/geosparql-dggs/ (accessed on 30 October 2021).
38. Albiston, G.L.; Osman, T.; Chen, H. GeoSPARQL-Jena: Implementation and benchmarking of a GeoSPARQL graphstore. 2018. Available online: https://www.semanticscholar.org/paper/GeoSPARQL-Jena%3A-Implementation-and-Benchmarking-of-Albiston-Osman/224bcc2ca5b39294e14eb1202f29ac63f534e8e3 (accessed on 30 October 2021).
39. Homburg, T.; Staab, S.; Janke, D. *GeoSPARQL+: Syntax, Semantics and System for Integrated Querying of Graph, Raster and Vector Data. The Semantic Web—ISWC 2020*; Springer: Berlin/Heidelberg, Germany, 2020; doi:10.1007/978-3-030-62419-4_15. [CrossRef]
40. Thiery, F.; Homburg, T. *SPARQLing Unicorn QGIS Plugin*; Zenodo: Geneva, Switzerland, 2021. [CrossRef]
41. Duckham, M.; Arnold, L.; Armstrong, K.; McMeekin, D.; Mottolini, D. Towards a spatial knowledge infrastructure. In Proceedings of the AGILE 2018, Lund, Sweden, 12–15 June 2018.
42. Ivánová, I.; Siao Him Fa, J.; McMeekin, D.A.; Arnold, L.M.; Deakin, R.; Wilson, M. From spatial data to spatial knowledge infrastructure: A proposed architecture. *Trans. GIS* **2020**, *24*, 1526–1558. [CrossRef]
43. Fierro, G.; Koh, J.; Agarwal, Y.; Gupta, R.K.; Culler, D.E. Beyond a House of Sticks: Formalizing Metadata Tags with Brick. In Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, New York, NY, USA, 13–14 November 2019; pp. 125–134. [CrossRef]
44. Terkaj, W.; Šojić, A. Ontology-based representation of IFC EXPRESS rules: An enhancement of the ifcOWL ontology. *Autom. Constr.* **2015**, *57*, 188–201. [CrossRef]
45. Verborgh, R.; Vander Sande, M.; Hartig, O.; Van Herwegen, J.; De Vocht, L.; De Meester, B.; Haesendonck, G.; Colpaert, P. Triple Pattern Fragments: A Low-cost Knowledge Graph Interface for the Web. *J. Web Semant.* **2016**, *37–38*, 184–206. [CrossRef]
46. Zhang, C.; Beetz, J.; de Vries, B. BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. *Semant. Web* **2018**, *9*, 829–855. [CrossRef]
47. Renz, J. A canonical model of the region connection calculus. *J. Appl. Non-Class. Logics* **2002**, *12*, 469–494. [CrossRef]