The Knowledge Engineering Review, Vol. 00:0, 1–24. © 2004, Cambridge University Press
DOI: 10.1017/S0000000000000000 Printed in the United Kingdom

A comprehensive overview of RDF for spatial and spatio-temporal data management

Fu Zhang [1], Qingzhe Lu [1], Zhenjun Du [2], Xu Chen [3], and Chunhong Cao[1,2]

[1] *School of Computer Science and Engineering, Northeastern University, Shenyang, China; zhangfu216@126.com*
[2] *SIASUN Robot & Automation CO.,Ltd.; duzhenjun@siasun.com*
[3] *North Minzu University, Yinchuan, Ningxia, China; chenxu@nun.edu.cn*

Abstract

Currently, a large amount of spatial and spatio-temporal RDF data has been shared and exchanged on the Internet and various of applications. Resource Description Framework (RDF) is widely accepted for representing and processing data in different (including spatio-temporal) application domains. The effective management of spatial and spatio-temporal RDF data is becoming more and more important. A lot of work has been done to study how to represent, query, store and manage spatial and spatio-temporal RDF data. In order to grasp and learn the main ideas and research results of spatial and spatio-temporal RDF data, in this paper, we provide a comprehensive overview of RDF for spatial and spatio-temporal data management. We summarize spatial and spatio-temporal RDF data management from several essential aspects such as representation, querying, storage, performance assessment, datasets, and management tools. In addition, the direction of future research and some comparisons and analysis are also discussed in depth.

keywords: Resource Description Framework (RDF); Spatio-temporal data management; Spatial RDF; Spatio-temporal RDF; Overview

1 Introduction

Resource Description Framework (RDF) and RDF vocabulary description language RDF Schema (RDFS) are the W3C (World Wide Web Consortium) recommendation normative language to describe information resources and their semantics on the Web [94]. The resources represented by RDF can be shared and exchanged among applications without loss of meaning. In recent years RDF has been a widespread data format for the Semantic Web (an extension of the current Web described by Tim Berners-Lee et al. in [12]) and the Linked Open Data (LOD) cloud [13]. Meanwhile, some topics related to RDF data management such as storage and query have attracted researchers' attentions, and some RDF data management systems have started to emerge such as Sesame and Jena [48]. Recently, several survey reports [66], [18], and [2] have been made to review the state of the art in RDF data management.

With the popularity of RDF, the usage of RDF is now wider than the Semantic Web, more and more companies and organizations (e.g. New York Times and BBC) have started to use RDF for representing and processing their data. RDF has been widely accepted and has rapidly gained popularity in diverse application domains (e.g. databases, social networking, digital signatures and medicine, just to name a few examples [17], [100]).

[1]Correspondence: Fu Zhang is with the School of Computer Science and Engineering, Northeastern University, Shenyang, China, PhD, Associate Professor, email: zhangfu@cse.neu.edu.cn, zhangfu216@126.com.

There is a huge amount of spatial and spatio-temporal information in many real-world applications, such as aviation, atmosphere, and geography. Such information should be shared and exchanged on the Web without semantic missing [77]. Naturally, RDF is employed to represent and process massive spatial and spatio-temporal application data. It is clear that temporally and spatially annotated RDF datasets will grow as well RDF properties will vary with time and space in many applications. As a result, the issues about spatio-temporal extensions of RDF have attracted attentions in the Semantic Web community as well as the OGC (Open Geospatial Consortium) [67], [113]. In particularly, to make spatial data more effectively available, van den Brink et al. in 2019 [119] summarized the work of the joint W3C/OGC Working Group on Spatial Data on the Web that identifies 14 best practices for publishing spatial data on the Web.

Currently, much work is being done in spatial and spatio-temporal RDF data management, which mainly involves representation, query, and storage of spatial and spatio-temporal RDF data. Up to now a huge number of spatial and spatio-temporal RDF data management techniques have been presented in the literature. Analyti and Pachoulakis (2008) [1] provided a survey on the models and query languages for *temporally annotated* RDF, where several temporally annotated RDF models and query techniques were introduced. Athanasiou et al. (2012) [3] and Patroumpas et al. (2014) [77] gave a survey of ways to encode a geometry in RDF and of RDF stores with *geospatial* support. Since that, as we have known, *there is no any comprehensive overview on spatial and spatio-temporal RDF data management.*

To this end, in this paper, we provide a full up-to-date overview of the current state of the art in spatial and spatio-temporal RDF data management. In detail, we make the following main contributions:

- We respectively summarize spatial and spatio-temporal RDF data management from several common and main aspects, including *representation*, *querying*, *storage*, *performance assessment* of storage and querying, and *datasets* and *management tools.*
- Regarding each category of approaches, we make more detailed and in-depth comparisons and discussions. Moreover, besides of some formal notions, we provide examples to explain each approach for illustrating the main idea of each approach.
- Also, the directions for future research and some comparisons and analyses are discussed in our whole survey.

The objective of this paper is twofold. The first is to provide a generic overview of the approaches that have been proposed to manage spatial and spatio-temporal RDF data. The second is to identify and analyze some research directions in the area of spatial and spatio-temporal RDF data management.

The rest of this paper is organized as follows. Section 2 introduces the basic knowledge about RDF and spatio-temporal data. Section 3 provides details of managing spatial RDF data. Section 4 provides details of managing spatio-temporal RDF data. Section 5 makes some discussions and provides some suggestions for possible research directions and Section 6 concludes the paper.

2 RDF and Spatio-Temporal Data

In this section, we present some background information on RDF and spatio-temporal data that are the basis for the different techniques discussed in the later sections.

2.1 Resource Description Framework (RDF)

Resource Description Framework (RDF) is a W3C Recommendation for the notation of metadata on the World Wide Web (WWW) [94]. The basic idea of RDF is a triple model, where:

- Anything is called "*resource*". A resource (identified by International Resource Identifier IRI) has "*properties*", and these properties have values, which may be literal values (e.g. string or integer) or other resources.

- The relationships among resources, properties and values can be expressed by "*statements*", which have the triple structure:

  <*subject predicate object*>. **or**
  <*subject*> <*predicate*> <*object*>.

  It is usually abbreviated as <*s p o*>. The subject or *s* identifies the thing that the statement is about, the predicate or *p* identifies the property or characteristic of the subject, and the object or *o* identifies the value of the property.

There are several kinds of syntax to describe RDF data, such as Turtle family of RDF languages, JSON-LD (JSON-based RDF syntax), RDFa (for HTML and XML embedding), RDF/XML (XML syntax for RDF) [94]. Here is an example of an RDF triple with Turtle syntax to describe a statement "Baron Way Building is located in Amsterdam":

```
<http:www.semanticwebprimer.org/ontology/apartments.ttl#BaronWayBuilding
http://dbpedia.org/ontology/location
http://dbpedia.org/resource/Amsterdam>.
```

Further, **RDF Schema (RDFS)** [94] can enrich and define semantic characteristics of RDF data. RDFS uses *class* (denoted by the syntactic form *rdfs:Class*) to specify categories that can be used to classify resources. The relation between an instance and its class is stated through the property *type* (*rdf:type*). With RDFS one can create hierarchies of classes and sub-classes (*rdfs:subClassOf*) and of properties and sub-properties (*rdfs:subPropertyOf*). Type restrictions on the subjects and objects of particular triples can be defined through domain (*rdfs:domain*) and range (*rdfs:range*) restrictions. Also, a set of entailment rules are defined for RDF and RDFS. Conceptually, these rules specify that an additional triple can be added to an RDF graph if the graph contains triples of a specific pattern. For example, if two triples *(x, rdfs:subClassOf, y)* and *(y, rdfs:subClassOf, z)*, one can infer that *(x, rdfs:subClassOf, z)*. Further, OWL (Web Ontology Language) [69] adds more vocabulary than RDFS for describing properties and classes.

For example, one can state that the IRI *http://www.example.org/friendOf* can be used as a property and that the subjects and objects of *http://www.example.org/friendOf* triples must be resources of class *http://www.example.org/Person*. Such example can be expressed by the following triples (where *http://www.example.org/* is denoted by the prefixe "ex"):

```
<ex:Person            rdf:type        rdfs:Class>.
<ex:is_a_friend_of    rdf:type        rdf:Property>.
<ex:is_a_friend_of    rdfs:domain     ex:Person>.
<ex:is_a_friend_of    rdfs:range      ex:Person>.
<ex:is_a_good_friend_of    rdfs:subPropertyOf    ex:is_a_friend_of>.
```

For indexing RDF data, the query language **SPARQL** [109] has been the standardized query language for RDF. SPARQL is based on matching graph patterns. Most forms of SPARQL query contain a set of triple patterns called a *basic graph pattern*. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. A basic graph pattern matches a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is RDF graph equivalent to the subgraph. The variable bindings can be restricted through filters, which are essentially functions returning a boolean. SPARQL specifies a number of built-in filter functions, such as *regex* or the usual comparison operators (>, ! =, . . .), and also allows additional filter functions that are identified by an IRI. Moreover, SPARQL also contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x dc:title ?title .
    OPTIONAL { ?x ns:price ?price . FILTER (?price < 30) }
    }
```

Please refer to [94] and [109] for more detailed introduction about RDF, RDFS, and SPARQL.

2.2 Spatial and Spatio-Temporal data

Here we first recall several basic notions frequently mentioned in the later text. **Raster** and **vector** are two very different but common data representations used to store geospatial data [3]. *Vector* data use x and y coordinates to define the locations of points, lines, and areas, or polygons, that correspond to physical entities in the real world, such as roads, parcels. *Raster* data use a matrix of square areas to define where physical entities are located, and such squares are also named as pixels, cells and grids.

Moreover, **features** and **geometries** are two crucial geospatial concepts [3], [43]:

- A *feature* is an abstraction of real-world phenomena (e.g., a Station, a Road, or Pretty much anything), which has some spatial location. A feature can occur as a type or an instance. Feature type or feature instance will be used when only one is meant. A feature type is a class of features having common characteristics. A feature instance is a single occurrence of the feature type and represented as an object in a data set.
- A *geometry* can be any geometric shape, typically a point, a polyline or a polygon, as well as several more complex variants (e.g., a polygon with holes). This geometry serves as the spatial location of a given feature using coordinates. Depending on the desired resolution and purpose of use, the location may be defined as one or more geometric shapes. For example, a train station may take the shape of a point (e.g., its centroid to be included in a city map), or a region (in a detailed map of more accuracy), and perhaps even a line (if only its boundary is important). A geometry can be a homogeneous collection (e.g., a multipoint denoting a set of measurements) or a heterogeneous one (e.g., a polygon along with its centroid can be considered as the representation of a region).

Note that, every geometry must be always georeferenced at a well defined system of coordinates (**spatial reference systems SRS** or **coordinate reference systems CRS**). A CRS defines how to relate the coordinates of a geometric object to real locations on the surface of Earth. An important part of the metadata associated with a geometry is its CRS. The elements of a coordinate reference system provide context for the coordinates that define a geometry in order to accurately describe their position and establish relationships between sets of coordinates. There are four parts that make up a CRS: a coordinate system, an ellipsoid, a datum, and a projection [68], [7]. Various types of CRS exist, e.g., geographic CRS (latitude-longitude and optionally ellipsoidal height, e.g., the World Geodetic System 1984-WGS84 is a well-known geographic CRS), three-dimensional cartesian geocentric CRS (X, Y, Z), and projected (X, Y) [50]. One common source of well defined coordinate reference systems is the European Petroleum Survey Group (EPSG), where each CRS is assigned a unique identifier.

Spatial data types [99] or geometric data types provide a fundamental abstraction for modeling the geometric structure of objects in space as well as their relations. One speaks of spatial objects as values of spatial data types. The familiar spatial data types include two-dimensional data types for points (for example, representing the location of a building), lines (for example, describing the ramifications of the Nile Delta), regions (for example, depicting air-polluted zones), spatial networks (for example, representing the routes of the Metro), and spatial partitions (for example, describing the states of a country and their exclusively given topological relationships of adjacency or disjointedness) as well as three-dimensional data types for surfaces (for example, modeling the shape of landscapes) or volumes (for example, representing urban areas).

Spatial information is usually represented by the *literals* of the spatial data types [105]. WKT (Well-Known Text) [43], [71] and GML (Geography Markup Language) [34] are universally used for the serialization of spatial literals. WKT is an OGC (Open Geospatial Consortium) standard [71] for the representation of vector geometry objects, CRSs (Coordinate Reference Systems), and transformation rules between different CRSs. It is also the ISO 19125-1 standard [43], dealing with representation and manipulation of simple features (A simple feature is a feature with all its spatial attributes described piecewise by straight line or planar interpolation between sets of points). As mentioned in [50], geometries in WKT are restricted to 0-, 1- and 2- dimensional geometric objects that exist in 2-, 3- or 4- dimensional coordinate space (i.e., $\mathbb{R}^2$, $\mathbb{R}^3$, or $\mathbb{R}^4$). Geometries that exsit in $\mathbb{R}^2$ consists of points with coordinates x and y, e.g., `POINT`(1, 2). Geometries that exsit in $\mathbb{R}^3$ consists of points with coordinates x, y and z or x, y and m, where m is a measurement. For instance, `POINT`(37.96, 23.71, 27) denotes the temperature (measured in Celcius degree) of a city. Geometries that exsit in $\mathbb{R}^4$ haves points with coordinates x, y, z and m, e.g., `POINT`(1, 1, 2, 27). GML, developed by OGC as well, is the most common XML-based encoding standard for the representation of geospatial data, and provides XML schemas for defining a variety of concepts that are of use in Geography.

Further, the common **spatial relations** include topological, directional and metric relations [19], [75]. *Topological* relations describe concepts of neighbourhood, incidence and overlap and stay invariant under transformations such as scaling and rotation, and can be used to check the relative position of spatial objects to each other like overlap, meet, disjoint, or inside. RCC-8 model [88] and Egenhofer's model [26] are the widely used models for analyzing topological relations. *Direction* relations give a qualitative support to evaluate the relative position of spatial objects like north or southeast. *Metric* relations measure the distance between spatial objects like far or nearby. Moreover, there is a set of spatial operations that can be applied on spatial data [31], including *spatial aggregation*, *topological relations* as mentioned above, and *numeric operations*. The spatial aggregation operators aggregate two or more spatial objects and return a new spatial object, such as Union, Intersection, ConvexHull, and Minimum-BoundingRectangle (MBR). The numeric operations take one or more spatial objects and return a numeric value, such as Perimeter, Area, NoOfInteriorRings, Distance, HaversineDistance, and NoOfGeometries.

Spatio-temporal data types [99] are used to represent the temporal evolution of spatial objects over time, e.g., as in the case of moving objects. Different *temporal types* are classified, including *user-defined time* (which has no special semantics, e.g., January 1st, 1963 when John has his birthday), *valid time* (which is the time a fact is true in the application domain, e.g., the time 2000-2012 when John is a professor), *transaction time* (which is the time when a fact is current in the database, e.g., the system time that gives the exact period when the tuple representing that John is a professor from 2000 to 2012 is current in the database), combination of transaction time and valid time [106], [23]. These evolutions can be discrete or continuous. The common *spatio-temporal data types* include moving point (for example, recording the route of a cell phone user), moving line (for example, representing the boundary of a tsunami), and moving region (for example, describing the motion of an air polluted cloud). Operations on spatio-temporal data types comprise, for instance, the spatio-temporal intersection, union, and difference of moving objects, the computation of the trajectory of a moving point as a line object, the determination of the location of a moving object at a particular time, the calculation of a moving object during a given set of intervals, and the test whether a moving point enters or crosses a moving region. Moreover, with the demand of spatio-temporal applications, a lot of works have been made to modeling, storing, and querying spatio-temporal data [80], [89], [60].

Table 1 Summarization and comparison of spatial RDF representation

Model	**RDF syntax**	**Supported main spatial data types**	**Serialization**	**Supported topological relations**
GeoSPARQL [68], [7]	<s, p, o>	most OGC and Simple Features spatial data types [43], [71]	WKT (Well-Known Text) GML (Geography Markup Language)	Simple Features [43], [71] RCC-8 [88] Egenhofer's relations [26]
W3C Basic Geo Vocabulary [123]	<s, p, o>	points		
NeoGeo [96], [97]	<s, p, o>	Geometry, Point, MultiPoint, LineString, MultiLineString, LinearRing, Polygon, MultiPolygon, GeometryCollection	WKT	RCC-8 [88]
Spatial Literals in RDF [14]	<s, p, o>	Simple Features spatial data types [71]	WKT	
GeoRDF [35]	<s, p, o>	points, lines, polygons		
RDF^i [62], [63]	<s, p, o>	Incomplete geospatial information (i.e., exist, but are unknown or partially known)		RCC-8 [88]

3 Spatial RDF techniques

In this section, we summarize and classify the existing spatial RDF techniques, which extend RDF in different forms to represent and handle spatial information in RDF. We first summarize the *representation* issue of spatial RDF in Section 3.1. Then, we introduce the spatial RDF *storage* and *querying* techniques in Section 3.2. Also, the *performance assessment* about the RDF storage and querying techniques are introduced. Moreover, we also summarize the existing spatial RDF *datasets* and *tools* for managing spatial RDF data in Section 3.3.

3.1 Spatial RDF representation

There have been several attempts for representing and managing geospatial data in RDF, and establishing geospatial RDF standard. As also mentioned in Athanasiou (2012) [3] and Patroumpas (2014) [77], early work included the W3C Basic Geo Vocabulary [123], which enabled the representation of points in WGS84 (World Geodetic System 1984, the most common CRS for spatial data on the web), and GeoRDF [35] as an RDF compatible profile for geographic information (points, lines and polygons). In particular, the OGC GeoSPARQL standard [68], [7] supports representing and querying geospatial data. Table 1 first summarize and compare the spatial RDF representation forms.

(1) GeoSPARQL [68], [7]

The OGC GeoSPARQL standard supports representing and querying geospatial data on the Semantic Web. GeoSPARQL defines a vocabulary for representing geospatial data in RDF, and it defines an extension to the SPARQL query language for processing geospatial data.

In detail, by following a modular design, GeoSPARQL comprises several different components for representing geospatial data in RDF:

- *Core.* Defines top-level RDFS classes for spatial objects. Two main classes are defined: *geo:SpatialObject* and *geo:Feature*, where *geo* is the XML namespace prefix of http://www.opengis.net/ont/geosparql#. The *geo:SpatialObject* represents everything that can have a spatial representation. The *geo:Feature* represents the top-level feature type and is superclass of all feature types. Note that, the *geo:Feature* is subclass of the *geo:SpatialObject* and is equivalent to the class GFI_Feature [44] (where GFI_Feature represents the set of all classes which are feature types and is an instance of the ≪ *metaclass* ≫ GF_FeatureType defined in ISO 19109 Rules for application schema [45]).
- *Topology Vocabulary Extension (relation_family).* Defines RDF properties for asserting and querying topological relations between spatial objects. Such component is parameterized so that different families of topological relations (such as Simple Features relations [43], [71], RCC-8 relations [88] and Egenhofer's Nine-Intersection relations [26]) may be used. For example, the RDF properties *geo:rcc8eq, geo:rcc8dc, geo:rcc8ec, geo:rcc8po, geo:rcc8tppi, geo:rcc8tpp, geo:rcc8ntpp, geo:rcc8ntppi* can be used for representing the RCC-8 topological relations. Note that, the OGC Simple Features relations build on the DE-9IM (Dimensionally

Extended Nine-Intersection Model [21]) and offer the relations: *Equals, Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps* and *Relate*.

- *Geometry Extension (serialization, version)*. Defines RDFS data types for serializing geometry data, geometry-related RDF properties, and non-topological spatial query functions for geometry objects. The *serialization* specifies the serialization standard (e.g., WKT and GML as mentioned in Section 2.2) to use when generating geometry literals and also the supported geometry types. The *version* specifies the version of the serialization format used. Note that the serialization chosen strongly affects the geometry conceptualization. The WKT serialization aligns the geometry types with ISO 19125-1 Simple Features [43]. The GML serialization aligns the geometry types with ISO 19107 Spatial Schema [46], which is a much wider range than the simple features allowed in WKT, including a lot of less commonly used types.
  In detail, a single root geometry class *geo:Geometry* is defined, which represents the top-level geometry type and is superclass of all geometry types. The class *geo:Geometry* is equivalent to the UML class *GM_Object* defined in ISO 19107 Spatial Schema [46]. Also, The class *geo:Geometry* is subclass of the *geo:SpatialObject* and is disjoint with the class *geo:Feature*. Moreover, two properties are defined for associating geometries with features. A property *geo:hasGeometry* is used to link a feature with a geometry that represents its spatial extent. A given feature may have many associated geometries. Also, a subproperty of the *geo:hasGeometry*, called *geo:hasDefaultGeometry*, is used to link a feature with its default geometry. In addition, several properties for *geo:Geometry* are defined for describing geometry metadata, such as *geo:dimension, geo:coordinateDimension, geo:spatialDimension, geo:isEmpty, geo:isSimple, geo:hasSerialization*, where the *geo:hasSerialization* property is used to connect a geometry with its text-based serialization (e.g., its WKT serialization). Further, an RDFS datatype *geo:wktLiteral* is defined for representing geometry data in RDF based on WKT as defined by ISO 19125-1 Simple Features [43]. For example, "Point(-83.38 33.95)"^^*geo:wktLiteral* encodes a point geometry using the default WGS 84 (World Geodetic System 1984, the most common CRS for spatial data on the web) geodetic longitude-latitude spatial reference system for Simple Features 1.0. At the same time, a subproperty of the *geo:hasSerialization*, called *geo:asWKT* is defined to link a geometry with its WKT serialization. Similarly, an RDFS datatype *geo:gmlLiteral* is defined for representing geometry data in RDF based on GML as defined by Geography Markup Language Encoding Standard [70], and a subproperty of the *geo:hasSerialization*, called *geo:asGML* is defined to link a geometry with its GML serialization.
- *Geometry Topology Extension (relation_family, serialization, version)*. Defines topological query functions, which will be introduced in the subsequent query section.
- *RDFS Entailment Extension (relation_family, serialization, version)*. Defines a mechanism for matching implicit RDF triples that are derived based on RDF and RDFS semantics. For example, the ISO 19125-1 Simple Features [43] specification presents a geometry class hierarchy. It is straightforward to represent this class hierarchy in RDFS. That is, the hierarchy relationship between Polygon and Surface classes from Simple Features 1.0 can be represented as <:Polygon *rdfs:subClassOf* :Surface>.
- *Query Rewrite Extension (relation_family, serialization, version)*. Defines rules for transforming a simple triple pattern that tests a topological relation between two features into an equivalent query involving concrete geometries and topological query functions.

The following Figure 2 shows the GeoSPARQL RDF representation (RDF Turtle syntax) of an application-specific spatial data in Figure 1.

(2) W3C Basic Geo Vocabulary [123]

W3C Basic Geo Vocabulary [123] provides a basic RDF vocabulary for representing points, using WGS84 (World Geodetic System 1984) as a reference datum.

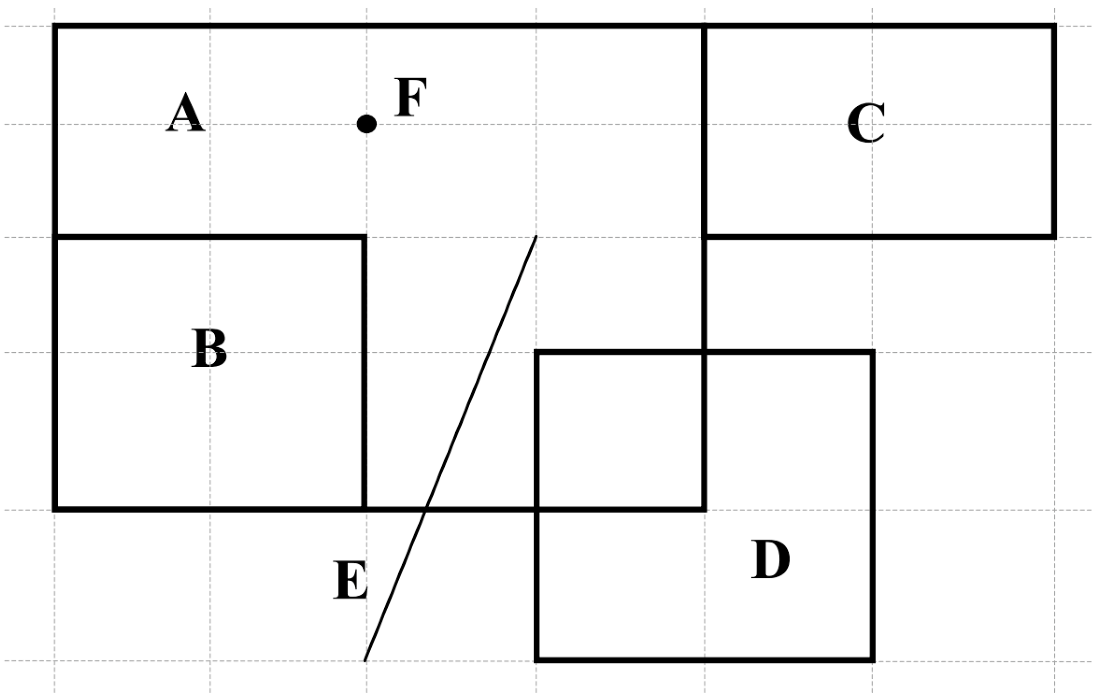
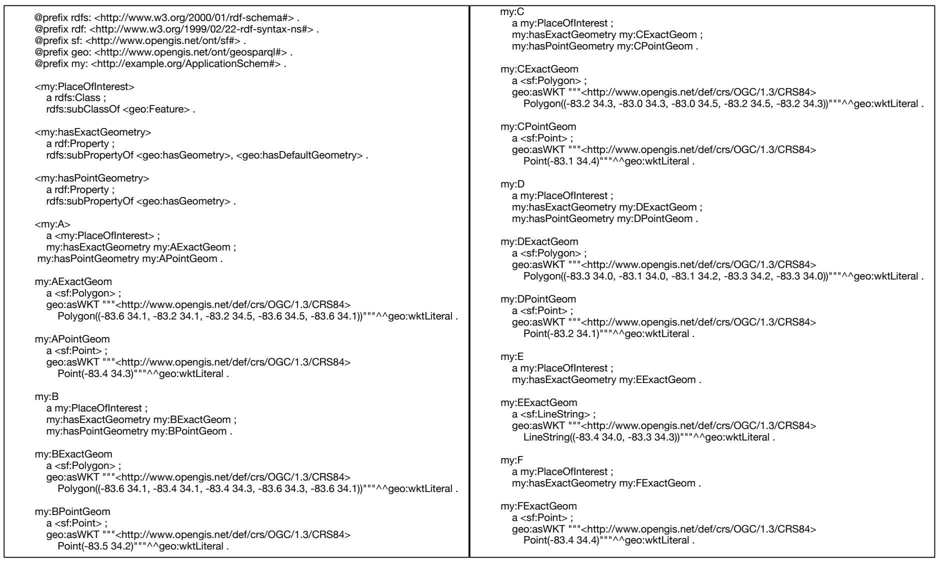


Figure 1 The application-specific spatial data



```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sf: <http://www.opengis.net/ont/sf#> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix my: <http://example.org/ApplicationSchem#> .

<my:PlaceOfInterest>
   a rdfs:Class ;
   rdfs:subClassOf <geo:Feature> .

<my:hasExactGeometry>
   a rdf:Property ;
   rdfs:subPropertyOf <geo:hasGeometry>, <geo:hasDefaultGeometry> .

<my:hasPointGeometry>
   a rdf:Property ;
   rdfs:subPropertyOf <geo:hasGeometry> .

<my:A>
   a <my:PlaceOfInterest> ;
   my:hasExactGeometry my:AExactGeom ;
 my:hasPointGeometry my:APointGeom .

my:AExactGeom
   a <sf:Polygon> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Polygon((-83.6 34.1, -83.2 34.1, -83.2 34.5, -83.6 34.5, -83.6 34.1))"""^^geo:wktLiteral .

my:APointGeom
   a <sf:Point> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Point(-83.4 34.3)"""^^geo:wktLiteral .

my:B
   a my:PlaceOfInterest ;
   my:hasExactGeometry my:BExactGeom ;
   my:hasPointGeometry my:BPointGeom .

my:BExactGeom
   a <sf:Polygon> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Polygon((-83.6 34.1, -83.4 34.1, -83.4 34.3, -83.6 34.3, -83.6 34.1))"""^^geo:wktLiteral .

my:BPointGeom
   a <sf:Point> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Point(-83.5 34.2)"""^^geo:wktLiteral .

my:C
   a my:PlaceOfInterest ;
   my:hasExactGeometry my:CExactGeom ;
   my:hasPointGeometry my:CPointGeom .

my:CExactGeom
   a <sf:Polygon> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Polygon((-83.2 34.3, -83.0 34.3, -83.0 34.5, -83.2 34.5, -83.2 34.3))"""^^geo:wktLiteral .

my:CPointGeom
   a <sf:Point> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Point(-83.1 34.4)"""^^geo:wktLiteral .

my:D
   a my:PlaceOfInterest ;
   my:hasExactGeometry my:DExactGeom ;
   my:hasPointGeometry my:DPointGeom .

my:DExactGeom
   a <sf:Polygon> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Polygon((-83.3 34.0, -83.1 34.0, -83.1 34.2, -83.3 34.2, -83.3 34.0))"""^^geo:wktLiteral .

my:DPointGeom
   a <sf:Point> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Point(-83.2 34.1)"""^^geo:wktLiteral .

my:E
   a my:PlaceOfInterest ;
   my:hasExactGeometry my:EExactGeom .

my:EExactGeom
   a <sf:LineString> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      LineString((-83.4 34.0, -83.3 34.3))"""^^geo:wktLiteral .

my:F
   a my:PlaceOfInterest ;
   my:hasExactGeometry my:FExactGeom .

my:FExactGeom
   a <sf:Point> ;
   geo:asWKT """<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      Point(-83.4 34.4)"""^^geo:wktLiteral .
```

Figure 2 The GeoSPARQL RDF representation of the spatial data in Figure 1

The vocabulary defines a class 'Point', whose members are points. Points can be described using 'lat (latitude)', 'long (longitude)' and 'alt (altitude)' properties, as well as with other RDF properties defined elsewhere. The 'lat' and 'long' properties take literal (i.e., textual values), each in decimal degrees. The 'alt' property is decimal metres about local reference ellipsoid.

The following is a simple example.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
_:a1234
   rdf:type geo:Point ;
   geo:lat "55.701" ;
   geo:long "12.552" .
```

Although the W3C Basic Geo vocabularies is widely used to describe WGS84 coordinates, it is does not allow the description of geometric shapes, such as the border of a country [96]. Similarly, as mentioned in [68], [7], The simple W3C Basic Geo vocabularies have limitations, for example the inability to specify different datums and coordinate systems, and were therefore not used in GeoSPARQL. The most existing geometry data encoded using these simple W3C Basic Geo vocabularies can easily be converted into GeoSPARQL representations.

Moreover, GeoRSS (www.georss.org) provided support for more geometric objects (such as lines, rectangles, or polygons) and GML application profiles. It is designed as a lightweight, community driven way to extend existing RSS (really simple syndication) feeds with simple geographic information. The main purpose of GeoRSS standard is not to represent geometry data in RDF, but to provide for encoding location in an interoperable manner so that applications can request, aggregate, share and map geographically tag feeds. The encoding may be expressed in a concrete form such as XML, RDF, etc. GeoJSON (https://geojson.org) is a geospatial data interchange format based on JavaScript Object Notation (JSON) and can encode a variety of shapes (points, polygons, etc.).

Note that, a further report is discussed by the W3C Geospatial Incubator Group (GeoXG) [124] to discusses the need for updating the W3C Basic Geo Vocabulary [123] following the GeoRSS feature model , and to allow for the description of points, lines, rectangles, and polygon geometries and their associated features.

(3) NeoGeo [96], [97]

By analyzing the representation of several existing geospatial datasets, Salas et al. [96] distill an integration vocabulary NeoGeo which covers the core set of classes and properties in existing data. They identify four main kinds of representation (point, bounding box, points in lists, points using a single property and literals) to support typical geometric objects (Geometry, Point, MultiPoint, LineString, MultiLineString, LinearRing, Polygon, MultiPolygon, GeometryCollection), as well as WKT serialization.

In the NeoGeo Vocabulary, everything is represented as an RDF resource, maximizing the vocabulary's expresivity. For example, the list of nodes which form a polygon is an RDF Collection containing an RDF resource for each node. That is, the polygons and lines are represented with an RDF collection of W3C Basic Geo points [123] . The basic classes are *spatial:Feature* and *geom:Geometry*, connected via the *geom:geometry* property. In more detail, a geometry vocabulary (http://geovocab.org/geometry) for describing geographical regions in RDF is defined, which including several classes (e.g., LineString and Polygon) and several properties (e.g., boundary and exterior). Also, a spatial vocabulary (http://geovocab.org/spatial) for describing RCC-8 topological relations between features is defined, which including a class Feature and several properties (e.g., DC(x, y) for 'x is disconnected from y' and EC(x, y) for 'x is externally connected with y').

The following example shows that *Karlsruhe* is a *spatial:Feature,* and related to http://example.org/geo/kageo, which is a *geom:Geometry.* And the *Argentina* is defined to be a tangential proper part (RCC-8 TTP) of *South America*, and is externally connected (RCC-8 EC) to other countries like *Uruguay*, *Brazil*, *Paraguay*, *Bolivia* and *Chile.*

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix geom: <http://geovocab.org/geometry#> .
@prefix spatial: <http://geovocab.org/spatial#> .
@prefix : <http://example.org/#> .

:Karlsruhe rdf:type spatial:Feature .
:Karlsruhe geom:geometry <http://example.org/geo/kageo> .
<http://example.org/geo/kageo> rdf:type geom:Geometry .

:Argentina spatial:TPP :South_America .
:Argentina spatial:EC :Uruguay, :Brazil, :Paraguay, :Bolivia, :Chile .
```

Moreover, they publish two geospatial datasets NUTS and GADM, and also develop an algorithm for finding equivalences for geometric shapes across multiple datasets for integration and mapping of multiple datasets.

(4) Spatial Literals in RDF [14]

Brodt et al. [14] represent spatial features in RDF as literals of a complex geometry type and store them in RDF literals. The literals contain the spatial features expressed in the Well-Known Text (WKT) format, as standardized in the OpenGIS Simple Features Specification [71]. The literals carry a type URI which denotes that the literal is to be processed as a spatial feature rather than as an ordinary string. For example, the position of a gas station can be represented as <:Station1234 :locatedAt "POINT (48.77 9.18)"^^geordf:geography>, where *geordf* is the namespace prefix of http://example.org/geo.

Note that, the approach does not assign a IRI to a spatial feature, so that it cannot be directly referenced or augmented by metadata. Still, the place resources would keep all information related to geographic coordinates in the single RDF statement which carries the spatial literal. All further statements related to the place resource are "ordinary" RDF and can be designed in any ontology. The following example shows the gas station using a place resource.

```
:Station1234   :place        :Place9274.
:Place9274     :clearName    "123 Main street".
:Place9274     :source       "GPS sensor 2000".
:Place9274     :locatedAt    "POINT (48.77 9.18)"^^geordf:geography.
```

(5) GeoRDF [35]

GeoRDF was intended as an RDF compatible profile for geographic information (points, lines and polygons). The representation of the points is similar to the W3C Basic Geo Vocabulary [123]. The lines and polygons can be represented by 'simple' profile or 'complex' profile. The 'simple' profile is based on Mikel's extension of the geo schema (http://mapufacture.com) and uses lists of comma-separated lat, long pair values to describe both lines and polygons as shown in the following example. The 'complex' profile is based on Chris Goad's more complex formulation (http://fabl.net/lib/geometry/1.1/revision1.html) and uses an rdf:Seq to express a list of points that are a line, a polyline or a polygon. Its two vocabularies (RDFGeom and its 2d companion RDFGeom2d) provide a framework that is extensible via subclassing to all kinds of geometric data.

```
<geo:line>50,-0.2,100 54,3,442 59,2.56,0</geo:line>
<geo:polygon>30,-120 30,-100 20,-100 30,-120</geo:polygon>
```

(6) RDF^i [62], [63]

Nikolaou et al. (2012 & 2013) [62], [63] address the problem of representing and querying *incomplete geospatial* information in RDF using the framework named RDF^i (where "i" stands for "incomplete"), which extends RDF with the ability to represent property values that *exist, but are unknown or partially known*, using a quantifier-free formula of a first-order *constraint language* $\mathcal{L}$. Formally, RDF^i extends the concept of an RDF graph to the concept of an RDF^i database which is a pair (G, ϕ) where G is an RDF graph possibly containing triples with *e-literals* in their object positions, and ϕ is a quantifier-free formula of $\mathcal{L}$. RDF^i extends RDF with the ability to define a new kind of literals for each datatype. These literals are called *e-literals* ("e" comes from the word "existential"). It should be noted that *e-literals* are allowed to appear only in the object position of triples in RDF^i.

For example, a real world situation that corresponds to a hotspot would be to state that there is a geographic region with *unknown* exact coordinates where a fire is taking place, and that region is included in a known 3km by 3km rectangle. Such case can be represented by RDF^i as shown in the following example.

```
hotspot1 rdf:type Hotspot .
fire1 rdf:type Fire .
hotspot1 correspondsTo fire1 .
fire1 occuredIn _R1 .
_R1 NTPP "x ⩾ 6 ∧ x ⩽ 23 ∧ y ⩾ 8 ∧ y ⩽ 19"
```

The fire *fire1* (red area of Figure 3) is asserted to have taken place inside region _R1, which is an *e-literal* datatype and is asserted to be inside the rectangle formed by the points (6,8) and (23,19) (rectangle P of Figure 3). This is stated with a constraint expressed in the language PCL (Polygon Constraint Language), a first-order constraint language that allows to represent topological properties for polygons. NTPP is the "non-tangential-proper-part" relation of RCC-8 [88]. In general, constraints in PCL can be used to express qualitative and quantitative spatial information about regions in $\mathbb{Q}^2$.

3.2 Spatial RDF Storage, Querying, and Performance Assessment

Facilities for the storage and querying of large quantities of spatial data are important to many applications, and thus many geographic information systems and spatial databases are developed [76]. After RDF is employed to represent massive spatial application data as introduced in Section 3.1, there has been significant community and research interest in storing and querying geospatial data based on RDF management techniques. Accordingly, some spatial RDF storage and querying techniques are proposed.

For storing and querying spatial RDF data, some *native* architectures and *hybrid* architectures are developed. As mentioned in [87], the *native* architecture needs to implement the *data indexing* system, the *query engine*, and the *physical storage.* The hybrid architecture uses existing systems as sub-components for the processing needed. Note that, the *data indexing* has vital roles in the performance of data insertion and query. The *query engine* may include a query parser (parsing the input query), a query optimizer (determining the most efficient way to execute a given query by considering all the possible query execution plans), and query executor (processing and returning the query results). The *physical storage* can be classified into two types, namely *native* (persistent disk-based and main memory-based) and *non-native storage* (relies on the relational database management system RDBMS to store RDF data permanently). Moreover, as also mentioned in [49] and [77], there are some primary types of spatial queries that should be covered, e.g., location queries, range queries, spatial joins, and nearest neighbor search.

In addition, the performance study and assessment of spatial RDF querying and storage techniques is crucial to select a suitable RDF engine for some applications and promote the development of RDF data management techniques. In the following, the subsection 3.2.1 will first summarize and compare the existing spatial RDF storage and querying techniques. Then, the subsection 3.2.2 will summarize the existing work about the performance assessment of spatial RDF querying and storage techniques.

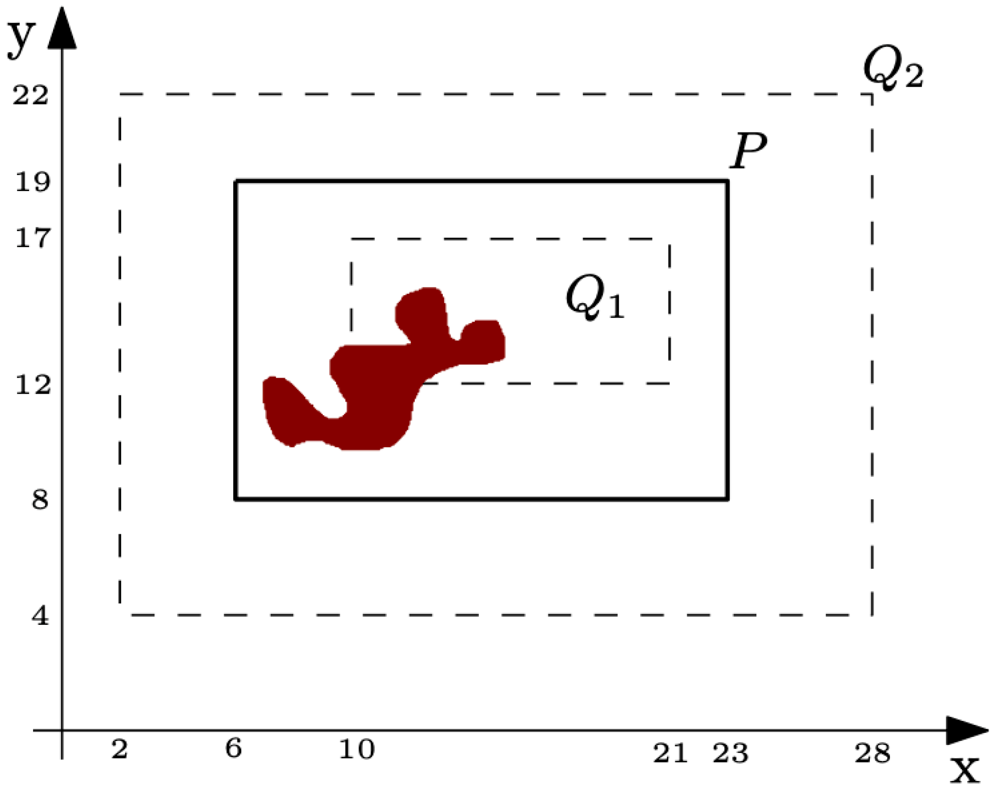


Figure 3 Rectangles mentioned in the example of RDF^i

Table 2 Qualitative comparison of spatial RDF storage and querying techniques

Query	**Query language**	**Main idea**	**Supported geospatial information**	**Index**	**Storage and Implementation**
SPARQL spatial filter query [14]	SPARQL extension	expressing spatial predicates as SPARQL filter functions	Simple Features spatial data types [71] with WKT serialization	R-Tree index of the libspatialindex C++ library	based on RDF-3X [61]
Geo-Store SPARQL Query [126]	SPARQL extension	using three spatial query filters to extend SPARQL	geographic objects with WKT serialization	ID-based indexing scheme	based on RDF-3X [61]
S-Store location-oriented query [127]	SPARQL extension	a spatial query is a list of spatial triple patterns with some spatial filter conditions	location objects modeled as the W3C Geo Vocabulary format [123]	SS-tree	
GeoSPARQL [68], [7]	SPARQL extension	defining a set of spatial functions for use in SPARQL queries	most OGC and Simple Features spatial data types [43], [71]	The indexes in the Parliament API [85]	based on Parliament [85]
SPARQL for querying incomplete geospatial information [62]	SPARQL extension	extending SPARQL FILTER expressions with first-order constraint language for constraining the values of spatial variables	The incomplete geospatial information (i.e., exist, but are unknown or partially known)		
SRX: Spatial RDF-3X query [58], [115]	SPARQL extension	augmenting the original join query graph of a SPARQL expression to include binding of spatial variables via spatial join conditions	RDF datasets with geospatial information, e.g., LinkedGeoData [59] and YAGO [41]	R-Tree, B$^+$-tree	based on RDF-3X [61]
STREAK: Top-k SPARQL Query [55]	SPARQL extension	extending SPARQL FILTER and LIMIT to support queries with spatial filters and top-k ranking functions	RDF datasets with spatial information, e.g., YAGO [41] and LinkedGeoData [59]	S-QuadTree	based on Quark-X [56] which is built on top of RDF-3X [61]
Top-k semantic place (kSP) search [103]	Keyword query	a query location and a set of query keywords as parameters and returns k semantic places	RDF datasets with place information, e.g., DBpedia [57] and YAGO [41]	R-tree, reachability index TFlabel, and the inverted index	memory-resident and implemented in Java
Collective spatial keyword query [47]	Keyword query	a spatial knowledge base and a query as inputs, and returns top-k valid groups	RDF datasets with place information, e.g., DBpedia [57] and YAGO [41]	R*-tree and the inverted index	memory-resident and implemented in Java
Semantic Region (SR) Keyword Search on RDF [131]	Keyword query	a spatial range and several query keywords as arguments and returns the qualified semantic region	RDF datasets with place information, e.g., DBpedia [57] and YAGO [41]	R*-tree, reachability index TFlabel, and the inverted index	memory-resident and implemented in Java
Top-k diversified semantic place (kDSP) search [16]	Keyword query	generalizing a kSP [103] query by combining relevance and diversity	RDF datasets with place information, e.g., DBpedia [57] and YAGO [41]	R-tree, reachability index TFlabel, and the inverted index	memory-resident and implemented in Java

3.2.1 Spatial RDF Storage and Querying

The following Table 2 first summarizes and compares the spatial RDF storage and querying techniques according to their support for spatial data management.

(1) SPARQL spatial filter queries [14]

In order that the data management systems are capable of querying large amounts of RDF data and supporting spatial query predicates, Brodt et al. (2010) [14] model spatial features as typed complex literals, and define spatial predicates as filter functions in SPARQL. The literals contain the spatial features expressed in the Well-Known Text (WKT) format, as standardized in the OpenGIS Simple Features Specification [71] as mentioned in Section 3.1. On this basis, they express spatial predicates as SPARQL filter functions on this type. This makes it possible to use W3C's standardized SPARQL query language as-is, i.e., without any modifications or extensions for spatial queries.

In more detail, the approach to express spatial query predicates in SPARQL uses filter functions that are identified by an IRI. It uses the functions of the OpenGIS Simple Features Specification [71] and defines an IRI for each of them. The filter functions act on variables which bind a spatially typed literal as mentioned in Section 3.1. Geometry constants to compare the value of the variable are specified as spatially typed literals, too; they are given in the well-known text (WKT) format and carry an IRI denoting the spatial type. They also implement a *spatial index* based on the R-Tree index of the libspatialindex C++ library (http://sourceforge.net/projects/spatialindexlib). The following example shows a fully standard-compliant SPARQL query to find all gas stations within a given area that specifies the spatial query predicate as a filter function (where *geordf* and *geo* are the namespace prefixes of http://example.org/geo and http://www.w3.org/2003/01/geo/wgs84_pos, respectively).

```
SELECT ?x WHERE {
?x rdf:type :GasStation.
?x geordf:hasGeography ?geo.
FILTER geo:within (?geo, "POLYGON((48.765 9.175,
48.775 9.175, 48.775 9.185, ...))"^^geordf:geography) }
```

They finally implement the approach based on the architecture of the triple store RDF-3X [61] and make some modifications to RDF-3X with spatial functionality. The evaluation shows that some of the modifications do create some extra overhead for queries selecting very large amounts of spatial features, but excellent performance for most common spatial query types.

(2) Geo-Store: spatially-augmented SPARQL query [126]

Wan et al. (2012) [126] propose Geo-Store, a novel spatially-augmented SPARQL evaluation system. By using spatial query filters to extend the standard SPARQL query language, Geo-Store can handle complex spatial queries with common spatial constraints. These spatial filters are designed based on spatially aware mapping (SAM) scheme. With SAM, spatial data can be pre-processed and encoded with their Hilbert values by using the Hilbert curve to accelerate spatial query evaluation without any compromise in accuracy.

The major components of Geo-Store include a spatially aware mapping (SAM) module, an internal processing (IP) module, a spatial query analyzer (SQA) module, and a dictionary decoding (DD) module. By the SAM module, Hilbert curves are generated and Hilbert values (integers) of the literals containing coordinate information (latitude/longitude) are calculated and then appended to the original triple data source as location indicators. The IP module maintains in memory all six possible permutations of <*s p o*> in six separate indices (i.e., SPO, SOP, OSP, OPS, PSO, and POS indices), and all six indices consist of integer IDs, which are assigned by SAM. Further, spatial queries are expressed by using spatial query filters in the SQA module. If a spatial filter is detected by SQA, the SQA will employ SAM to identify a list of Hilbert values (serving as location indicators) that are involved with the filter. Then, the SQA will translate the derived Hilbert values into query patterns and append them as clauses to the original SPARQL query expression. For multiple spatial query filters, the SQA will apply intersection operations to find the geographic locations satisfying all the spatial filters. Finally, the DD module transforms the obtained IDs back to their original literals as the query results.

Geo-Store supports three spatial filters, named window filter, range filter, and nearby filter. These filters operate on geographic objects (and variable bindings) expressed in the Well-Known Text (WKT) format.

- window filter: specifies a rectangular geographic constraint on a SPARQL variable. The syntax of this filter is *within(?var, xmin, ymin, xmax, ymax)*. The following gives an example of a SPARQL query with a window filter. The query defines three query variables (?e, ?name, and ?location) and returns the *name* and *location* of records whose locations satisfy the window filter.

```
SELECT ?name ?location WHERE {
?e <name> ?name.
?e <coordinates> ?location.
within(?location, 32.5955, -85.4909, 32.6122, -85.4739) }
```

- range filter: specifies a constraint region that is within a certain distance of a geographic point. This filter essentially defines a circular region around a central point. There are two variations of this filter, the first takes literal coordinates of the center, and the second takes a variable as the center.
- nearby filter: sorts the bounded values based on the Euclidean distance between the bounded values and a center point, and returns the closest k bindings.

The Geo-Store system is built on top of RDF-3X [61], a scalable RDF triple store. Moreover, it provides a graphical query interface for users to easily compose spatial and non-spatial constraints to construct SPARQL queries.

(3) S-store: location-oriented spatial SPARQL query [127]

In order to realize the location-oriented spatial information query, Wang et al. (2013) [127] propose spatial SPARQL (a variant of SPARQL language) for querying RDF data with spatial features. In their work, a spatial triple pattern is a four-tuple ¡s, p, o, l¿, where s, p, o and l represent for subject, predicate, object and location respectively, and the "location" feature is modeled as the W3C Geo Vocabulary format [123]. In this case, a spatial query is a list of spatial triple patterns with some spatial filter conditions. If there's no spatial filter condition, the spatial query is reduced to a traditional SPARQL query. For example, a location-oriented spatial query about a person who died in a popular place near coordinates (48.39841,9.99155) can be expressed as follows:

```
SELECT ?x WHERE {
?x diedIn ?y.
?y rdf:type PopulatedPlace .
Filter dist(sl(?y), (48.39841, 9.99155)) <1 }
```

Where $sl(?y)$ denotes the spatial label of variable $?y$. Besides, *dist(a, b)* $< r$ denotes the Euclidean distance between a and b should below the threshold r, where a and b should be a specific location or a variable. If either of a and b is a constant, the query is called a range query. If both of a and b are variables, the query is called a spatial join query. Note that a spatial query can be a range query and a spatial join query at the same time.

Further, they classify the entities into two categories: the spatial entities and the non-spatial entities. Based on these two categories, they define a tree-style index structure (called SS-tree), which is a hybrid tree-style index integrating the semantic features and the spatial features. They finally evaluate their approach on YAGO2 [41].

(4) GeoSPARQL [68], [7]

In order to solve the problem of spatial inference with the amount of spatial context data, a SPARQL extension language, called GeoSPARQL, was defeloped in [68], [7] for unifying the data access of the geospatial Semantic Web. GeoSPARQL is also the OGC (Open Geospatial Consortium) standard. GeoSPARQL defines a vocabulary for representing geospatial data in RDF as mentioned in Section 3.1, and it defines an extension to the SPARQL query language for processing geospatial data. Here, we will introduce the query part of GeoSPARQL in detail.

The GeoSPARQL specification contains three main components:

1. The definition of a vocabulary to represent features, geometries, and their relationships as mentioned in Section 3.1.
2. A set of domain-specific, spatial functions for use in SPARQL queries.
3. A set of query transformation/rewrite rules, which allows for an additional layer of abstraction in SPARQL queries.

Regarding to the spatial functions for use in SPARQL queries, GeoSPARQL provides two different types of functionality. *First*, there are operator functions which take multiple geometries as predicates and produce either a new geometry or another datatype as a result. An example of this is the function `geof:intersection` (where *geof* is the prefix of http://www.opengis.net/def/function/geosparql/). This function takes two geometries and returns a geometry that is their spatial intersection. Other functions like `geof:distance` produce an `xsd:double` as a result. The *second* type of functionality is boolean topological tests of geometries. An example of the topological functions is `geof:sfContains`, which returns true if one geometry contains another geometry (where `sfContains` represents the relartion `contains` of Simple Features relations [43], [71]).

For example, given the spatial RDF data in Figures 1 and 2 in Section 3.1, the query "Find all features that feature `my:A` contains, where spatial calculations are based on `my:hasExactGeometry`." can be expressed as the following GeoSPARQL query:

```
SELECT ?f WHERE {
my:A my:hasExactGeometry ?aGeom .
?aGeom geo:asWKT ?aWKT .
?f my:hasExactGeometry ?fGeom .
?fGeom geo:asWKT ?fWKT .
FILTER (geof:sfContains(?aWKT, ?fWKT) && !sameTerm(?aGeom, ?fGeom)) }
```

In this example, the property `geo:asWKT` is defined to link a geometry with its WKT serialization as introduced in Section 3.1, and the `sameTerm` returns TRUE if *term1* and *term2* are the same RDF term. After running the query above, the features `my:B` and `my:F` in Figure 1 will be returned.

Moreover, They also have worked to update the open source triple store Parliament to support GeoSPARQL. In addition, some work attempted to further extend or use the GeoSPARQL for special applications. The work in [135] proposes a parallel approach for improving the query performance of geospatial ontology in a GeoSPARQL query by separating spatial and non-spatial components. The work in [137] proposes a BimSPARQL query language by extending SPARQL functions for querying Industry Foundation Classes (IFC) building 3D geometry data. Several simpler approaches in [133], [134] also discuss the need for adding topological predicates to SPARQL, and the OGC Simple Features relations [43], [71] and a subset of geometries are used as the basis for their ontologies.

(5) SPARQL for querying incomplete geospatial information [62]

As mentioned in Section 3.1, an RDF extension model, called RDF^i is proposed for representing the incomplete geospatial information (i.e., exist, but are unknown or partially known) [62], [63]. Further, for querying the incomplete geospatial information, Nikolaou et al. (2012) [62] define an extension language of SPARQL. For example, the query "Find all fires that have occurred in a region which is a non-tangential proper part (NTPP of RCC-8 [88]) of rectangle Q_1 of Figure 3" may be expressed by the extension of SPARQL as follows:

```
SELECT ?F WHERE {
?F rdf:type Fire .
?F occuredIn ?R .
FILTER ( NTPP(?R, "x ⩾ 10 ∧ x ⩽ 21 ∧ y ⩾ 12 ∧ y ⩽ 17") )}
```

From the example, they extend FILTER expressions of standard SPARQL and allow to the expression of a first-order constraint language for constraining the values of spatial variables. More detailed definitions for the extension of SPARQL language can be found in [62].

(6) SRX: Spatial RDF-3X query [58], [115]

For efficiently supporting spatial queries, such as range selections (e.g., find entities within a given range) and spatial joins (e.g., find pairs of entities whose locations are close to each other), Liagouris et al. (2014) [58] propose an extension for RDF stores that supports efficient spatial data management, including providing effective coding schemes for entities with spatial locations, introducing dynamic spatial filters and spatial join algorithms, and giving some optimizations that minimize the overhead of geometry and dictionary accesses.

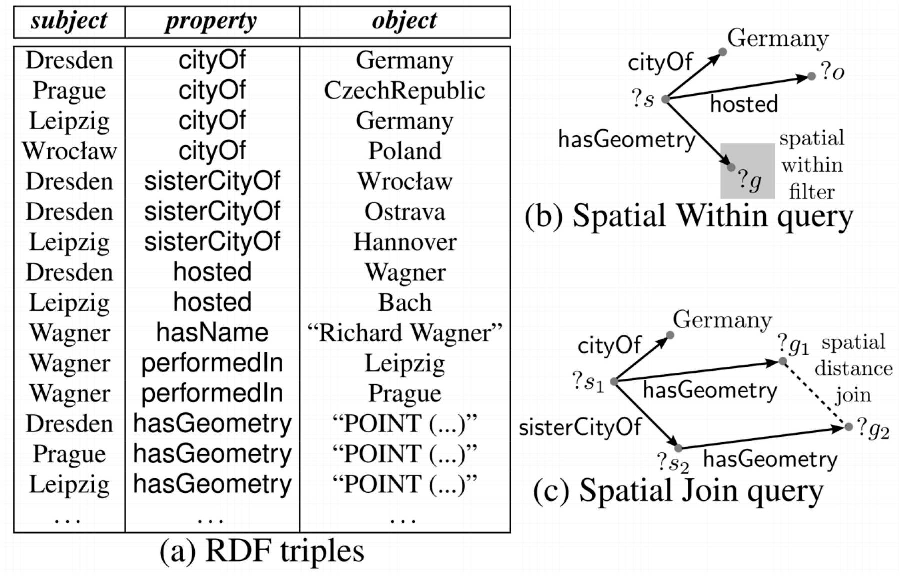
In more detail, they present the details of a system by extending the open-source RDF-3X engine [61] in the following several aspects (where RDF-3X encodes all values that appear in <*s p o*> triples by identifiers with a help of a dictionary and models the RDF knowledge base as a single, long table of ID triples. A SPARQL query can then be modeled as a multi-way join on the triples table.):

- Index Support for Spatial Queries. The system includes a spatial index R-tree for the geometries associated to the spatial entities.
- Spatial Encoding of Entities. The system encodes spatial approximations inside the IDs of entities (i.e., resources) associated to spatial locations and geometries.
- Spatial Join Algorithms. The system designs spatial join algorithms tailored to the above encoding scheme.
- Spatial Query Optimization. The system augments the original join query graph of a SPARQL expression to include binding of spatial variables via spatial join conditions. That is, the `Filter` clause in a SPARQL query may include one or more *spatial predicates*. The system considers only `WITHIN` range predicates (for spatial selections in Figure 4) and `DISTANCE` predicates (for spatial joins in Figure 4). For example, an RDF knowldege base with spatial information and a corresponding SPARQL query including a spatial join predicate is shown in Figure 4.

They experimentally evaluate the system using two real datasets LinkedGeoData (LGD) [59] and YAGO2 [41]. The work is further extended in [115], where a system called SRX (Spatial RDF-3X) was implemented, which extends RDF-3X with support for three types of spatial queries: range selections and spatial joins as mentioned above, and spatial k-nearest neighbors (e.g., find the three closest entities from a given location).

(7) STREAK:Top-k SPARQL Queries [55]

Leeka et al. (2017) [55] present an RDF data management system called STREAK to process kinds of queries with spatial filters (including complex joins, higher-order relationships over spatially enriched datasets), along with top-k requirements over arbitrary user-defined ranking functions, expressed in the order by the `LIMIT` clauses. They consider the following SPARQL query structure:



subject	*property*	*object*
Dresden	cityOf	Germany
Prague	cityOf	CzechRepublic
Leipzig	cityOf	Germany
Wrocław	cityOf	Poland
Dresden	sisterCityOf	Wrocław
Dresden	sisterCityOf	Ostrava
Leipzig	sisterCityOf	Hannover
Dresden	hosted	Wagner
Leipzig	hosted	Bach
Wagner	hasName	"Richard Wagner"
Wagner	performedIn	Leipzig
Wagner	performedIn	Prague
Dresden	hasGeometry	"POINT (...)"
Prague	hasGeometry	"POINT (...)"
Leipzig	hasGeometry	"POINT (...)"
...	...	...

(a) RDF triples

```
Select ?s1 ?s2
Where ?s1 cityOf Germany . ?s1 sisterCityOf ?s2 .
        ?s1 hasGeometry ?g1 . ?s2 hasGeometry ?g2 .
Filter DISTANCE(?g1, ?g2) < "300km";
```

Figure 4 RDF spatial data and SPARQL spatial join predicate query in SRX

```
SELECT [projection clause]
WHERE [graph pattern]
FILTER [spatial distance function]
ORDER BY [ranking function]
LIMIT [top-k results]
```

Where the `FILTER` clause restricts solutions to those that satisfy the spatial predicates. In their work, they focus only on the use of `DISTANCE` predicate for distance joins. The `ORDER BY` clause helps establish the order of results using a user-defined ranking function. The `LIMIT` clause controls the number of results returned. Together, `ORDER BY` and `LIMIT` form the top-k result retrieval with ad-hoc user-defined ranking function.

In the STREAK system, they present a S-QuadTree index, a novel RDF-specific extension of Quadtree [27] (a popular in-memory spatial index structure). S-QuadTree stores all the spatial entities in the knowledge graph, and embeds the associated soft-schema information. Further, they propose a spatial-join algorithm and an adaptive query plan generation algorithm called Adaptive Processing for Spatial filters (APS), that switches with very low plan-switching overhead, to select between alternate query plans, based on a novel spatial join cost model.

They implement these features within their previous framework Quark-X [56], a top-k SPARQL processing system based on RDF-3X [61]. They evaluate their approach on LinkedGeoData (LGD) [59] and YAGO2 [41].

(8) Spatial Keyword Search on RDF data [16], [47], [103], [131]

Considering that RDF data is traditionally accessed using SPARQL as mentioned in Section 3.1. Some work stated that SPARQL may be not friendly to common users, since query issuers need to understand the language itself and to be aware of the data domain, and thus it limits data access mostly to domain experts. This leaves room for a keyword search model on RDF data. Therefore, some work investigated the spatial keyword search on RDF data.

Top-k relevant semantic place (kSP) search [103]. Shi at al. (2016) propose a novel way of searching spatial RDF data, namely top-k relevant Semantic Place retrieval (kSP) query. It takes a query location and a set of query keywords as parameters and combines keyword-based and location-based retrieval. The query returns k semantic places that are subgraphs rooted at place entities that have associated locations.

Semantic Region (SR) search [131]. Wu et al. (2020) propose a generalization of semantic place retrieval, namely semantic region (SR) retrieval. Specifically, an SR query takes a spatial range and a set of query keywords as arguments and returns the qualified semantic region that minimizes a scoring function. The semantic region is composed of a subgraph $T(r, P)$ connecting a set of places that are in the query spatial range and the so-called keyword-relevant paths of the query keywords. The scoring function considers both the graph proximity of the occurrences of query keywords in the RDF graph to the places and the graph proximity among the places. An SR query aims to retrieve multiple places that are spatially close to the query location such that each place is relevant to one or more the query keywords. An algorithm and optimization techniques are proposed for the efficient processing of SR queries. The proposed techniques are evaluated on DBpedia [57] and YAGO [41], two large real RDF data sets. The results show that applying all techniques enables processing SR queries efficiently.

Top-k diversified semantic place (kDSP) search [16]. Cai et al. (2020) introduce diversified spatial keyword search on RDF data. Their framework enables a trade-off between relevance and diversity. Namely, the output places, in addition to being relevant to the query, should minimize the number of common nodes in their subgraphs and should have diverse locations w.r.t. direction. To this end, they propose a top-k diversified semantic place (kDSP) query, which generalizes the above kSP [103] query by combining a relevance function to the query and a diversity function on the set of query results that considers their relative location and content. Note that, all of the keyword search approaches mentioned above use the index techniques such as R-tree, reachability index TFlabel, and the inverted index.

Collective spatial keyword query [47]. Jin et al. (2018) stated that (kSP) query [103] aims to find a subtree that is rooted at a place vertex, covers all query keywords, and minimizes a ranking function. But user requirements may not be satisfied by a single subtree in some application scenarios. A group of subtrees should be combined together to collectively cover the query keywords. To this end, they propose and study a novel way of searching on a spatial knowledge, namely collective spatial keyword query on a knowledge base (CoSKQ-KB). It takes a spatial knowledge base and a query as inputs, and returns top-k valid groups w.r.t. a ranking function. A valid group is composed of several semantic places, i.e., the subtrees rooted at place vertices, and the member semantic places collectively cover all the query keywords. A ranking function is designed to measure the spatial closeness as well as the semantic keyword relevance of a valid group w.r.t. a query. The valid group with a lower ranking score has a higher chance to satisfy a user's requirement. Further, they propose two algorithms for CoSKQ-KB, BCK and iSCK. BCK employs efficient pruning techniques to reduce search spaces. iSCK improves BCK with carefully designed dynamic bounds to terminate iterations earlier. Also, the R*-tree and the inverted indexes are used and the empirical experiments on DBpedia [57] and YAGO [41] are conducted to show the efficiency and effectiveness of the proposed algorithms.

(9) Some early spatial RDF storage and querying techniques

As also mentioned in Athanasiou et al. (2012) [3], Battle et al. (2012) [7] and Patroumpas et al. (2014) [77], there are several early spatial RDF storage techniques, including:

- *Parliament* [85]: a high-performance triple store and reasoner designed for the Semantic Web. The storage structure consists of a resource table, a statement table, and a resource dictionary. It also includes a rule engine for the inference, and currently supports RDFS plus parts of OWL. Parliament is compatible with the RDF, RDFS, OWL, SPARQL. *With respect to the geospatial RDF*, it supports GeoSPARQL standards, allowing GeoSPARQL data to be indexed and provides a query engine that supports GeoSPARQL queries using standard R-trees index.

- *Virtuoso Universal Server* [122]: a middleware and database engine hybrid. For storing RDF data, it implememts a quad GSPO (i.e., Graph, Subject, Predicate, Object), and all quads are stored in one table that relies on an underlying RDBMS (relational database management system). SPARQL is embedded and translated into SQL for querying RDF data stored in the database. *With respect to the geospatial RDF*, it supports the W3C Basic Geo Vocabulary points [123] as mentioned in Section 3.1. Topological relations can be queried but are limited to some built-in predicates (st_intersects, st_contains, st_within). Note that, spatial types and functions are limited in Virtuoso and currently not compliant with GeoSPARQL.
- *GraphDB (former OWLIM)* [36]: a family of semantic repositories rebranded as GraphDB that intend to offer scalability, loading and query evaluation performance for native RDF engines. It is compatible with all the common RDF syntaxes (XML, N3, N-Triples, Turtle, TRIG, TRIX) and supports the SPARQL, and also includes an inference engine. *With respect to the geospatial RDF*, it supports 2-dimensional points that use the W3C Basic Geo Vocabulary [123] with coordinates in WGS84, and supports the SPARQL to query points within ad-hoc rectangles, polygons and circles and to compute distances (in kilometers) between points.
- *uSeekM* [117]: an add-on library for Semantic Databases (also known as Triple-stores, Quad-stores, or RDF-databases) that use the Sesame (an open-source framework for storing, querying, and analysing RDF data, now is RDF4J project [110]) Java interface. uSeekM adds powerful indexing and querying capabilities to Semantic Databases, and provides integrations with other tools and frameworks. Most of its functionality is provided through wrappers. *With respect to the geospatial RDF*, it supports all OGC geometry types specified in the OGC Simple Features Implementation Specification for SQL and most operations in the GeoSPARQL standard.
- *Stardog* [111]: a commercial knowledge graph product that supports parsing, storing, inferencing, and querying RDF data. It supports SPARQL 1.1 and both ontological and rule-based reasoning with a query rewriting strategy. It supports a few GeoSPARQL query functions.

3.2.2 Performance Assessment of Spatial RDF Storage and Querying Techniques

With the widely use of RDF for representing and processing spatial data, some spatial RDF querying and storage framework are proposed as mentioned above. Accordingly, some work are developed for assessing spatially enabled RDF querying and storage techniques. Before the GeoSPARQL, in the early work Kolas et al. (2008) [49] extends LUBM (Lehigh University Benchmark, a well-known RDF benchmark) to include spatial entities and test the performance of spatially enabled RDF stores. Afterwards, the GeoKnow project (2012) [3] provides a survey and evaluation of spatially enabled RDF stores, and the selective stores in GeoKnow include Virtuoso, Parliament, OWLIM, uSeekM, and Strabon (a storage and query module for spatio-temporal RDF data as will be mentioned in the next Section 4), as well as spatially enabled relational databases, i.e., Oracle Spatial and PostgreSQL with PostGIS extension. Battle and Kolas (2012) [7] demonstrate the geospatial capacity of Parliament and successfully achieved lots of GeoSPARQL-compliant queries as mentioned in Section 3.2.1. Patroumpas et al. (2014) [77] perform an evaluation of the performance of the geospatial RDF stores.

The following Table 3 further summarizes and compares several recent work about the performance study and assessment of spatial RDF querying and storage techniques.

As shown in Table 3, Huang and Raza (2019) [40], [90] comprehensively assess and benchmark five popular and well-known spatially enabled RDF stores, i.e., RDF4J, GeoSPARQL-Jena, Virtuoso, Stardog, and GraphDB. Along with the investigation of features, the performance evaluation of these RDF stores has also been conducted by measuring the execution times of a

Table 3 Performance assessment of spatial RDF storage and querying techniques

	Assessing spatially enabled RDF storing and querying systems	**Aspects of assessment**	**Used datasets**
Geographica by Ioannidis, Garbis et al. (2013 and 2019) [29], [42]	Strabon, Parliament, uSeekM, Virtuoso, GraphDB, RDF4J, System X, System Y	The efficiency of primitive spatial functions and the performance of in real use case scenarios by using three different real workloads: *Real World Workload*: aims at evaluating the efficiency of basic spatial functions that a geospatial RDF store should offer; *Synthetic Workload*: aims at insuring that the evaluation of geospatial RDF stores can be performed in a controlled environment, so that the performance can be measured with great precision; *Scalability Workload* aims at discovering the limits of the systems under test as the number of triples in the dataset increase	Seven datasets for Real World Workload: GAG, CLC, OSM, GeoNames, DBpedia, Hotspots, Census; Several datasets for Synthetic Workload; Reference dataset for Scalability Workload
Huang and Raza (2019) [40], [90]	RDF4J, GeoSPARQL-Jena, Virtuoso, Stardog, GraphDB	GeoSPARQL compliance; geospatial query performance	ICOS Carbon Portal Metadata; datasets from the above Geographica benchmark
Bellini and Nesi (2018) [6]	Virtuoso, GraphDB, Oracle, Stardog, etc.	*With respect to the spatial aspect*, the RDF stores average query time of spatial queries are assessed	Km4City knowledge base
Quoca et al. (2019) [87]	Virtuoso, Stardog, Apache Jena, RDF4J, Strabon (a spatio-temporal RDF querying and storage system)	the data loading performance of the RDF stores over the linked sensor dataset (where the loading performance of spatial data is evaluated separately), and the query execution performance (where the geo-spatial search is evaluated)	the linked meteorological dataset (a subset of their GoT dataset)

set of GeoSPARQL queries. The evaluation query set consists of non-topological, spatial selection as well as spatial join queries adopted from a spatial benchmark, Geographica. It is encouraging to see the increasing maturity of the technical environment for the support of geospatial linked data, as well as the increasing compliance with GeoSPARQL compared with previous benchmarks.

Ioannidis et al. (2019) [42] revisit their previous Geographica benchmark in 2013 [29] which uses both real world and synthetic data to test the performance and functionality of geospatial RDF stores. In the new version Geographica 2, eight systems (Strabon, Parliament, uSeekM, Virtuoso, GraphDB, RDF4J, System X, System Y) are evaluated out of which six adequately support GeoSPARQL and two offer limited spatial support. Geographica 2 tests the efficiency of primitive spatial functions in RDF stores and the performance of the RDF stores in real use case scenarios by using the following three different real workloads:

- Real World Workload: aims at evaluating the efficiency of basic spatial functions that a geospatial RDF store should offer. In addition, this workload includes five real use case application scenarios (three typical application scenarios, namely "Geocoding", "Reverse Geocoding", "Map Search and Browsing"; two more sophisticated scenarios, namely "Rapid Mapping for Fire Monitoring", Computing Statistics of Geospatial Datasets").
- Synthetic Workload: aims at insuring that the evaluation of geospatial RDF stores can be performed in a controlled environment, so that the performance can be measured with great precision. It relies on a generator that produces synthetic datasets of various sizes and instantiates query templates that can produce queries with varying thematic and spatial selectivity. The synthetic workload generator produces SPARQL queries corresponding to spatial selection and spatial joins using the two query templates.
- Scalability Workload: aims at discovering the limits of systems under test as the number of triples in the dataset increase. Each system is tested against six increasingly bigger, proper subsets of the reference dataset. For each system-dataset combination they measure (*i*) the repository size on disk, (*ii*) the bulk loading time taking into consideration the limitations of loading methods of each system and (*iii*) the response time in three queries which represent a spatial selection, a heavy spatial join with high spatial selectivity and a lighter spatial join with lower spatial selectivity.

Bellini and Nesi (2018) [6] assess several well-known RDF stores, including Virtuoso, GraphDB, Oracle, and Stardog, for semantically enabled smart city services. The assessment model allows a full understanding of whether an RDF store is suitable to be used as a basis for Smart City modeling and applications. The benchmark is based on the Florence Smart City model, and the used datasets and tools are available online. *With respect to the spatial aspect*, the Spatial search at Basic level (meaning that it is able to index and retrieve only geolocated points) or at Advanced level (meaning that it is able to index complex shapes, for example polylines) are considered, and the RDF stores average query time of spatial queries are assessed.

Quoca et al. (2019) [87] present the empirical studies for selecting a suitable RDF engin for Linked Sensor Data. The selected RDF stores include Virtuoso, Stardog, Apache Jena, RDF4J, Strabon. And the evaluation coveres two aspects: the data loading performance of the RDF stores over the linked sensor dataset (where the loading performance of spatial data is evaluated separately), and the query execution performance (where the geo-spatial search is evaluated). Here, besides the overall query execution time, they also measure the execution performance of query parsing, query optimization and query execution. The query parsing time is calculated from the time the system retrieves the query string to the time the query algebra tree is generated. Similarly, the query optimization process is considered starting from the time the query tree and it is finished when an execution plan is delivered. For simplicity, any other run-time decisions are considered as part of the query execution rather than part of the query optimization. The query execution is finished when the last result has been received. The evaluation is conducted over the linked meteorological dataset, a sub-set of their GoT dataset (a unified integrated and live view for heterogeneous Internet of Things data sources using Linked Data, called the Graph of Things-GoT, see http://graphofthings.org/).

3.3 Spatial RDF Datasets and Management Tools

With the application of RDF in spatial data process, some spatial RDF datasets and management tools are developed. Several common **spatial RDF datasets** include:

- LinkedGeoData [59], [108]: a core for a Web of Spatial Open Data. It is an effort to add a spatial dimension to the Web of Data / Semantic Web. LinkedGeoData uses the information collected by the OpenStreetMap [73] project and makes it available as an RDF knowledge base according to the Linked Data principles. It interlinks this data with other knowledge bases in the Linking Open Data initiative.
- GeoNames [37]: a geographical database that covers all countries and provides Linked Data under a Creative Commons attribution license. It provides information for over eight million geospatial features. The data is exposed via an RDF webservice that exposes information on a per resource basis.
- DBPedia [57]: Data from Wikipedia is extracted into RDF. Many of these extracted entities are geospatial in nature (cities, counties, countries, landmarks, etc.) and many of these entities already contain some geospatial location information.

Moreover, there are other significant sources of geospatial data, but not specially available as an RDF knowledge base, including OpenStreetMap [73] (a world map that is built by people like you and can be freely used under an open license agreement), PostGIS [86] (a spatial database extender for PostgreSQL object-relational database, and supports for geographic objects allowing location queries to be run in SQL), other geospatial datasets (e.g., UN FAO Geopolitical Ontology, Uberblic, RAMON NUTS as mentioned in [96]).

Meanwhile, in order to efficiently manage spatial data in RDF, some spatial RDF data **management tools** are developed, including:

- *FAGI-gis* [30]: A tool for fusing geospatial RDF data. It is implemented in Java and Javascript and can be operated either via a command line utility or via a web-based graphical user interface. A user can use FAGI-gis's map based UI to perform several fusion actions on linked geospatial entities, considering both spatial and non-spatial properties of them. The input of FAGI-gis is two separate RDF datasets and a set of links that interlink entities from one dataset to the other. The output of the tool is a unified dataset, where the geometries of the linked entities, along with the rest, non-spatial properties, are fused according to selected fusion actions. The supported vocabularies for representing geospatial features include GeoSPARQL [68], [7] with WKT (Well-Known Text [43], [71] as mentioned in Section 2.2) serialization of geospatial features and Basic Geo [123].

- *TripleGeo* [78]: an open-source ETL (Extract-Transform-Load) utility that can extract geospatial features from various sources and transform them into triples for subsequent loading into RDF stores. TripleGeo can directly access both geometric representations and thematic attributes either from standard geographic formats or widely used DBMSs. It can also reproject input geometries on-the-fly into a different Coordinate Reference System, before exporting the resulting triples into a variety of notations. Most importantly, TripleGeo supports the OGC GeoSPARQL standard [68], [7], although it can extract geometries into other vocabularies as well. The development is based on open-source geometry2rdf library [38], but with notable modifications and substantial enhancements to meet interoperability needs in RDF stores. In detail, TripleGeo enables users to: (*i*) Extract spatial data from a source; (*ii*) Transform this data into a triple format and geometry vocabulary prescribed by the target RDF store; and (*iii*) Load resulting triples into the target RDF store.
- *geometry2rdf* [38]: a library for generating RDF files for geometrical information (which could be available in GML or WKT). The current version of the library works with Oracle geospatial databases and relies on Jena.
- *Shp2GeoSPARQL* [112]: an extension of geometry2rdf [38] which transforms Shapefiles into RDF in the cadastral domain. The geometries obtained from shp2GeoSPARQL are consistent with the GeoSPARQL geometry vocabulary. The most important advantage is that it supports the GeoSPARQL standard.
- *GeomRDF* [39]: a tool developed within the Datalift platform that transforms geospatial dataset from traditional GIS formats into RDF, and overcomes the limitations of some tools (geometry2rdf [38], TripleGeo [78], Shp2GeoSPARQL [112]). GeomRDF is based on a vocabulary that reuses and extends GeoSPARQL [68], [7] and NeoGeo [96], [97] so that geometries can be defined in any CRS, and represented both as structured geometries and GeoSPARQL standard compliant.
- *GeoTriples* [54]: an open-source tool for transforming geospatial data from their original formats into RDF. It allows the transformation of geospatial data stored in raw files (shapefiles, CSV, KML, XML, GML and GeoJSON) and spatially-enabled RDBMS (PostGIS and MonetDB) into RDF graphs using well-known vocabularies like GeoSPARQL [68], [7] and stSPARQL, but without being tightly coupled to a specific vocabulary. It is very efficient and scalable especially when its mapping processor is implemented using Apache Hadoop.
- *CHOROS* [20]: a qualitative spatial reasoning engine implemented in Java. CHOROS provides consistency checking and query answering over spatial data represented with the Region Connection Calculus (RCC) and the Cone-Shaped directional logic formalism (CSD). It supports all RCC-8 and CSD-9 relations as well as standard RDF/OWL semantic relations, both represented in RDF/OWL. As such, it can answer mixed SPARQL queries over spatial and non-spatial relation types.
- *generic-GML2RDF* [118]: a semi-automated conversion tool of geo-information models and GML data to RDF using Extensible Stylesheet Language Transformations (XSLT). The tool can transform any GML to RDF using generic transformation rules, and without knowing the data model.
- *Facete* [107]: a spatial data exploration and visualization application, which is a client-side JavaScript application and interacts with a server-side SPARQL endpoint. It is a Single Page Application (SPA) whose user interface comprises several UI components. Comprehensive accompanying materials including the open-source code, screencasts and documentation can be found at http://facete.aksw. org.
- *QB4SOLAP* [31]: allows users to publish multidimensional (MD) spatial data in RDF format. It extends the QB4OLAP (an extended vocabulary based on the RDF Data Cube Vocabulary QB, with a full MD metamodel, supporting Online Analytical Processing OLAP operations directly over RDF data with SPARQL queries) vocabulary with spatial concepts. It defines a number of spatial OLAP (SOLAP) operators over the defined QB4SOLAP cubes, allowing

spatial analytical queries over RDF data. It provides algorithms for generating spatially extended SPARQL queries for individual and nested SOLAP operators, which allows writing SOLAP queries without knowledge of RDF/SPARQL.

- *Oracle Spatial and Graph* [84]: uses a relational schema to store RDF data and processes SPARQL queries by translating them to equivalent SQL queries that are executed by the parallel SQL engine in Oracle Database. It is the only mainstream commercial RDF triplestore that supports significant components of the OGC GeoSPARQL standard [68], [7] .
- *Ontop-spatial* [10], [11]: the first OBDA system with geospatial support. It extends Ontop (a well-known Ontology-Based Data Access OBDA system) to enable the on-the-fly GeoSPARQL-to-SQL translation on top of geospatial databases. It is able to connect to a geospatial database (currently PostGIS or Spatialite) and create virtual geospatial RDF graphs on top of it, using ontologies and mappings. It supports the following components of GeoSPARQL [68], [7]: Core, Topology Vocabulary, Geometry topology extension, RDFS entailment and a subset of Geometry Extension.

Also, there are other case studies and tools for integrating, publishing, or linking geospatial RDF data. The work in [101] systematacially investigates the automating geospatial RDF dataset integration and enrichment, and provides concepts, approaches and cases that facilitate the integration and enrichment of information with other data types that are already present on the Linked Data Web with a focus on geo-spatial data. The case study in [116] attempts to publish linked spatial data and link to the external data sources, and the implementation is demonstrated with an example use case of linking a local road data to the external data source DBpedia [57]. The work in [130] focuses on the interaction of SDI (Spatial Data Infrastructure) and Semantic Web standards, and implements a spatial data fusion prototype using SDI and the Semantic Web standards for Linked Data, particularly RDF and SPARQL. The project [93] discusses the next generation of spatial data infrastructure based on the lessons from Linked Data implementations across Europe. The work in [98] develops a benchmark generator SPgen for spatial link discovery tools. It can check whether spatial link discovery systems can identify DE-9IM topological relations [21] between LineStrings and Polygons.

4 Spatio-Temporal RDF techniques

Besides the techniques of spatial RDF data management, there are lots of applications which are not only related to the spatial context but also associated with the temporal context. In particular, the geospatial objects are complex abstractions and they have parts and can be constituents of others, when the temporal dimension is further added, an even higher level of complexity arises. As a result, there has been significant community and research interest in managing spatio-temporal data based on RDF techniques.

In this section, we summarize and classify the existing spatio-temporal RDF techniques, which extend RDF in different forms to represent spatio-temporal information in RDF. We first summarize the *representation* issue of spatio-temporal RDF in Section 4.1. Then, we introduce the spatio-temporal RDF *storage* and *querying* in Section 4.2. Finally, we summarize the existing spatio-temporal RDF datasets and management tools in Section 4.3. Also, some examples and comparisons are provided.

4.1 Spatio-Temporal RDF Representation

There have been several attempts for representing spatio-temporal data based on RDF. Table 4 first summarizes and compares the spatio-temporal RDF representation forms.

(1) stRDF based on linear constraints [51]

Koubarakis and Kyzirakos (2010) [51] develop a constraint-based RDF extension called stRDF, with the ability to represent spatial and temporal data. stRDF follows the main ideas of constraint

Table 4 Summarization and comparison of spatio-temporal RDF representation

Model	**Syntax**	**Supported temporal data**	**Temporal types**	**Supported spatial geometries**
stRDF [51] based on linear constraints	quad (s, p, o, τ)	time points	valid time	spatial geometries, e.g., points, lines, line segments, polygons, k-dimensional unions of convex polygons possibly with holes
stRDF^i [50], [53]: a new version of stRDF [51]	(s, p, o)	XML Schema datatypes: xsd:dateTime, xsd:time, xsd:date, xsd:gYearMonth, xsd:gYear, xsd:gMonthDay, xsd:gDay, xsd:gMonth	user-defined time	Simple Features spatial data types [71] with WKT and GML serializations. Incomplete or indefinite spatial information
STT [102], [82]	$(s, p, o) : [t]$ or $(s, p, o) : [t_1, t_2]$	time points time intervals	valid time	spatial geometries based on the GeoRSS GML specification
gst-Store [128], [129]	$\langle s, p, o, l, t\rangle$	time points time intervals	valid time	range shapes (rectangle and circle)
YAGO2 [41]	SPOTL tuples (SPO + Time + Location)	time points in `YYYY-MM-DD` (ISO 8601)		location (a pair of a latitude and a longitude value)
stRDFS [138]	$(s, p{:} <t, l>, o)$	time points	valid time	location (a pair of a latitude and a longitude value)

databases [91], [52] and represents spatial and temporal objects as quantifier-free formulas in a first-order logic of *linear constraints.*

They define stRDF following three steps:

First, they define the formulae of constraints. Constraints will be expressed in the first-order language $\mathcal{L} = \{\leqslant, +\} \cup \mathbb{Q}$ over the structure $\mathcal{Q} = \langle\ \mathbb{Q}, \leqslant, +, (q)_{q\in\mathbb{Q}}\ \rangle$ of the linearly ordered, dense and unbounded set of the rational numbers, denoted by $\mathbb{Q}$, with rational constants and addition. Further, let S be a subset of $\mathbb{Q}^k$ (where k is a positive integer). S is called semi-linear if there is a quantifier-free formula $\phi(x_1, ..., x_k)$ of $\mathcal{L}$ where $x_1, ..., x_k$ are variables such that $(a_1, ..., a_k) \in S$ iff $\phi(a_1, ..., a_k)$ is true in the structure $\mathcal{Q}$. The symbol $\varnothing$ is used to denote the empty subset of $\mathbb{Q}^k$ represented by any inconsistent formula of $\mathcal{L}$.

Then, they define the model sRDF which extends RDF with the ability to represent spatial data. Assuming the existence of pairwise-disjoint countably infinite sets I, B and L that contain IRIs (International Resource Identifiers), blank nodes and literals respectively. sRDF also assumes the existence of an infinite sequence of sets C_1, C_2,... that are pairwise-disjoint with I, B and L. The elements of each C_k, k = 1, 2,... are the quantifier-free formulae of the first-order language $\mathcal{L}$ with k free variables, and C will represent the infinite union $C_1 \cup C_2 \cup \cdots$. Now, an sRDF triple (s, p, o) is an element of the set $(I \cup B) \times I \times (I \cup B \cup L \cup C)$. Being different from the RDF triple, the object of an sRDF triple can be a quantifier-free formula with constraints, i.e., a quantifier-free formula with k free variables, which is a finite representation of a (possibly infinite) semi-linear subset of $\mathbb{Q}^k$. Note that, the semi-linear subsets of $\mathbb{Q}^k$ can capture a great variety of spatial geometries, e.g., points, lines, line segments, polygons, k-dimensional unions of convex polygons possibly with holes. But they also cannot be used to represent other geometries that need higher-degree polynomials e.g., circles. The following example shows a sensor and its location using a conjunction of linear constraints.

```
ex:s1 rdf:type, ex:Sensor .
ex:s1 ex:has_location “x=10 and y=20”^^strdf:SemiLinearPointSet
```

Where a new kind of typed literals (i.e., quantifier-free formulae with linear constraints) is defined in sRDF, and the datatype of these literals is `strdf:SemiLinearPointSet`. The values of this datatype are typed literals (called spatial literals) that encode geometries using Boolean combinations of linear constraints in $\mathbb{Q}^2$. For example, $(x \geqslant 0 \wedge y \geqslant 0 \wedge x + y \leqslant 1) \vee (x \leqslant 0 \wedge y \leqslant 0 \wedge x + y \geqslant -1)$ is such a literal encoding the union of two polygons in $\mathbb{Q}^2$.

Final, they extend sRDF to stRDF so that thematic and spatial data with a temporal dimension can be represented, where stRDF extends sRDF with the ability to represent the valid time of a triple (i.e., the time that the triple was valid in reality). The time structure in stRDF is the set of rational numbers $\mathbb{Q}$ (i.e., time is assumed to be linear, dense and unbounded). The temporal constraints are expressed by quantifier-free formulas of the language $\mathcal{L}$, atomic temporal constraints are formulas of $\mathcal{L}$ of the following form: $x \sim c$, where x is a variable, c is a rational number and $\sim$ is $<, \leqslant, \geqslant, >, =$ or $\neq$. Temporal constraints are Boolean combinations of atomic temporal constraints using a single variable. Formally, an stRDF quad is (a, b, c, τ), where the temporal constraint τ defines the set of *time points* that the fact represented by the triple (a, b, c) is *valid* in the real world. An stRDF graph is a set of sRDF triples and stRDF quads.

(2) stRDF^i [50], [53]: a new version of stRDF [51]

Koubarakis et al. (2012) [50], [53] further propose a new version of the data model stRDF [51] as mentioned above. In the new version of stRDF, called stRDF^i, they opt for a more practical solution to the problem of representing geospatial data and use the OGC standards WKT and GML instead of linear constraints. The new version also allows the representation and querying of incomplete spatial information in RDF.

In stRDF^i, a new datatype `srdf:geometry` is defined for modelling geometric objects. The values of this datatype are spatial literals that encode geometric objects using the WKT format. Also, stRDF^i allows the presence of time in the object part of a triple (this is called user-defined time in the terminology of temporal databases [106]), where the objects can have the following XML Schema datatypes: xsd:dateTime, xsd:time, xsd:date, xsd:gYearMonth, xsd:gYear, xsd:gMonthDay, xsd:gDay, xsd:gMonth. The following example defines a hotspot, its reliability and its 2-dimensional geometry. The latter is a spatial literals of the new datatype `srdf:geometry` and is encoded in WKT. Note that, the *urn* is used to denote that the coordinates that give the point of the hotspot are projected to the Greek Geodetic Reference System (GGRS87).

```
ex:hotspot1 rdf:type noa:Hotspot .
ex:hotspot1 noa:hasReliability “100”^^xsd:decimal .
ex:hotspot1 noa:hasLocation “POINT(669062 4238286); urn:epsg:ggrs87”^^srdf:geometry .
ex:hotspot1 noa:hasDetectionTime “2009-08-17T17:30:01”^^xsd:dateTime .
```

Moreover, based on the constraint languages, stRDF^i also allows the representation and querying of *qualitative spatial relations* as they have traditionally been studied by the qualitative spatial reasoning community [92]. stRDF^i uses a new kind of literals to represent spatial regions about which the known information is *incomplete* or *indefinite*, e.g., region A is inside a known rectangle R but we do not know its exact geographic location, or region A is north of region B and it overlaps region C etc. More detailed introduction about the techniques for incomplete spatial information in RDF can be found in [8]. For example, the region “_region1” in the following example is a new kind of literal, called an *unkonwn literal*, which is asserted to be inside the recrangle formed by the points (0, 0) and (3000, 3000) on $\mathbb{Q}^2$ (where NTPP is the “non-tangential-proper-part” relation of RCC-8 [88]).

```
ex:hotspot1 rdf:type noa:Hotspot .
ex:fire1 rdf:type noa:Fire .
ex:hotspot1 noa:correspondsTo ex:fire1 .
ex:fire1 noa:occuredIn _region1 .
_region1 NTPP “x ⩾ 0 ∧ x ⩽ 3000 ∧ y ⩾ 0 ∧ y ⩽ 3000”
```

(3) STT [102], [82]

In order to handle spatial and temporal data, Sheth and Perry [102], [82] present the idea of spatial, temporal and thematic (STT) processing of Semantic Web data and describe the Web infrastructure needed to support it.

They use the temporal RDF graphs in [32], [33] to model linear discrete absolute time. Given a set of discrete, linearly ordered time points T, a temporal triple is an RDF triple with a temporal label $t \in T$ that represents its valid time, i.e., $(s, p, o) : [t]$ or $(s, p, o) : [t_1, t_2]$ (which meas $(s, p, o) : [t] | t_1 \leqslant t \leqslant t_2)$. A temporal RDF graph is a set of temporal triples. For example, a soldier s1 is assigned to the 1st armored division (1stAD) from 3 April 1942 until 14 June

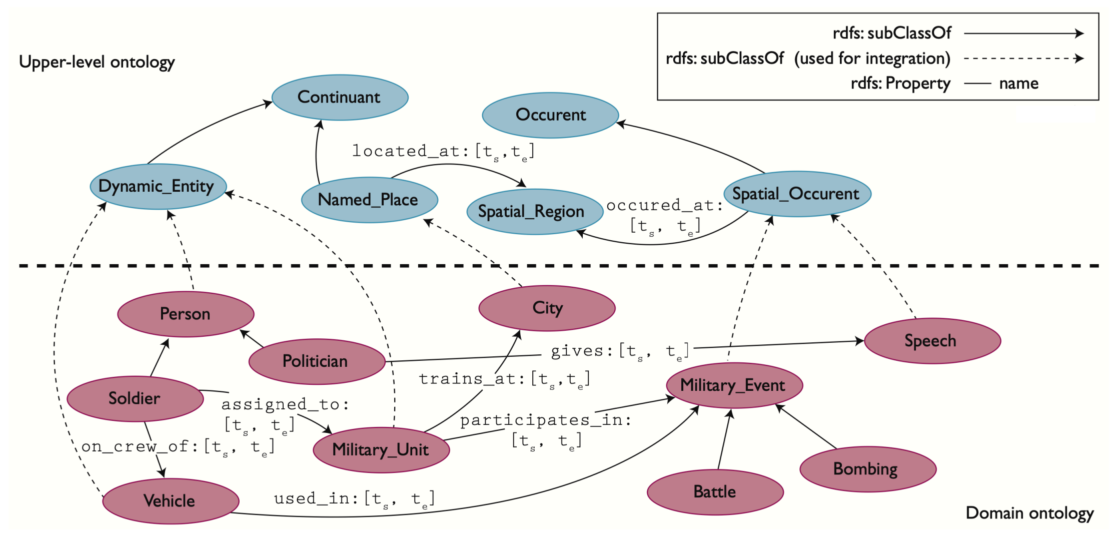


Figure 5 An STT example

1943 and then assigned to the 3rd armored division (3rdAD) from 15 June 1943 until 18 October 1943. This would yield the following triples: (s1, assigned_to, 1stAD) : [04:03:1942, 06:14:1943], (s1, assigned_to, 3rdAD) : [06:15:1943, 10:18:1943].

They further define a small upper-level ontology that defines the basic classes and relationships of the thematic and spatial domains. The upper-level ontology distinguishes between *continuants*, which persist over time and maintain their identity through change, and *occurrents*, which represent processes and events. *Spatial_Occurrents* and *Named_Places* are spatial entities directly linked with *Spatial_Regions* that record their geographic location by *occured_at* and *located_at*, and *Dynamic_Entities* represent those with dynamic spatial behavior. Temporal intervals on relationships denote when the relationship holds (valid time).

The following Figure 5 shows an SST model of space, time, and theme. An upper-level ontology defining basic classes and relationships is shown in blue, and a sample military domain ontology is shown in magenta for illustration.

A web infrastructure EventWeb is envisioned, with key components in its architecture coming from combining spatio-temporal data management research in geographic information systems (GIS) and database communities with current Semantic Web research and technologies (ontologies, representation languages, query languages, and so on).

(4) gst-Store [128], [129]

Wang et al. (2014) [128] implement a demo of spatiotemporal information integrated RDF data management system, called g^{st}-Store. In g^{st}-Store, some entities have spatio-temporal features, and some statements have valid time intervals and locations.

For representing spatio-temporal features, each statement as a five-tuple $\langle s, p, o, l, t\rangle$, where s, p, o, l, t represent for the features subject, predicate, object, *location* and *time interval* respectively. Note that l and t can be *null* if there is no spatial nor temporal information. Specifically, they use the longitude and latitude $\langle x, y\rangle$ to denote the happening location of a statement. The "time interval" feature is a bigram $\langle start, end\rangle$, i.e., it has a start time and an end time. Both the start time and the end time are dates with format "yyyy-mm-dd".

(5) YAGO2 [41]

Hoffart et al. (2013) [41] present a spatio-temporal model YAGO2, an extension of the YAGO knowledge base, in which entities, facts, and events are anchored in both time and space. YAGO2 is built automatically from Wikipedia, GeoNames [37], and WordNet [125].

YAGO2 represents the spatio-temporal knowledge by SPOTL tuples (SPO + Time + Location), an extension of the traditional SPO-triple RDF model to space and time.

- Regarding the temporal dimension, YAGO2 contains a data type `yagoDate` that denotes *time points*, typically with a resolution of days but sometimes with cruder resolution like years. Dates are denoted in the standard format `YYYY-MM-DD` (ISO 8601). If only the year is known, it writes dates in the form `YYYY-##-##` with # as a wildcard symbol. In YAGO2, facts can only hold at time points; time spans are represented by two relations that together form a time interval (e.g. *wasBornOnDate* and *diedOnDate*). Further, *entities* are assigned a time span to denote their existence in time. *Facts* are assigned a time point if they are instantaneous events, or a time span if they have an extended duration with known begin and end. In this case, YAGO2 assigns begin and/or end of time spans to all entities, to all facts, and to all events, if they have a known start point or a known end point. If no such time points can be inferred from the knowledge base, it does not attempt any assignment.
- Regarding the spatial dimension, YAGO2 introduce a new class `yagoGeoEntity`, which groups together all geo-entities, i. e. all entities with a permanent physical location on Earth. The `yagoGeoEntity` contains several subclasses, e.g., location, track, and land. Geographical coordinates, consisting of latitude and longitude, can describe the position of a geo-entity. YAGO2 introduces a special data type to store geographical coordinates, `yagoGeoCoordinates`. An instance of `yagoGeoCoordinates` is a pair of a latitude and a longitude value. Each instance of `yagoGeoEntity` is directly connected to its geographical coordinates by the `hasGeoCoordinates` relation. Further, YAGO2 assigns a location to both entities and facts wherever this is ontologically reasonable and wherever this can be deduced from the data. The location of facts and entities is given by a geo-entity. For example, the location of the Summer of Love is San Francisco, which is an instance of `yagoGeoEntity`.

Formally, in YAGO2, each *base-fact* has an identifier, which in turn can be used in the *S* or *O* role in another fact, a *meta-fact*. For example, suppose we know the base-fact #1: *GratefulDead performed TheClosingOfWinterland* about the rock band Grateful Dead. Adding knowledge about the place and time of this concert is expressed by two meta-facts #2: #1 *occursIn SanFrancisco* and #3: #1 *occursOnDate* 1978-12-31.

(6) stRDFS [138]

Zhu et al. (2020) [138] define a spatiotemporal data model stRDFS based on RDF. The model is a triple (s, p: $<t$, $l>$, o), where t is a temporal label and l is a spatial label. Also, several spatio-temporal classes (e.g., SpatiotemporalObject and SpatiotemporalGeo) are introduced. They also define 11 kinds of topological relations to describe the relations among spatio-temporal entities: Equal, Disjoint, Meet, Overlap, Cover, CoveredBy, Inside, Contain, Before, Now, and After.

On this basis, the spatiotemporal semantics and the spatiotemporal algebraic operations were investigated. They define five types of graph algebras (union, intersection, difference, Cartesian product, and filter), and the filter operations can filter the spatiotemporal graphs using a graph pattern. Besides this, they put forward a spatiotemporal RDF syntax specification to help users browse, query, and reason with spatiotemporal RDF graphs.

4.2 Spatio-Temporal RDF Storage and Querying

For implementing the efficient storage and querying of spatio-temporal RDF data, there has been significant community and research interest in storing and querying spatio-temporal data based on RDF techniques. Table 5 first summarizes and compares the spatio-temporal RDF storage and querying techniques.

(1) stSPARQL [51]

Table 5 Summarization and comparison of spatio-temporal RDF storage and querying

	Query language	**Main idea**	**Supported spatio-temporal RDF model**	**Supported relations**	**Index**	**Storage and Implementation**
stSPARQL [51]	SPARQL extension	extending SPARQL by introducing spatial and temporal variables	stRDF [51]	topological relations [22] Allen-13 interval relations [4]		
stSPARQL^i [50], [53]	SPARQL extension	incorporating some functions (defined in the OpenGIS Simple Features Access standard for SQL [72]) into SPARQL	stRDF^i [50], [53]	topological relations [22], [26] qualitative spatial relations RCC-8 [88]	B+ tree R-tree-over-GiST	Strabon (a fully-implemented, open-source, storage and query evaluation system)
SPARQL-ST [82], [83]	SPARQL extension	extending SPARQL by introducing spatial and temporal variables and filters	STT [102], [82]		special spatial and temporal indexing schemes	A framework on Oracle 10g [74]
DiStRDF [64], [65]	SPARQL	developing a Distributed spatio-temporal RDF engine, which is built on a distributed in-memory processing framework Spark	RDF triples with Point geometries and temporal interval		Hilbert Hash, Z-order Hash	DiStRDF (Distributed spatio-temporal RDF engine on Spark)
g^{st}-Store [128], [129]	SPARQL extension	introducing spatiotemporal assertions into SPARQL queries	gst-Store [128], [129]		ST-tree	A simple demo
kSPT [132]	Keyword query	adding a temporal constraint to the top-k relevant semantic place retrieval (kSP) query [103]	RDF triples with spatial coordinates or timestamps			

Koubarakis and Kyzirakos (2010) [51] propose an extension of SPARQL, called stSPARQL, to query spatial and temporal data expressed in stRDF [51] as introduced in Section 4.1.

stSPARQL has a new kind of variables called *Spatial variables*, which can be used in basic graph patterns to refer to spatial literals denoting semi-linear point sets. They can also be used in spatial filters, a new kind of filter expressions introduced by stSPARQL that is used to compare spatial terms using spatial predicates. Spatial terms include spatial constants (finite representations of semi-linear sets, e.g., "$0 <= x <= 10$ and $0 <= y <= 10$"), spatial variables and complex spatial terms (e.g., ?GEO INTER "x=10 and y=10" which denotes the intersection of the value of spatial variable ?GEO and the semi-linear set "x=10 and y=10"). In stSPARQL, only the topological relations [22] can be used as predicates in a spatial filter expression e.g., filter(?GEO1 inside ?GEO2).

Moreover, stSPARQL offers one more new kind of variables in addition to spatial ones: *temporal variables*. A temporal term in stRDF is a temporal variable or a temporal constant (i.e., an element of the set C_1 e.g., "$(t >= 0$ and $t <= 2)$ or $(t >= 5$ and $t <= 7)$"). stRDF allows any of the Allen-13 interval relations [4] to be used as the interval predicates e.g, filter(?T contains (t = 11)).

They further define the semantics of stSPARQL by following the algebraic approach pioneered for SPARQL in [81]. The details definitions and symbols can be found in [51].

(2) stSPARQL^i [50], [53]

For querying the spatio-temporal RDF model *stRDF^i* [50], [53] in Section 4.1, Koubarakis and Kyzirakos (2012) [50], [53] propose an extension of SPARQL[51], called *stSPARQL^i*.

They incorporate the functions (defined in the OpenGIS Simple Features Access standard for SQL [72]) into SPARQL, where a URI for each of the functions is defined, and then these URIs are used either in the select or the filter part of a SPARQL query. For example, in order to use the ST_Contains function (i.e., "ST_Contains(A:Geometry, B:Geometry): Boolean" which is defined in [72] for representing spatially contains between two geometric objects A and B) that is not part of the core SPARQL specification, the following function is defined in stSPARQL^i, which take as arguments two spatial literals g1 and g2, checks whether g1 spatially contains g2 and returns the appropriate result encoded as an xsd:boolean typed literals.

```
xsd:boolean srdf:Contains(srdf:geometry g1, srdf:geometry g2)
```

That is to say, the arguments of the functions can be variables, spatial literals encoded in WKT or GML or other extension functions. For example, to check whether the variable ?GEO is bound to a spatial literal that contains the point "POINT(669062 4238286); urn:epsg:ggrs87"^^srdf:geometry, the following expression can be used in the filter clause of a standard SPARQL query:

```
srdf:Contains(?GEO, "POINT(669062 4238286); urn:epsg:ggrs87"^^srdf:geometry)
```

Moreover, stSPARQL^i allows spatial functions to be used in the select part of a SPARQL query as well (as mentioned in SPARQL 1.1). For example, to obtain the buffer of a spatial literal that is bound to the variable ?GEO (the buffer function "ST_Buffer(A:Geometry, distance:Double): Geometry" is defined in [72], which returns a geometric object that represents all Points whose distance from the given geometric object is less than or equal to distance), the following select expression can be defined in stSPARQL^i:

```
select srdf:Buffer(?GEO) as ?GEOBUFFER
```

They also implement a system called Strabon 3.0, which is a fully-implemented, open-source, storage and query evaluation system for stSPARQL^i and the corresponding subset of GeoSPARQL [68], [7]. A detailed evaluation of the system using a workload based on linked data and a synthetic workload is done. Strabon is implemented by extending the widely-known RDF store Sesame [110], and it uses Sesame 2.6.3 and comprises three modules: the storage manager, the query engine and PostGIS (a spatially enabled DBMS).

- Storage manager: utilizes a bulk loader to store $stSPARQL^i$ triples [50], [53] using the "one table per predicate" scheme of Sesame and dictionary encoding. For each predicate table, two B+ tree two-column indices are created. For each dictionary table a B+ tree index on the *id* column is created. All spatial literals are also stored in a table with schema *geo_values(id int, value geometry, srid int)*. Each tuple in the *geo_values* table has an *id* that is the unique encoding of the spatial literal based on the mapping dictionary. The attribute *value* is a spatial column whose data type is the PostGIS type `geometry` and is used to store the geometry that is described by the spatial literal. The geometry is transformed to a uniform, user-defined CRS and the original CRS is stored in the attribute *srid*. Additionally, a B+ tree index on the *id* column and an R-tree-over-GiST spatial index on the *value* column are created.
- Query engine: consists of a parser, an optimizer, an evaluator and a transaction manager.

Note that, the version of the system Strabon [50], [53] as mentioned above has not implemented the temporal dimension of this data model and query language. This situation is remedied in [8], [9] by introducing all the details of the temporal dimension of stRDF and stSPARQL and implementing it in Strabon.

(3) SPARQL-ST [82], [83]

Perry (2008) [82], [83] proposes SPARQL-ST, an extension of SPARQL that allows querying the STT (spatial, temporal and thematic) RDF model [102], [82] as mentioned in Section 4.1.

In detail, SPARQL-ST adds two new types of variables, namely spatial and temporal ones, to the standard SPARQL variables. Temporal variables are identified using a '#' prefix, and spatial variables are identified using a '%' prefix. Furthermore, in SPARQL-ST two special filters are introduced: SPATIAL FILTER and TEMPORAL FILTER. These filters are used to filter the results with spatial and temporal constraints.

The following provides a SPARQL-ST query example in [82]: At what times does John Linder represent a district that borders a district represented by a member of a different political party, and who is the other representative? This query uses a SPATIAL FILTER to join two disjoint graph patterns, but it also selects the intersect interval that represents the times the spatial relation holds.

```
SELECT ?n, intersect(#t1, #t2, #t3, #t4, #t5, #t6, #t7, #t8, #t9, #t10, #t11, #t12)
WHERE {
?l foaf:name ”John Linder” #t1 .
?l usgov:party ?p1 #t2 .
?l usgov:hasRole ?r #t3 .
?r usgov:forOffice ?o #t4 .
?o usgov:represents ?q #t5 .
?q stt:locatedat %g #t6 .
?a foaf:name ?n #t7 .
?a usgov:party ?p2 #t8 .
?a usgov:hasRole ?b #t9 .
?b usgov:forOffice ?c #t10 .
?c usgov:represents ?d #t11 .
?d stt:locatedat %h #t12 .
FILTER (?p1 <> ?p2)
SPATIAL FILTER (touch(%g, %h)) }
```

They also implement SPARQL-ST using the extensibility framework of Oracle 10g [74]. The implementation builds on Oracle's existing support for storage and querying of RDF data and spatial data. It provides a single SQL table function that inputs a valid SPARQL-ST query and returns a table of the resulting variable mappings. And the special Spatial Indexing Scheme and Temporal Indexing Scheme are defined. The experiments are conducted using two RDF datasets called SynHist and GovTrack.

(4) DiStRDF [64], [65]

Nikitopoulos et al. (2019) [64], [65] address the problem of efficiently storing and querying spatio-temporal RDF data in parallel. They develop a DiStRDF system (Distributed spatio-temporal RDF engine), which is built on a well-known distributed in-memory processing framework Spark.

DiStRDF comprises of two main modules as shown in Figure 6:

- The Storage Layer provides distributed storage of large amount of historical spatio-temporal RDF data of moving objects, with careful data organization on disk and storage layout.
- The Processing Layer is a query engine that provides both logical and physical operators for spatio-temporal RDF data, thereby offering the opportunity for different execution plans to increase performance gains. A SPARQL query and a separate spatio-temporal constraint are provided by a user to the DiStRDF Processing Layer.

Note that, DiStRDF currently supports only Point geometries but is being extended to support also other types of geometries. Moreover, to encode the spatio-temporal information of a moving entity, they propose an encoding scheme for spatio-temporal RDF data and map its spatial location in a integer value, which is also mentioned in [120]. DiStRDF uses the Hilbert curve or the Z-order curve to produce an encoding (hash) value; these values are grouped in ranges to enable efficient pruning groups of data which do not satisfy query predicates. Finally, the experimental evaluation, on real data from both aviation and maritime domains, demonstrates the efficiency of the DiStRDF system, when using various spatio-temporal range constraints.

(5) g^{st}-Store [128], [129]

Wang et al. (2014&2017) [128], [129] introduce a variety of spatiotemporal assertions to express users' query requirements into traditional SPARQL queries based on the previous work *gStore* [136] and S-store [127] as mentioned in Section 3.2. The spatial range assertions restrict the area of an entity that locates in or an event (an event is denoted as a triple statement) that happens in. The spatial join assertions require that the distance between the locations of two entities/events

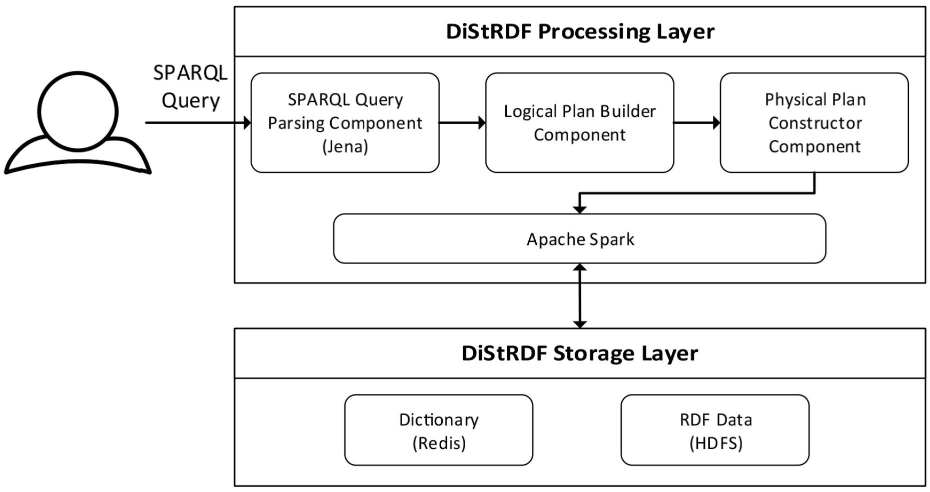

SPARQL Query
DiStRDF Processing Layer
SPARQL Query Parsing Component (Jena)
Logical Plan Builder Component
Physical Plan Constructor Component
Apache Spark
DiStRDF Storage Layer
Dictionary (Redis)
RDF Data (HDFS)

Figure 6 The structure of DiStRDF

is no more than a given distance threshold r. The temporal range assertions restrict the time interval of an entity or an event. The temporal join assertions express the relative position of the time intervals of two entities/events.

A SPARQL query with spatio-temporal assertions can be modeled as a query graph Q . It contains the following elements.

- Vertices in Q which denote the subjects or objects, and edges which indicate the triple patterns in a SPARQL query.
- The textual features of vertices or edges, i.e. the non-variable subjects, predicates and objects.
- A list of spatio-temporal assertions of variables.
- A list of spatio-temporal assertions of statements.

The goal of query processing is to find the subgraph matches in the RDF graph meanwhile satisfying the spatio-temporal assertions. The matches of a query is considered as the query results. g^{st}-Store organizes each match as a list of statements corresponding to the query patterns and visualizes the matches on the map.

In detail, g^{st}-Store provides two range shapes, i.e., a rectangle or a circle. It uses the range of the longitude and latitude to denote a rectangle "$([x_1, x_2], [y_1, y_2])$", and use the longitude and latitude of the center O as well as the radius r to denote a circle $(O = (x, y), r = R)$. Further, it uses expressions like "$[expression, r = D]$" to denote a spatial join assertion, which can restrict the maximum distance between a variable/statement and another variable/statement.

Similarly, the temporal assertions also have range and join semantics. In temporal range assertions, g^{st}-Store uses comparison symbols $(>, <, =, >= or <=)$ or a temporal range (*year-month-day, year-month-day*) to limit the boundary of the both ends of the time interval (where *year-month-day* denotes a time point). Specially, if a user cannot determine the exact date, the last bits can be replaced by "#" as a wildcard. For instance, "19##-##-##" means any day in the 20th century. It also uses expressions like "[*symbol, id, year-month-day*]" to denote a temporal join assertion, where *"symbol"* is a comparison symbol "$>, <, or <>$" to point the direction of the assertion, "$>$" ("$<$") means the assertion is after(before) a time point (or the end time of a time interval), and "¡¿" means this statement happens during a time interval that around the corresponding statements within a given length; "*id*" is the input serial number to indicate the corresponding statement query pattern. In order to point out the start time or the end time or the *id*th statement query pattern, it allows to add "*.s*" (start time) or "*.e*" (end time) following the id; "*year-month-day*" denotes the interval length from the constrained time point. Note that if the last characters of *id* are "*.s*" or "*.e*", it means that the start time or the end time of the corresponding statement query pattern. For example, "[<,1.e,10-##-##]" means that the time point is earlier than the first statement's end time for at least 10 years, and the "*year-month-day*" denotes the time length from the constrained time point.

Finally, the query processing is based on a tree-style index (ST-tree, which is built based on the "insert" and the "split" operations similar as the R-tree [28] and the VS-tree [136]) and a top-down search algorithm.

(6) *k*SPT [132]

Wu et al. (2019) [132] propose the top-k relevant semantic place with temporal constraint retrieval (kSPT) query, which adds a temporal constraint to the top-k relevant semantic place retrieval (kSP) query [103] as mentioned in Section 3.2.1. Both kSP and kSPT queries share the same motivation as RDF keyword queries; they are independent of the data domain and do not rely on structured languages such as SPARQL. The kSPT query uses two ways to incorporate temporal information. One way is considering the temporal differences between the keyword-matched vertices and the query timestamp. The other way is using a temporal range to filter keyword-matched vertices. The proposed techniques are evaluated on two large real RDF data sets, i.e., DBpedia [57] and YAGO [41].

4.3 Spatio-Temporal RDF Datasets and Tools

With the development of spatio-temporal RDF techniques as mentioned above, several spatio-temporal RDF data management tools and datasets are explored.

- YAGO2 [41]: an extension dataset of the YAGO knowledge base that includes spatial and temporal knowledge. YAGO includes more than 10M entities (like persons, organizations, cities) and contains more than 120M facts about these entities.
- EO [114]: a dataset resulting from the integration process of three different sources: land cover data of a specific French winery geographic area, its administrative units, and their land registers. Stored and published as an RDF triplestore, it is exposed through a SPARQL endpoint and exploited by a semantic query interface.
- AllegroGraph [5]: a Horizontally Distributed, Multi-model (Document and Graph), Entity-Event Knowledge Graph technology that enables businesses to extract sophisticated decision insights and predictive analytics from their highly complex, distributed data that can't be answered with conventional databases. With respect to the spatio-temporal RDF, AllegroGraph can *store geospatial and temporal data types as native data structures.* Combined with its indexing and range query mechanisms, AllegroGraph lets you perform geospatial and temporal reasoning efficiently. It provides a novel mechanism for efficient storage and retrieval of multi-dimensional data, and spatial data in particular. Also, the SPARQL magic properties for multi-dimensional geospatial are defined. It also covers temporal points, temporal intervals, and how to locate each of these on a one-dimensional timeline. At the same time it provides a number of Prolog functors for specifying temporal constraints in select queries.
- LandCover2RDF [24]: an API for computing the land cover of a geographical area and generating the RDF graph. It is a REST API (and a web user interface) that allows for computing the percentage of land cover classes of a geographic area according to a given map. The computed data is then represented as an RDF graph based on an ontology dedicated to this kind of data focusing on their temporal and spatial dimensions.
- Mapping tool [121]: a prototype for mapping spatiotemporal data to RDF using R2RML (a standard language that allows defining customized mappings from relational databases to RDF datasets). Also, a SPARQL endpoint is implemented using Strabon [50], [53].

- SPARTAN [95]: a framework for real-time semantic integration of big spatio-temporal data from streaming and archival sources, aiming at providing enriched trajectories that are exploited by higher level analysis tasks.

5 Discussions and Research Directions

Based on the introduction in the previous sections, it is shown that spatial and spatio-temporal RDF management techniques and related issues have been widely investigated in order to handle spatial and spatio-temporal information in the context of the geographic information systems (GIS), the Semantic Web and other real-world applications. However, the researches on spatial and spatio-temporal RDF management techniques are still in a developing stage and still the full potential of RDF has not been exhaustively explored. The following issues may be important in order for RDF technologies to be more widely adoptable in many application domains:

- *Ability of representation* of spatio-temporal RDF needs to be further investigated to satisfy the requirement of applications. Adding time and space to RDF is a challenging theme [15], [77]. With the introduction of various of spatio-temporal dimensions (e.g., user-defined time, transaction time, valid time, complex spatial objects, and combinations of these different grained dimensions) into RDF and RDFS, an even higher level of complexity of querying or related techniques arises. Therefore, regarding to a new spatio-temporal RDF model, the work needs to comprehensively discuss its core model components and the extension of RDF/RDFS vocabulary, and further provide the efficient query support.
- *Compatibility* with the standards (e.g., OGC, W3C) and the practice suggestions expect to be considered. As mentioned in [119], several suggested practices for publishing spatial data on the Web are proposed for making spatial data more effectively available. Accordingly, more discussions and practice guidelines for extending RDF to implement the efficient management of spatio-temporal data need to be investigated in depth.
- Efficiency of spatio-temporal *indexing* over extended RDF models should be improved and performed faster. As also mentioned in [77], [104], apart from simple spatial objects (e.g., points), this structure should be able to host diverse geometric shapes (lines, polygons, etc.) and facilitate related computations. Taking advantage of that index to natively evaluate more types of spatial queries would also be worthwhile. The recent researches [79] and [25] introduced several indexes.
- More *storage* frameworks and systems need to be investigated and implemented. As mentioned in Section 3.2, lots of spatial RDF storage and querying systems have not implemented the temporal dimension of their data model and query language. Currently, there are only several implementions for supporting both the temporal and spatial dimensions (e.g., Strabon [50], [53], [8], [9], SPARQL-ST framework [82], [83], and DiStRDF [64], [65] as mentioned in Section 4.2). With the rapid increase in spatio-temporal data in real-world applications, more systems need to be investigated and implemented for efficiently retrieving and managing spatio-temporal data.
- *Performance study and assessment* of spatio-temporal RDF querying and storage techniques need to be discussed in depth. As shown in Section 3.2.2, some work are developed for assessing spatially enabled RDF querying and storage techniques, but there is less work about the assessment of spatio-temporal RDF techniques. The comprehensive and in-depth assessment on the popular and well-known spatio-temporally enabled RDF stores is crucial for promoting and implementing the efficient management of spatio-temporal RDF data.
- Other important issues about spatio-temporal RDF data management, such as integration, construction, and reasoning, will be very interesting topics for future research. For example, reasoners should be further exploited for supporting the efficient reasoning of spatio-temporal RDF relations. More integration techniques and systems need to be developed for performing the fusion actions on linked spatio-temporal RDF entities, considering both spatio-temporal and non-spatiotemporal properties of them.

6 Conclusions

In this paper we provide a comprehensive overview of RDF for spatial and spatio-temporal data management. We summarize spatial and spatio-temporal RDF data management from several essential aspects, including presentation, querying, storage, performance assessment, datasets, and management tools. As shown in the overview, many proposals for extensions to RDF with different formalisms are proposed in order for RDF to represent spatial and spatio-temporal information. Further, for achieving efficient querying and storage on the spatial and spatio-temporal RDF data, kinds of variants of spatial and spatio-temporal RDF querying and storing techniques are subsequently developed. Also, the corresponding performance study and assessment about the querying and storage techniques are done. And some datasets and management tools are developed that has arisen from practical needs. Final, some discussions and several future research directions are identified. The overview in this paper may help readers understand and catch some key techniques about the issue. In our future work we will further track the technologies and cover more topics about RDF for spatial and spatio-temporal data management.

Acknowledgments. The authors really appreciate the hard work and the time that the reviewers for reviewing this paper. The work is supported by the Natural Science Foundation of Ningxia Province (NO. 2020AAC03212), Major scientific and technological innovation projects of Shandong Province (2019JZZY010128), and the National Natural Science Foundation of China (61672139).

References

[1] Analyti A, Pachoulakis I. A survey on models and query languages for temporally annotated RDF. International Journal of Advanced Computer Science & Applications, 2008, 1(3): 28-35.

[2] Ali W, Saleem M, Yao B, et al. Storage, Indexing, Query Processing, and Benchmarking in Centralized and Distributed RDF Engines: A Survey. arXiv preprint arXiv:2009.10331, 2020.

[3] Athanasiou S, Bezati L, Giannopoulos G, Patoumpas K. and Skoutas D. GeoKnow - Making the Web an Exploratory for Geospatial Knowledge. Market and Research Overview. 2012.

[4] Allen J F. Maintaining knowledge about temporal intervals. Communications of the ACM, 1983, 26(11): 832-843.

[5] AllegroGraph, https://allegrograph.com

[6] Bellini P, Nesi P. Performance assessment of RDF graph databases for smart city services. Journal of Visual Languages & Computing, 2018, 45: 24-38.

[7] Battle R, Kolas D. Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL. Semantic Web, 2012, 3(4):355–370.

[8] Bereta K, Dogani K, Garbis G, et al. An implementation of a temporal and spatial extension of RDF and SPARQL on top of MonetDB-Phase II. 2013.

[9] Bereta K, Smeros P, Koubarakis M. Representation and querying of valid time of triples in linked geospatial data. ESWC. 2013: 259–274.

[10] Bereta K, Xiao G, Koubarakis M, et al. Ontop-spatial: Geospatial data integration using GeoSPARQL-to-SQL translation. Proceedings of the 15th International Semantic Web Conference, Posters & Demonstrations Track (ISWC). 2016.

[11] Bereta K, Xiao G, Koubarakis M. Ontop-spatial: Ontop of geospatial databases. Journal of Web Semantics, 2019, 58: 100514.

[12] Berners-Lee T, Hendler J, Lassila O. The semantic web. Scientific american, 2001, 284(5): 34-43.

[13] Bizer C, Heath T, Berners-Lee T. Linked Data-The Story So Far. International Journal on Semantic Web and Information Systems, 2009, 5(3): 1-22.

[14] Brodt A, Nicklas D, Mitschang B. Deep integration of spatial query processing into native RDF triple stores. Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2010: 33-42.

[15] Claramunt C. Ontologies for geospatial information: Progress and challenges ahead. Journal of Spatial Information Science, 2020, 2020(20): 35-41.

[16] Cai Z, Kalamatianos G, Fakas G J, et al. Diversified spatial keyword search on RDF data. The VLDB Journal, 2020: 1-19.

[17] Candan K S, Liu H, Suvarna R. Resource description framework: metadata and its applications. ACM SIGKDD Explorations Newsletter, 2001, 3(1): 6-19.

[18] Chawla T, Singh G, Pilli E S, et al. Storage, partitioning, indexing and retrieval in Big RDF frameworks: A survey. Computer Science Review, 2020, 38: 100309.

[19] Chbeir R, Amghar Y, Flory A. Novel indexing method of relations between salient objects. Effective databases for text & document management. IGI Global, 2003: 174-182.

[20] Christodoulou G. CHOROS: A Reasoning and Query Engine for Qualitative Spatial Information. Dissertion Thesis, Technical University of Crete, Greece, 2011.

[21] Clementini E, Di Felice P. A comparison of methods for representing topological relationships. Information sciences-applications, 1995, 3(3): 149-178.

[22] Cui Z, Cohn A G, Randell D A. Qualitative and Topological Relationships in Spatial Databases. In: Advances in Spatial Databases. 1993.

[23] Date C J, Darwen H, Lorentzos N. Temporal data & the relational model. Elsevier, 2002.

[24] Dorne J, Aussenac-Gilles N, Comparot C, et al. LandCover2RDF: an API for computing the land cover of a geographical area and generating the RDF graph. European Semantic Web Conference. Springer, Cham, 2020: 73-78.

[25] Eom S, Jin X, Lee K H. Efficient generation of spatiotemporal relationships from spatial data streams and static data. Information Processing & Management, 2020, 57(3): 102205.

[26] Egenhofer M, Herring J. Categorizing binary topological relationships between regions, lines and points in geographic database. Technical report, Department of Surveying Engineering, University of Maine, Urono, ME, 1991.

[27] Finkel R A, Bentley J L. Quad trees a data structure for retrieval on composite keys. Acta informatica, 1974, 4(1): 1-9.

[28] Guttman A. R-trees: A dynamic index structure for spatial searching. Proceedings of the 1984 ACM SIGMOD international conference on Management of data. 1984: 47-57.

[29] Garbis G, Kyzirakos K, Koubarakis M. Geographica: A benchmark for geospatial RDF stores (long version). In Proceedings of the International Semantic Web Conference, Sydney, NSW, Australia. 2013: 343–359.

[30] Giannopoulos G, Vitsas N, Karagiannakis N, et al. FAGI-gis: A tool for fusing geospatial RDF data. European Semantic Web Conference. Springer, Cham, 2015: 51-57.

[31] Gür N, Pedersen T B, Zimányi E, et al. A foundation for spatial data warehouses on the semantic web. Semantic Web, 2018, 9(5): 557-587.

[32] Gutierrez C, Hurtado C, Vaisman A. Temporal RDF. Proc. European Conf. Semantic Web, Springer, 2005: 93–107.

[33] Gutierrez C, Hurtado C, Vaisman A. Introducing Time into RDF. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(2): 207–218.

[34] GML, Geography Markup Language. https://www.ogc.org/standards/gml

[35] GeoRDF, https://www.w3.org/wiki/GeoRDF

[36] GraphDB (former OWLIM), https://www.ontotext.com/products/graphdb/

[37] GeoNames, http://www.geonames.org

[38] geometry2rdf Utility, https://www.oeg-upm.net/index.php/en/technologies/151-geometry2rdf/index.html

[39] Hamdi F, Abadie N, Bucher B, Feliachi A. Geomrdf: a geodata converter with a fine-grained structured representation of geometry in the web. The 1st International Workshop on Geospatial Linked Data (GeoLD), 2014: 1-12.

[40] Huang W, Raza S A, Mirzov O, et al. Assessment and benchmarking of spatially enabled RDF stores for the next generation of spatial data infrastructure. ISPRS International Journal of Geo-Information, 2019, 8(7): 310.

[41] Hoffart J, Suchanek F M, Berberich K, et al. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. Artificial Intelligence, 2013, 194: 28-61.

[42] Ioannidis T, Garbis G, Kyzirakos K, et al. Evaluating Geospatial RDF stores Using the Benchmark Geographica 2. arXiv preprint arXiv:1906.01933, 2019.

[43] ISO 19125-1: 2004 Geographic information-Simple feature access.

[44] ISO 19156, Geographic information-Observations and measurements.

[45] ISO 19109:2005 Geographic information-Rules for application schema.

[46] ISO 19107: 2003 Geographic Information-Spatial Schema.

[47] Jin X, Shin S, Jo E, et al. Collective keyword query on a spatial knowledge base. IEEE Transactions on Knowledge and Data Engineering, 2018, 31(11): 2051-2062.

[48] Jena, http://jena.apache.org/.

[49] Kolas D. A benchmark for spatial semantic web systems. International Workshop on Scalable Semantic Web Knowledge Base Systems. 2008.

[50] Koubarakis M, Kyzirakos K, Nikolaou B, et al. A data model and query language for an extension of RDF with time and space. Technical Report, 2012.

[51] Koubarakis M, Kyzirakos K. Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. Extended Semantic Web Conference. Springer, Berlin, Heidelberg, 2010: 425-439.

[52] Kuper G, Ramaswamy S, Shim K, Su J. A Constraint-based Spatial Extension to SQL. Proceedings of the 6th International Symposium on Advances in Geographic Information Systems, 1998.

[53] Kyzirakos K, Karpathiotakis M, Koubarakis M. Strabon: a semantic geospatial DBMS. International Semantic Web Conference. Springer, Berlin, Heidelberg, 2012: 295-311.

[54] Kyzirakos K, Savva D, Vlachopoulos I, et al. GeoTriples: Transforming geospatial data into RDF graphs using R2RML and RML mappings. Journal of Web Semantics, 2018, 52: 16-32.

[55] Leeka J, Bedathur S, Bera D, et al. STREAK: An Efficient Engine for Processing Top-k SPARQL Queries with Spatial Filters. arXiv:1710.07411v1, 2017.

[56] Leeka J, Bedathur S, Bera D, et al. Quark-X: An efficient top-k processing framework for RDF quad stores. Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. 2016: 831-840.

[57] Lehmann J, Isele R, Jakob M, et al. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic web, 2015, 6(2): 167-195.

[58] Liagouris J, Mamoulis N, Bouros P, et al. An effective encoding scheme for spatial RDF data. Proceedings of the VLDB Endowment, 2014, 7(12): 1271-1282.

[59] LinkedGeoData, LGD, http://linkedgeodata.org

[60] Nandal R. spatio-temporal Database and Its Models: A Review. IOSR Journal of Computer Engineering, 2013, 11(2): 91-100.

[61] Neumann T, Weikum G. Rdf-3x: a risc-style engine for rdf. PVLDB, 2008, 1(1): 647–659.

[62] Nikolaou C, Koubarakis M. Querying Linked Geospatial Data with Incomplete Information. In 5th International Terra Cognita Workshop - Foundations, Technologies and Applications of the Geospatial Web and in conjunction with the 11th International Semantic Web Conference, 2012.

[63] Nikolaou C, Koubarakis M. Incomplete information in RDF. International Conference on Web Reasoning and Rule Systems, 2013: 138-152.

[64] Nikitopoulos P, Vlachou A, Doulkeridis C, et al. DiStRDF: Distributed spatio-temporal RDF Queries on Spark. EDBT/ICDT Workshops. 2018: 125-132.

[65] Nikitopoulos P, Vlachou A, Doulkeridis C, et al. Parallel and scalable processing of spatio-temporal RDF queries using Spark. GeoInformatica, 2019: 1-31.

[66] Özsu M T. A survey of RDF data management systems. Frontiers of Computer Science, 2016, 10(3): 418-432.

[67] OGC, http://www.opengeospatial.org/.

[68] OGC GeoSPARQL - A Geographic Query Language for RDF Data. OGC 11-052r4, 2012.

[69] OWL 2 Web Ontology Language Document Overview (Second Edition), W3C Recommendation 11 December 2012. https://www.w3.org/TR/owl2-overview/

[70] OGC 07-036, Geography Markup Language (GML) Encoding Standard, Version 3.2.1.

[71] OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture (05-126, 06-103r3, 06-103r4), current version 1.2.1.

[72] OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option, 2010.

[73] OpenStreetMap dataset, http://www.openstreetmap.org/

[74] Oracle. 2005. Oracle spatial resource description framework (RDF) 10g release 2.

[75] Papadias D, Theodoridis Y. Spatial relations, minimum bounding rectangles, and spatial data structures. International Journal on Geographic Information Systems, 1997, 11(2): 111-138.

[76] Paton N W, Williams M H, Dietrich K, et al. VESPA: A benchmark for vector spatial databases. British National Conference on Databases. 2000: 81-101.

[77] Patroumpas K, Giannopoulos G, Athanasiou S. Towards GeoSpatial semantic data management: strengths, weaknesses, and challenges ahead. Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2014: 301-310.

[78] Patroumpas K, Alexakis M, Giannopoulos G, et al. TripleGeo: an ETL Tool for Transforming Geospatial Data into RDF Triples. EDBT/ICDT Workshops, 2014: 275-278.

[79] Pandey V, van Renen A, Kipf A, et al. The case for learned spatial indexes. 2nd International Workshop on Applied AI for Database Systems and Applications (AIDB'20), 2020: 1-9.

[80] Pelekis N, Theodoulidis B, Kopanakis I, et al. Literature review of spatio-temporal database models. Knowledge Engineering Review, 2004, 19(3): 235-274.

[81] Pérez J, Arenas M, Gutierrez C. Semantics and complexity of SPARQL. ISWC 2006. 2006: 30–43

[82] Perry M. A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data. PhD thesis, Wright State University, 2008.

[83] Perry M , Jain P , Sheth A . SPARQL-ST: Extending SPARQL to Support Spatiotemporal Queries. Semantic Web & beyond, 2011.

[84] Perry M, Estrada A, Das S, et al. Developing GeoSPARQL Applications with Oracle Spatial and Graph. ISWC. 2015: 57-61.

[85] Parliament, https://github.com/SemWebCentral/parliament

[86] PostGIS, https://postgis.net

[87] Quoca H N M, Serranob M, Mauc H N, et al. A Performance Study of RDF Stores for Linked Sensor Data. 2019.

[88] Randell D, Cui Z, Cohn A. A spatial logic based on regions and connection. Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning (KR-92), Cambridge, MA, 1992: 165-176.

[89] Rathee S, Yadav A. Survey on spatio-temporal Database and Data Models with relevant Features. International Journal of Scientific and Research Publications, 2013, 3(1): 1-5.

[90] Raza A. Comparison of geospatial support in RDF stores: Evaluation for ICOS Carbon Portal metadata. Master Thesis in Geographical Information Science, 2019.

[91] Revesz, P Z. Introduction to Constraint Databases. Springer, Heidelberg, 2002.

[92] Renz J, Nebel B. Qualitative spatial reasoning using constraint calculi. Handbook of spatial logics. Springer, Dordrecht, 2007: 161-215.

[93] Ronzhin S, Folmer E, Lemmens R, et al. Next generation of spatial data infrastructure: lessons from linked data implementations across Europe. International journal of spatial data infrastructures research, 2019, 14: 83-107.

[94] RDF 1.1 primer, W3C Recommendation. https://www.w3.org/TR/rdf11-mt/.

[95] Santipantakis G M, Apostolos G, Kostas P, et al. SPARTAN: Semantic integration of big spatio-temporal data from streaming and archival sources. Future Generation Computer Systems, 2020, 110: 540–555.

[96] Salas J, Harth A. Finding spatial equivalences accross multiple RDF datasets. Proceedings of the Terra Cognita Workshop on Foundations, Technologies and Applications of the Geospatial Web. Bonn, Germany: CEUR, 2011: 114-126.

[97] Salas J, Harth A, et al. Neo-Geo Vocabulary: Defining a shared RDF representation for GeoData. Public draft, May 2011.

[98] Saveta T, Fundulaki I, Flouris G, et al. SPgen: A Benchmark Generator for Spatial Link Discovery Tools. International Semantic Web Conference. Springer, Cham, 2018: 408-423.

[99] Schneider M. Spatial and spatio-temporal Data Models and Languages. in L. Liu, M. T. Özsu, (Eds.). Encyclopedia of Database Systems, Springer US, 2009: 2681-2685.

[100] Sejdiu G, Ermilov I, Lehmann J, et al. DistLODStats: Distributed computation of RDF dataset statistics. International Semantic Web Conference. 2018: 206-222.

[101] Sherif M A M. Automating Geospatial RDF Dataset Integration and Enrichment. Universität Leipzig, 2016: 1-165.

[102] Sheth A, Perry M. Traveling the semantic web through space, time, and theme. IEEE Internet Computing, 2008, 12(2): 81-86.

[103] Shi J, Wu D, Mamoulis N. Top-k relevant semantic place retrieval on spatial RDF data. Proceedings of the 2016 International Conference on Management of Data. 2016: 1977-1990.

[104] Simon G. An Introduction to Geo Indexes and their performance characteristics. 2018.

[105] Smeros P, Koubarakis M. Discovering Spatial and Temporal Links among RDF Data. In WWW Workshop: Linked Data on the Web (LDOW), 2016.

[106] Snodgrass R, Ahn I. A Taxonomy of Time in Databases. In Proceedings of ACM SIGMOD International Conference on Management of Data, 1985: 236-246.

[107] Stadler C, Martin M, Auer S. Exploring the web of spatial data with facete. Proceedings of the 23rd International Conference on World Wide Web. 2014: 175-178.

[108] Stadler C, Lehmann J, Höffner K, et al. Linkedgeodata: A core for a web of spatial open data. Semantic Web, 2012, 3(4): 333-354.

[109] SPARQL 1.1 Query Language W3C Recommendation, https://www.w3.org/TR/sparql11-query/. 21 March 2013.

[110] Sesame (now is RDF4J project), https://rdf4j.org

[111] Stardog, https://www.stardog.com

[112] Shp2GeoSPARQL, https://github.com/jasaavedra/shp2geosparql

[113] Taylor K, Parsons E. Where is everywhere: bringing location to the web. IEEE Internet Computing, 2015, 19(2): 83-87.

[114] Tran B H, Aussenac-Gilles N, Comparot C, et al. Semantic Integration of Raster Data for Earth Observation: An RDF Dataset of Territorial Unit Versions with their Land Cover. ISPRS International Journal of Geo-Information, 2020, 9(9): 503, 1-20.

[115] Theocharidis K, Liagouris J, Mamoulis N, et al. SRX: efficient management of spatial RDF data. The VLDB Journal, 2019, 28(5): 703-733.

[116] Ulutaş Karakol D, Kara G, Yılmaz C, et al. Semantic Linking Spatial Rdf Data to the Web Data Sources. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, 2018.

[117] uSeekM, https://www.openhub.net/p/useekm

[118] van den Brink L, Janssen P, Quak W, et al. Linking spatial data: automated conversion of geo-information models and GML data to RDF. International Journal of Spatial Data Infrastructures Research, 2014, 9: 59-85.

[119] van den Brink L, Barnaghi P, Tandy J, et al. Best practices for publishing, retrieving, and using spatial data on the web. Semantic Web, 2019, 10(1): 95-114.

[120] Vlachou A, Doulkeridis C, Glenis A, et al. Efficient spatio-temporal RDF query processing in large dynamic knowledge bases. Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. 2019: 439-447.

[121] Vaisman A, Chentout K. Mapping Spatiotemporal Data to RDF: A SPARQL Endpoint for Brussels. ISPRS International Journal of Geo-Information, 2019, 8(8): 353.

[122] Virtuoso Universal Server, https://virtuoso.openlinksw.com

[123] W3C GEO, http://www.w3.org/2003/01/geo/, W3C Semantic Web Interest Group, 2003.

[124] W3C Geospatial Vocabulary, W3C Incubator Group Report 23 October 2007, https://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/

[125] Fellbaum C. WordNet: An Electronic Lexical Database. MIT Press, 1998.

[126] Wang C J, Ku W S, Chen H. Geo-store: a spatially-augmented sparql query evaluation system. Proceedings of the 20th International Conference on Advances in Geographic Information Systems. 2012: 562-565.

[127] Wang D, Zou L, Feng Y, et al. S-store: An Engine for Large RDF Graph Integrating Spatial Information. In DASFAA, 2013: 31-47.

[128] Wang D, Zou L, Zhao D. gst -Store: An Engine for Large RDF Graph Integrating Spatiotemporal Information. Proceeding of the 17th International Conference on Extending Database Technology (EDBT 2014), 2014: 652-655.

[129] Wang D, Zou L, Zhao D. gst-store: querying large spatiotemporal RDF graphs. Data and Information Management, 2017, 1(2): 84-103.

[130] Wiemann S, Bernard L. Spatial data fusion in spatial data infrastructures using linked data. International Journal of Geographical Information Science, 2016, 30(4): 613-636.

[131] Wu D, Hou C, Xiao E, et al. Semantic Region Retrieval from Spatial RDF Data. International Conference on Database Systems for Advanced Applications. Springer, Cham, 2020: 415-431.

[132] Wu D, Zhou H, Shi J, N Mamoulis. Top-k relevant semantic place retrieval on spatiotemporal RDF data. VLDB, 2020, 29(4): 893–917.

[133] Xiao Z, Huang L, Zhai X. Spatial information semantic query based on sparql. In Proceedings of SPIE, volume 7492, October 2009.

[134] Zhai X, Huang L, Xiao Z. Geo-spatial query based on extended sparql. In 2010 18th International Conference on Geoinformatics. 2010: 1–4.

[135] Zhao T, Zhang C, Anselin L, et al. A parallel approach for improving Geo-SPARQL query performance. International Journal of Digital Earth, 2015, 8(5): 383-402.

[136] Zou L, Mo J, Chen L, et al. gStore: answering SPARQL queries via subgraph matching. Proceedings of the VLDB Endowment, 2011, 4(8): 482-493.

[137] Zhang C, Beetz J, de Vries B. BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. Semantic Web, 2018, 9(6): 829-855.

[138] Zhu L, Li N, Bai L. Algebraic Operations on Spatiotemporal Data Based on RDF. ISPRS International Journal of Geo-Information, 2020, 9(2): 80.