

Data Models and Query Languages for Linked Geospatial Data

Manolis Koubarakis, Manos Karpathiotakis, Kostis Kyzirakos,
Charalampos Nikolaou, and Michael Sioutis

Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens
Athens, Greece

{koubarak,mk,kkyzir,charnik,sioutis}@di.uoa.gr

Abstract. The recent availability of geospatial information as linked open data has generated new interest in geospatial query processing and reasoning, a topic with a long tradition of research in the areas of databases and artificial intelligence. In this paper we survey recent advances in this important research topic, concentrating on issues of data modeling and querying.

1 Introduction

Linked data is a new research area which studies how one can make RDF data available on the Web, and interconnect it with other data with the aim of increasing its value for users [11]. The resulting “Web of data” has recently started being populated with geospatial data. Ordnance Survey is the first national mapping agency that has made various kinds of geospatial data from Great Britain available as open linked data¹. Another representative example of such efforts is LinkedGeoData² where OpenStreetMap data is made available as RDF and queried using the declarative query language SPARQL [8]. Finally, a similar effort is GeoLinked Data³ where geospatial data from Spain is made public using RDF [50]. With the recent emphasis on open government data, some of it encoded already in RDF⁴, these efforts demonstrate that the development of useful Web applications utilizing geospatial data might be just a few SPARQL queries away.

This recent, pragmatic emphasis on linked geospatial data continues the vision of the *Semantic Geospatial Web*, first articulated by Max Egenhofer in [19]. [19] invited GIS researchers to pay special attention to geospatial information on the Web, and contribute to the Semantic Web effort by developing geospatial ontologies, query languages and query processing techniques for geospatial information on the Web, etc.

¹ <http://www.ordnancesurvey.co.uk/oswebsite/products/os-opendata.html>

² <http://linkedgeodata.org/>

³ <http://geo.linkeddata.es/>

⁴ <http://data.gov.uk/linked-data/>

There are many research topics and relevant questions that deserve the attention of researchers in the area of linked geospatial data. For example:

1. *Data models and query languages.* How do we model geospatial information on the Web? What are useful geospatial ontologies? What extensions of relevant Semantic Web frameworks such as RDF, description logics, OWL, rules, etc. are appropriate? How do we express queries that target geospatial Web data? What extensions of SPARQL or other query languages are needed?
2. *System architecture and implementation.* What kinds of storage, indexing and query processing techniques are appropriate for linked geospatial data? How can we develop scalable systems for handling such data? What are appropriate benchmarks for comparing the performance of existing systems?
3. *User interfaces.* How can we develop user interfaces for linked geospatial data (e.g., ones based on natural language processing techniques or graphical ones)? What are appropriate high-level APIs that ease the rapid development of such user interfaces? Can we build on already deployed geospatial Web platforms such as Google Maps, Bing Maps or OpenStreetMap?
4. *Datasets.* What are interesting geospatial data sets already existing in other formats that can be transformed into RDF and made available as linked data? Can we develop useful tools that make this transformation as easy as possible? How do we extract geospatial information already available on the Web in textual form and encode it in linked geospatial data? How do we extend large Web ontologies, such as YAGO⁵ or the ontology of the KnowItAll project⁶, with geospatial knowledge available on the Web?
5. *Applications.* What are interesting applications of linked geospatial data in target sectors such as environment, leisure, transportation, earth observation, public security? For example, what applications are possible if we combine linked geospatial data with the huge amounts of user-generated geospatial content that is made available on line these days through, e.g., social networks?

In this paper we concentrate mainly on the first of the above topics and compare data models and query languages that have been proposed recently for the representation and querying of linked geospatial data. We also survey all the relevant implemented systems, but only discuss data modeling and query language issues for them. Finally, we give some examples of existing datasets, so that the reader can appreciate the variety of linked geospatial data sources that currently exist publicly and the applications that can be built on top of them.

The literature that we will survey concentrates mainly on data models based on XML and RDF since these are the most popular formats for data on the Web today. We also cover relevant existing standards in this area from the Open Geospatial Consortium (OGC), the International Organization for Standardization (ISO) and the World Wide Web Consortium (W3C). A lot of existing geospatial and Web data today uses these standards and we expect their use to

⁵ <http://www.mpi-inf.mpg.de/yago-naga/yago/>

⁶ <http://www.cs.washington.edu/research/knowitall/>

increase in the near future. We also include a short survey of relevant research in the area of relational databases. Relational DBMSs are now an established technology and geospatial extensions of well-known DBMSs have been available for some time. Therefore, geospatial relational DBMSs are a core technology that can be used to build systems for managing linked geospatial data. Thus, researchers should know well what previous research in this area has achieved, and what functionalities existing spatial DBMSs can offer, before they proceed to develop their own techniques and systems for linked geospatial data.

The organization of this paper is as follows. The next section discusses existing linked geospatial datasets and their applications. Section 3 gives a survey of existing OGC and ISO standards for geospatial data. Section 4 surveys the state of the art in geospatial extensions of relational DBMSs. Section 5 discusses data models based on XML. Then, Section 6 presents the core data models and query languages to be discussed in this paper; they are all based on RDF. Section 7 looks at existing RDF stores and examines which of the data models discussed in Section 6 are used in these systems. Finally, Section 8 concludes the paper.

2 Linked Geospatial Data

In this section we review some of the recent efforts to make geospatial information available on the Web as linked open data.

In the context of the open data effort of the UK Government⁷, Ordnance Survey⁸ (the national mapping agency for Great Britain) is committed to making publicly available various UK geospatial datasets. Some of these datasets have been made available as linked data (the 1:50,000 Scale Gazetteer, Code-Point Open, and the Administrative geography gazetteer for Great Britain) and can be queried through a SPARQL Endpoint. As an example, the Administrative geography of Great Britain describes the hierarchy of administrative units and the topological relations among them [22]. The corresponding ontology⁹ includes classes that represent the administrative units of Great Britain, and properties that describe qualitative topological relations. Two ontologies, the Geometry Ontology and the Spatial Relations Ontology, are used to provide geospatial vocabulary. These ontologies describe abstract geometries and topological (equivalent to RCC-8) relations respectively. Boundary-Line, a polygon dataset of areas defined by electoral and administrative boundaries, has been used to generate the topological relations among the covered areas based on the provided names and boundary information. The resulting dataset has been then combined with addresses, roads, land use, height and other datasets available in RDF.

⁷ <http://www.direct.gov.uk/en/index.htm>

⁸ <http://www.ordnancesurvey.co.uk/oswebsite/>

⁹ <http://www.ordnancesurvey.co.uk/ontology/AdministrativeGeography/v2.0/AdministrativeGeography.rdf>

There has been a number of interesting mashups that have been produced using the above linked data sets. See for example the Research Funding Explorer by Talis¹⁰.

GeoNames¹¹ is a gazetteer that collects both spatial and thematic information for various placenames around the world. GeoNames data is available through various Web services and is also published as linked data¹². The placenames in GeoNames are interlinked with each other defining regions that are inside other placenames, neighboring countries or placenames that have certain distance with the underlined placename. GeoNames data is linked to the DBpedia data and other linked data sources. Beyond names of places in various languages, data stored include latitude, longitude, elevation, population, administrative subdivision and postal codes. All coordinates use the World Geodetic System 1984 (WGS84) to be introduced in Section 3.1 below.

In [8], the LinkedGeoData effort is described where OpenStreetMap (OSM) data is transformed into RDF and made available on the Web. OSM data is represented by adhering to a relatively simple data model. It comprises three basic types, *nodes*, *ways* and *relations*, each of which defines instances that are uniquely identified by a numeric id. Nodes represent points on Earth and have longitude and latitude values in the WGS84 coordinate reference system. They can be used to model, e.g., the location of a tourist attraction, a shop, etc. Ways are ordered sequences of nodes that form a polyline or a polygon. They can be used to represent, e.g., a road. Finally, relations are groupings of multiple nodes and/or ways. Each individual element in OSM (node, way, relation) can have an arbitrary number of key-value pairs, called *tags*, associated with them. The ontology of the LinkedGeoData dataset has been derived mainly from OSM tags, counting up to 500 classes, 50 object properties and ca. 15,000 datatype properties. For example, the Acropolis in OSM is represented by an instance of class **node** and has data properties **name** and **historic** with values **Acropolis** and **archaeological site** respectively. The point coordinates for Acropolis are defined by the tuple (37.972, 23.726). Tags are not restricted to a single vocabulary, i.e., a user is free to populate the ontology with tags of his own preference. Furthermore, an effort has been made to match DBpedia with LinkedGeoData resources, taking into account the common classes between the two ontologies, using `owl:sameAs` links. Data is published with the aid of the Virtuoso Universal Server¹³. Finally, a facet-based browser¹⁴ and editor for linked geographical data has been implemented in order to showcase the benefits of revealing the structured information in OSM. The authors of [8] conclude that spatial data can be retrieved and interlinked on an unprecedented level of granularity and they plan to extend the mapping approach that they currently use for interlinking LinkedGeoData with other data sources.

¹⁰ <http://bis.clients.talis.com/>

¹¹ <http://www.geonames.org/>

¹² <http://www.geonames.org/ontology/documentation.html>

¹³ A SPARQL endpoint is available at <http://linkedgeodata.org/sparql>.

¹⁴ <http://browser.linkedgeodata.org/>

[50] and [77] describe the process that was followed for making public several heterogeneous Spanish datasets that are related to administrative, hydrographic and statistical domains. In contrast to [8] that just manages every resource as a point (represented by a coordinate of latitude and longitude), Alexander de León et al. deal with more complex geometries as well, such as line strings. This effort (called GeoLinked Data) is an open initiative whose aim is to enrich the Web of data with Spanish geospatial data available from the National Geographic Institute and the National Statistic Institute of Spain, in order to study national issues involving geography, but also statistical variables such as unemployment, population, dwelling, industry, and building trade. After identifying the data sources, an ontology was constructed according to the NeOn methodology [73], by reusing existing ontologies and vocabularies. The statistical Core Vocabulary [31] was chosen for describing complex statistics and the FAO Geopolitical Ontology¹⁵, the hydrOntology [76], the GML Ontology¹⁶, and the WGS84 vocabulary were chosen as geospatial vocabulary. To model temporal information, the Time Ontology¹⁷ was chosen. The authors developed a software library, called GEOMETRYtoRDF, to create RDF triples from geometrical information in GML and WKT formats (to be presented in Section 3). Alignment of datasets is carried out by identifying `owl:sameAs` relationships between administrative units and statistical information, similarly to [8]. Data is published and visualised with the aid of Virtuoso Universal Server and Pubby respectively. A faceted browser that works on top of those two systems has also been implemented. Future work will focus on identifying and interlinking with other datasets available as linked open data, mainly DBpedia and GeoNames. In addition, coverage of more complex geometrical information like polygons is also planned.

Yago2 [32] is a large ontology, a new version of the original Yago ontology [74], where entities, facts, and events are augmented with relevant temporal and spatial information. Yago2 contains approximately 80 million facts for 10 million entities that are automatically harvested from Wikipedia, GeoNames, and WordNet. In [32] the authors present the methodology of harvesting the data, augmenting them with spatial and temporal information and representing the produced knowledge base in SPOTL. SPOTL is a data model based on 5-tuples that are defined as standard RDF triples augmented by time and location. In Yago2, entities are assigned a time span to denote their existence in time, while facts may be assigned a time point or a time span. Spatial information is restricted to latitude/longitude points.

The NUTS classification (Nomenclature of territorial units for statistics)¹⁸ and the GADM (Global Administrative Areas)¹⁹ datasets have also been published recently as linked data. The NeoGeo vocabulary has been used to model the RDF representation of these datasets. These datasets have also been inter-

¹⁵ <http://www.fao.org/countryprofiles/geoinfo.asp?lang=en>

¹⁶ <http://loki.cae.drexel.edu/~wbs/ontology/2004/09/ogc-gml.owl>

¹⁷ <http://www.w3.org/TR/owl-time/>

¹⁸ <http://nuts.geovocab.org/>

¹⁹ <http://gadm.geovocab.org/>

linked with other linked geospatial datasets such as DBpedia, GeoNames and the geopolitical information dataset of FAO.

TELEIOS²⁰ is a recent European project that addresses the need for scalable access to petabytes of Earth Observation data and the discovery of knowledge hidden in them that can be used in applications. To achieve this, TELEIOS builds on scientific database technologies (array databases, SciQL [35], data vaults [33]) and Semantic Web technologies (stRDF and stSPARQL) implemented on top of the database system MonetDB and RDF store Strabon. [41] discusses how TELEIOS technologies are used to develop a fire monitoring service for the partner of TELEIOS National Observatory of Athens (NOA). As part of the processing chain underlying this service, geospatial data in the form of ESRI shapefiles are transformed into RDF. This geospatial data consists of longitude/latitude points that describe hotspots that have been detected using satellite image processing and classification techniques²¹. The OWL ontology (called the NOA ontology²²) links the generated hotspot products with linked geospatial data for Greece that has been developed by our group for TELEIOS (Coastline of Greece, Greek Administrative Geography), and with other linked geospatial data available on the Web (Corine Land Cover, LinkedGeoData, GeoNames). These data sources have been developed with the aim to allow NOA personnel to quickly develop maps depicting the progress of a forest fire front, burned area maps in the aftermath of a fire, etc. [41].

3 Background on OGC Standards

In this section we present some well-known standards developed by the Open Geospatial Consortium (OGC)²³. OGC is an international consortium of companies, government agencies and universities participating in a consensus process to develop publicly available interoperability standards for geospatial data. OGC standards focus on solutions for geospatial data and services, GIS data processing and geospatial data sharing.

The OGC Abstract Specification²⁴ is the conceptual foundation for the OGC interoperability specifications. The purpose of this specification is to define a conceptual model which includes the core concepts related to geospatial data, for which OGC standards will be developed in other technical documents. The OGC Abstract Specification also defines the basic terminology to be used in the rest of the OGC specifications. A small subset of this terminology (which might seem unfamiliar to a newcomer) will be used many times in the rest of this paper, thus we introduce it here.

²⁰ <http://www.earthobservatory.eu/>

²¹ A hotspot is a pixel of a satellite image corresponding to a geographic region that is probably on fire.

²² <http://www.earthobservatory.eu/ontologies/noaOntology.owl>

²³ <http://www.opengeospatial.org/>

²⁴ <http://www.opengeospatial.org/standards/as>

In OGC terminology, a *geographic feature* (or simply *feature*) is an abstraction of a real world phenomenon and can have various attributes that describe its *thematic* and *spatial* characteristics. For example, a feature can represent an airport. Thematic information about an airport can include its name, the company that manages it, etc., while a spatial characteristic is its location on Earth. The spatial characteristics of a feature are represented using *geometries* such as points, lines, and polygons. Each geometry is associated with a *coordinate reference system* which describes the coordinate space in which the geometry is defined.

The organization of the rest of this section is as follows. In Section 3.1 we will define the main concepts underlying coordinate reference systems, and give some background information on the coordinate reference systems we most often find in applications. In many research papers coordinate systems are given only a passing mention, and the well-known Cartesian plane (or some subset of it) is assumed to be the geographic space occupied by features. However, this assumption is not always true in applications, so the following section surveys the most popular coordinate systems that are in use today. In Section 3.2 we will present the Well-Known Text OGC standard that can be used for representing geometries and their coordinate reference systems. In Section 3.3 we will present the OpenGIS Simple Feature Access standard that defines a standard SQL schema that supports storage, retrieval, query and update of collections of features using SQL. This standard is today used in all relational DBMSs that offer support for geospatial data. It is also the basis for the stSPARQL and GeoSPARQL query languages discussed later, thus it is covered in detail. In Section 3.4 we will present the Geography Markup Language which is a more recent standard defined by OGC for representing geographical features in XML. Both of these standards have been recently used in data models and query languages for linked geospatial data, thus we present them in detail here.

3.1 Coordinate Reference Systems

A *coordinate* is one of n scalar values that determines the position of a point in an n -dimensional space. A *coordinate system* is a set of mathematical rules for specifying how coordinates are to be assigned to points. For example, the well-known Cartesian coordinate system assigns positions to points relative to n mutually perpendicular axes. A *coordinate reference system* is a coordinate system that is related to an object (e.g., the Earth, a planar projection of the Earth, a three dimensional mathematical space such as \mathbb{Z}^3) through a so called *datum* which specifies its origin, scale, and orientation. In the relevant literature, a coordinate reference system is also referred to as a *spatial reference system*. Various kinds of coordinate reference systems exist. In what follows we discuss geographic and projected coordinate systems because they are the ones most often found in GIS applications, and have also been used by previous research in the area covered by this survey paper.

A *geographic coordinate reference system* is a three-dimensional coordinate system that utilizes latitude, longitude, and optionally elevation, to capture ge-

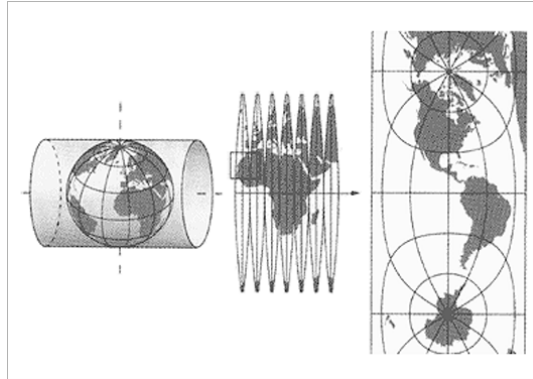


Fig. 1. The transverse Mercator projection of UTM

ographic locations on Earth. Detailed definitions for latitude, longitude and elevation (technically, geodetic height) can be found in [51]. The World Geodetic System (WGS) is the most well-known geographic coordinate reference system and its latest revision is WGS84²⁵. WGS84 is the reference coordinate system used by the Global Positioning System.

Although a geographic coordinate system such as WGS84 is a comprehensive way to describe locations on Earth, some applications work on a *projection* of the Earth even though there is a price to pay in terms of distortions that such a projection causes. In these cases a *projected coordinate reference system* is used that transforms the 3-dimensional ellipsoid approximation of the Earth into a 2-dimensional surface. In general, projected coordinate reference systems are always associated with a geographic coordinate system and it is important to understand the compromises made by it when computing the projection of the Earth.

An example of a projected coordinate reference system with world coverage is the Universal Transverse Mercator (UTM) system. UTM uses the WGS84 ellipsoid approximation of the Earth as the underlying geographic coordinate system. It is based on the transverse Mercator projection of the Earth on a 2-dimensional plane. Intuitively, this projection is obtained if we form a cylinder by wrapping a piece of paper around the Earth's poles, projecting on it the ellipsoid of Earth, and then unwrapping it (see Figure 1). As shown on the same figure, UTM is not a single projection system. It is based on a grid which divides the Earth into sixty zones of equal width, so that each zone can use a fine-tuned transverse Mercator projection that is capable of projecting the corresponding region of the Earth with a low amount of distortion.

Individual countries or states (e.g., in the USA) have their own projected coordinate reference systems that are more precise for their geographic area. For

²⁵ <http://en.wikipedia.org/wiki/WGS84/>

example, the Greek Geodetic Reference System 1987 (GGRS87)²⁶ is a projection system commonly used in Greece which is based on the local coordinate reference system Geodetic Reference System 1980 (GRS80)²⁷.

Various authorities provide information about popular coordinate reference systems. For example, OGC maintains a list with the URIs of various systems²⁸. The European Petroleum Survey Group (EPSG) also provides a big collection of coordinate reference systems²⁹. The identifiers of coordinate reference systems assigned by these authorities are used in geospatial data standards, e.g., Well-Known Text, to be discussed below.

3.2 Well-Known Text

Well-Known Text (WKT) is a widely used OGC standard for representing geometries. WKT can be used for representing geometries, coordinate reference systems, and transformations between coordinate reference systems. WKT is described in the “OpenGIS Simple Feature Access - Part 1: Common Architecture” specification [4] that is the same as the ISO 19125-1 standard. This standard concentrates on ways to represent and manipulate simple features. A *simple feature* is a feature with all spatial attributes described piecewise by a straight line or a planar interpolation between sets of points.

Geometries in WKT are restricted to 0-, 1- and 2- dimensional geometries that exist in \mathbb{R}^2 , \mathbb{R}^3 , or \mathbb{R}^4 . Geometries that exist in \mathbb{R}^2 consist of points with coordinates x and y , e.g., POINT(1 2) in WKT syntax. Geometries that exist in \mathbb{R}^3 consist of points with coordinates x , y , and z , or x , y , and m where m is a measurement. For example, the point POINT(37.96 23.71 27) might be used to represent the temperature of the city of Athens measured in Celcius degrees; where 37.96 is the latitude of Athens, 23.71 its longitude, and 27 its temperature. Geometries that exist in \mathbb{R}^4 have points with coordinates x , y , z , and m with similar semantics.

Geometries represented using WKT have the following properties:

- All geometries are topologically closed which means that all the points that comprise the boundary of the geometry are assumed to belong to the geometry, even though they may not be explicitly represented in the geometry.
- All coordinates within a geometry are in the same coordinate reference system.
- For geometric objects that exist in \mathbb{R}^3 and \mathbb{R}^4 , spatial operations work on their “map geometries”, that is, their projections on \mathbb{R}^2 . Therefore, the z and m values are not reflected in calculations (e.g., when invoking functions equals, intersects, etc. defined in Section 3.3 below) or in the generation of new geometry values (e.g., when invoking functions buffer, minimum bounding box, etc. defined in Section 3.3). Thus, the WKT specification is

²⁶ <http://spatialreference.org/ref/epsg/4121/>

²⁷ <http://spatialreference.org/ref/epsg/6121/>

²⁸ <http://www.opengis.net/def/crs/>

²⁹ <http://www.epsg-registry.org/>

inherently about 2-dimensional geometries, but it also allows z and m values associated with these geometries and functions to access them.

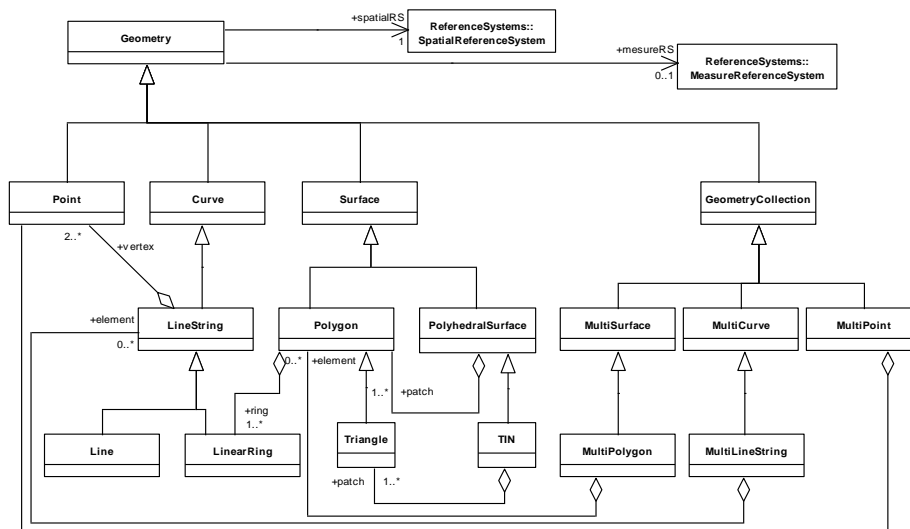


Fig. 2. The classes of geometries in WKT (figure from [4])

Let us now present the part of the standard that defines how to represent vector geometries. In Figure 2 we present the class hierarchy for simple feature geometries represented in WKT as defined in [4]. The top Geometry class has subclasses Point, Curve, Surface, and Geometry Collection. The Geometry Collection is further specialized to classes of 0-, 1- and 2- dimensional geometries named MultiPoint, MultiLineString and MultiPolygon respectively. Each geometry is linked to a specific coordinate reference system and optionally to a measure reference system. A *measure reference system* may be used to interpret the third or fourth dimension of geometries that exist in \mathbb{R}^3 or \mathbb{R}^4 . For example, a fire hotspot can be modeled as a point that has x , y , and m coordinates, where the coordinates x and y are used to represent the location of the hotspot, while the m coordinate represents the measured temperature.

Let us provide some more information for each class depicted in Figure 2.

- **Point.** A point represents a single location in coordinate space. A point has x and y coordinate values and may have z and m depending on the associated coordinate reference system.
- **Curve.** A curve is a 1-dimensional geometry. The subtypes of class curve define the type of interpolation that is used between points.
- **LineString.** A line string is a subtype of class curve that uses linear interpolation between points. A line string is closed if its start point is equal to its end point. A line string is simple if it has no self-intersections.

- **Line.** A line is a line string with exactly two points.
- **LinearRing.** A linear ring is a line string that is both closed and simple.
- **Surface.** A surface is a 2-dimensional geometry. This class is abstract (i.e., it may not be instantiated). A simple surface may consist of a single “patch” that has one “exterior” boundary and 0 or more “interior” boundaries (e.g. a polygon with holes).
- **Polygon.** A polygon is a simple surface that is planar. It has exactly one exterior boundary and may have several non-intersecting interior boundaries. Each polygon is topologically closed and no two boundaries (interior or exterior) cross. However, two boundaries may intersect at a point, but only as a tangent. The interior of a polygon is a connected point-set while the exterior of a polygon with holes is not connected.
- **Triangle.** A triangle is a polygon with 3 distinct, non-collinear vertices and no interior boundary.
- **Polyhedral Surface.** A polyhedral surface is a contiguous collection of polygons which share common boundary segments. Each pair of polygons that touch has a common boundary that is expressed as a finite collection of line strings. Each such line string is a part of the boundary of at most 2 polygon patches.
- **Triangulated Irregular Network.** A triangulated irregular network is a polyhedral surface consisting only of triangle patches.
- **Geometry Collection.** A geometry collection is a set of distinct geometries.
- **MultiPoint.** This is a geometry collection whose elements are points that are not connected.
- **MultiCurve.** A multi-curve is a geometry collection whose elements are curves.
- **MultiLineString.** A multi-line string is a geometry collection whose elements are line strings.
- **MultiSurface.** A multi-surface is a 2-dimensional geometry collection whose elements are surfaces. The geometric interiors of any two surfaces may not intersect. The boundaries of any two surfaces may not cross but may touch at a finite number of points.
- **MultiPolygon.** A multi-polygon is a multi-surface collection whose elements are polygons. The boundaries of each polygon may not intersect.


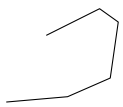
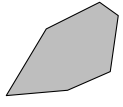
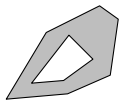
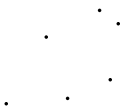
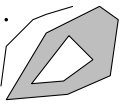
The syntax of the WKT representation of a geometry is presented in detail in [4]. Some examples of geometries represented in WKT are shown in Table 1.

The interpretation of the coordinates of a geometry depends on the coordinate reference system that is associated with the geometry. Note that according to the WKT standard, the coordinate reference system that is associated to a geometry is never embedded in the object’s representation, but is given separately using appropriate notation.

3.3 OpenGIS Simple Feature Access

The “OpenGIS Simple Feature Access - Part 2: SQL Option” standard [3] defines a standard SQL schema that supports storage, retrieval, query and update

Table 1. Examples of geometries represented in WKT

Geometry type	WKT representation	Geometry
Point	Point(5 5)	
LineString	LineString(5 5,28 7,44 14,47 35,40 40,20 30)	
Polygon	Polygon((5 5,28 7,44 14,47 35,40 40,20 30,5 5))	
Polygon	Polygon((5 5,28 7,44 14,47 35,40 40,20 30,5 5), (28 29,14.5 11,26.5 12,37.5 20,28 29))	
MultiPoint	MultiPoint((5 5),(28 7),(44 14), (47 35),(40 40),(20 30))	
Geometry Collection	GeometryCollection(Point(5 35), LineString(3 10,5 25,15 35,20 37,30 40), Polygon((5 5,28 7,44 14,47 35,40 40,20 30,5 5), (28 29,14.5 11,26.5 12,37.5 20,28 29)))	

of sets of simple features using SQL. This OGC standard is the same as the ISO 19125-2 standard. The simple features supported by this standard have both spatial and non-spatial (thematic) attributes. The spatial attributes are geometries of the types described in Section 3.2. This standard assumes that sets of simple features are stored as relational tables and each feature is a row in a table. The spatial attributes of the features are represented as geometry-valued columns, while non-spatial attributes are represented as columns whose types are the standard SQL data types. There are also additional tables that are used to store information about features and coordinate reference systems. The standard describes schemas for two types of feature table implementations: implementa-

tions using only the SQL predefined data types and SQL with geometry types. Only the latter approach is covered here.

The “SQL with geometry types” approach uses WKT geometry classes presented in Section 3.2 to define new geometric data types for SQL together with SQL functions on those types.

The following SQL functions have been defined for requesting the desired representation of a geometry, checking whether some condition holds for a geometry and returning some properties of the geometry:

1. `ST_Dimension(A:Geometry):Integer`, returns the inherent dimension of the geometry A, which must be less than or equal to the coordinate dimension³⁰.
2. `ST_GeometryType(A:Geometry):String`, returns the name of the instantiable subtype of Geometry as defined in [4], of which the geometry A is an instantiable member.
3. `ST_AsText(A:Geometry):String`, exports the geometry A as its WKT representation.
4. `ST_AsBinary(A:Geometry):Binary`, exports the geometry A as its Well-Known Binary³¹ representation.
5. `ST_SRID(A:Geometry):Integer`, returns the coordinate reference system identifier for the geometry A.
6. `ST_IsEmpty(A:Geometry):Boolean`, returns true if the geometry A is the empty geometry. Otherwise, it returns false.
7. `ST_IsSimple(A:Geometry):Boolean`, returns true if the geometry A has no anomalous geometric points, such as self intersection or self tangency. Otherwise, it returns false.

The following SQL functions have been defined for testing topological spatial relationships between two geometries. These functions correspond to the relations from the dimensionally extended 9-intersection model of Egenhofer and Clementini defined in [14].

1. `ST_Equals(A:Geometry, B:Geometry):Boolean`, returns true if the geometry A is “spatially equal” to the geometry B. Otherwise it returns false.
2. `ST_Disjoint(A:Geometry, B:Geometry):Boolean`, returns true if the geometry A is “spatially disjoint” to the geometry B. Otherwise it returns false.
3. `ST_Intersects(A:Geometry, B:Geometry):Boolean`, returns true if the geometry A “spatially intersects” the geometry B. Otherwise it returns false.
4. `ST_Touches(A:Geometry, B:Geometry):Boolean`, returns true if the geometry A “spatially touches” the geometry B. Otherwise it returns false.

³⁰ The prefix ST is a historical one and comes from the spatial part of the SQL/MM standard [72]. It was meant to denote “spatial and temporal”, but temporal issues were not included finally in that standard.

³¹ Well-Known Binary is an equivalent to the WKT format for the representation of a geometry and is also defined in [4]. The object is represented as a contiguous stream of bytes, thus facilitating its exchange between a database client and server.

5. `ST_Crosses(A:Geometry, B:Geometry):Boolean`, returns true if the geometry A “spatially crosses” the geometry B. Otherwise it returns false.
6. `ST_Within(A:Geometry, B:Geometry):Boolean`, returns true if the geometry A is “spatially within” to the geometry B. Otherwise it returns false.
7. `ST_Contains(A:Geometry, B:Geometry):Boolean`, returns true if the geometry A “spatially contains” the geometry B. Otherwise it returns false.
8. `ST_Overlaps(A:Geometry, B:Geometry):Boolean`, returns true if the geometry A “spatially overlaps” the geometry B. Otherwise it returns false.
9. `ST_Relate(A:Geometry, B:Geometry, intersectionPatternMatrix:String):Boolean`, returns true if the geometry A is “spatially related” to the geometry B by testing for intersections between the interior, boundary and exterior of the two geometries as specified by the values in the `intersectionPatternMatrix` according to the Egenhofer and Clementini intersection pattern matrix (DE-9IM) of [13,14].

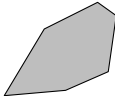
The following SQL functions have been defined for constructing new geometric objects from existing geometries.

1. `ST_Boundary(A:Geometry):Geometry`, returns a geometry that is the boundary of the geometry A.
2. `ST_Envelope(A:Geometry):Geometry`, returns a geometry that is the minimum bounding box for the input geometry A.
3. `ST_Intersection(A:Geometry, B:Geometry):Geometry`, returns a geometry that represents the point set intersection of the geometries A and B.
4. `ST_Union(A:Geometry, B:Geometry):Geometry`, returns a geometry that represents the point set union of the geometries A and B.
5. `ST_Difference(A:Geometry, B:Geometry):Geometry`, returns a geometry that represents the point set difference of the geometries A and B.
6. `ST_SymDifference(A:Geometry, B:Geometry):Geometry`, returns a geometry that represents the point set symmetric difference of the geometries A and B.
7. `ST_ConvexHull(A:Geometry):Geometry`, returns a geometry that represents the convex hull of the geometry A.
8. `ST_Buffer(A:Geometry, distance:Double):Geometry`, returns a geometry that represents all points whose distance from the geometry A is less than or equal to `distance`. The relevant calculation is done in the coordinate reference system of this geometry.

The `ST_Distance(A:Geometry, B:Geometry):Double` SQL function is defined for calculating the shortest distance between two geometries. The calculated scalar value corresponds to the shortest distance of these two geometries measured in the unit system of their coordinate reference system.

The standard ISO 13249 SQL/MM is an international standard for multimedia and other application extensions of SQL. Part 3 of this standard defines a set of types and methods for representing, processing, storing and querying spatial data in relational databases [72,54]. The SQL/MM standard for spatial data is very close to the OpenGIS Simple Feature Access standard presented

Table 2. Examples of geometries represented in GML

Geometry type	GML representation	Geometry
Point	<pre><gml:Point gml:id="p1" srsName="urn:ogc:def:crs:EPSG:6.6:4326"> <gml:coordinates>5,5</gml:coordinates> </gml:Point></pre>	.
Polygon	<pre><gml:Polygon gml:id="p3" srsName="urn:ogc:def:crs:EPSG:6.6:4326"> <gml:exterior> <gml:LinearRing> <gml:coordinates> 5,5 28,7 44,14 47,35 40,40 20,30 5,5 </gml:coordinates> </gml:LinearRing> </gml:exterior> </gml:Polygon></pre>	

here and, in fact, the two efforts influenced each other. Therefore, we do not give any details about it in this paper.

3.4 Geography Markup Language

The Geography Markup Language (GML) [1] is the most common XML-based encoding standard for the representation of geospatial data. GML was developed by the OGC and it is based on the OGC Abstract Specification. GML provides XML schemas for defining a variety of concepts that are of use in Geography: geographic features, geometry, coordinate reference systems, topology, time and units of measurement. Initially, the GML abstract model was based on RDF and RDFS, but later the consortium decided to use XML and XML Schema. The GML *profiles* are logical restrictions of GML that might be of use to applications that do not want to use the whole of GML. GML profiles can be specified through an XML document, an XML schema, or both. Some of the profiles that have been proposed for public use are: (i) Point Profile, which defines a simple point geometry in GML, (ii) GML Simple Features Profile, which is the GML encoding of Simple Features for SQL discussed in Section 3.3, (iii) a GML profile for JPG, and (iv) a GML profile for RSS. It should be noted that GML profiles are different from application schemas. The profiles are part of the GML namespaces (OpenGIS GML) and define restricted subsets of GML. Applications schemas are XML vocabularies that are application-specific and are valid

inside the application-specific namespaces. Application schemas can be built on specific GML profiles or use the full GML specification.

The GML Simple Features Profile [2] and the Simple Features for SQL presented in Section 3.3 have similar structure and describe similar geometries. However, the GML Simple Features Profile can also have geometries in three dimensions while Simple Features for SQL can have geometries of up to only two dimensions. In the GML Simple Features Profile, a feature can have any number of geometric properties, and every geometry should be referenced to a coordinate reference system that has 1, 2 or 3 dimensions.

Since the GML Simple Features Profile can represent similar geometries with WKT, we present in Table 2 two examples only showing the GML representation of a point and a polygon. The complete syntax of the GML representation of a geometry is presented in [1].

GML completes our discussion of OGC standards. We now move to discuss a category of successful systems that use these standards today: relational DBMSs with geospatial data support.

4 Spatial Relational Database Systems

The development of spatial database systems has gained a lot of attention since the early 90s [67]. Researchers in spatial databases have studied in depth all the relevant research topics including data modeling, query languages, indexing and query processing, system architectures and implementations. In the area of spatial data modeling, researchers have studied possible representations of geographic data by extending the relational model appropriately, e.g., by spatial abstract data types [26] or constraints [66]³². In the former approach, one defines a set of spatial data types and a set of functions that operate on elements of these data types. For example, one could define data types and functions that correspond to the geometry classes and functions of the OpenGIS Simple Features Access standard presented earlier in Section 3.3, and, in fact, this is how commercial DBMSs with a geospatial extension implement this standard as we will see below. In the latter approach, the standard relational model is extended with infinite relations that are finitely represented by constraints expressed in an appropriate first-order constraint language, e.g., linear constraints [34,64]. Linear constraint relations can easily represent spatial data (e.g., all the points of the union of two convex polygons with holes representing the location of a forest on the map), and the well-known relational algebra can be used for querying these relations [24,21]. The most important data models in this area are the ones of the system Dédale [66] and the query language CSQL, a constraint-based extension of SQL presented in [47].

³² Extensions of other models, e.g., object-oriented have been considered [69,53]. We do not deal with these models here due to space considerations. To the best of our knowledge, these works have not resulted in implemented spatial DBMSs that are available commercially today.

Independently of whether the assumed data model was based on datatypes or constraints, work on query languages mainly concentrated on designing query languages for spatial relational DBMSs (e.g., spatial extensions of the relational algebra [27] or SQL [18]) and graphical representations for both the input queries and the output results. Work on indexing and query processing has concentrated on spatial indices, e.g., R-trees [29,68], and efficient query processing techniques for the proposed query languages. Finally, in the areas of system architectures and implementations, researchers dealt with integrating the proposed spatial extensions into standard DBMS architectures. Here most of the work focused on the spatial datatypes approach which was also followed by commercial DBMSs as we will see below. The constraint-based approach resulted mostly in many interesting theoretical results (see, e.g., the book [48] and the survey paper [46]) with only a few prototypes being built [66,64], none of which has any industrial relevance today. For a nice survey of spatial database research, the interested reader can refer to the survey [26] (which is now rather dated) or the more recent textbooks [67,68].

Due to the importance of spatial information in applications, the research advances in spatial databases were immediately picked up by the database industry. Thus, most of the open source or commercial DBMSs available today offer support for spatial data. For example, PostGIS³³ is an open source software which extends PostgreSQL with the ability to store and query geospatial data. Similar support is available in commercial systems such as Oracle Spatial³⁴.

Most of today's spatial DBMSs support the standards we discussed in Section 3 and especially the OpenGIS Simple Feature Access for SQL and the ISO 13249 SQL/MM. For the exact details of how each system implements these standards, the reader is advised to consult the relevant system manuals. For example, PostGIS provides different datatypes for each standard, e.g., the `ST_Geometry` datatype is based on the ISO SQL/MM specification while the `Geometry` datatype follows the OpenGIS Simple Feature Access for SQL.

More recently, database researchers have also concentrated on *spatiotemporal databases* which deal with geometries changing over time, e.g., as in the case of moving objects. An overview of this area of research which is still very active today is given in the books [28,65,45]. The results of this research have not found their way to commercial systems yet, but comprehensive prototypes exist, e.g., Secondo³⁵.

5 XML-based Languages for Spatial Data

To the best of our knowledge, there has been so far no significant work in extending the XML data model and related query languages with spatial data for the purposes of building geospatial XML databases. However, since XML is the technology of choice for exchanging information on the Web, there has

³³ <http://postgis.refractory.net/>

³⁴ <http://www.oracle.com/technology/products/spatial/index.html>

³⁵ <http://dna.fernuni-hagen.de/Secondo.html>

been significant previous work in using XML to encode geospatial data that can be queried, if needed, using standard XML query languages such as XPath and XQuery. The most important work in this area is work on GML, its profiles and various other languages that build on it. Since GML is an OGC standard, it was discussed in Section 3.4, so we do not repeat it here.

GeoRSS³⁶ is an XML specification that enables RSS feeds to encode location. GeoRSS-Simple and GeoRSS-GML are two different encodings of GeoRSS. GeoRSS-Simple is a very lightweight format that developers and users can quickly and easily add to their existing feeds with little effort. It supports basic geometries (point, line, box, polygon) and covers the typical use cases when encoding locations. For a more feature-rich option, GeoRSS GML is a formal GML profile, and supports a greater range of features, notably coordinate reference systems other than WGS84 latitude/longitude.

Since Web document presentation languages such as HTML5 can be encoded in XML (see work on XHTML) and the spatial layout of Web documents is an important aspect of presentation, the authors of [55] have proposed extensions of XML and XPath that can be used to capture not just the content, but also the spatial layout of a Web page. The authors of [55] define the spatial document object model (SDOM) which enables the representation of structural, as well as spatial, aspects of HTML pages. They have also defined the query language Spatial XPath (SXPath), which is an extension of XPath 1.0 that includes spatial navigation of SDOM documents. SXPath extends XPath with spatial axes and spatial position functions. Spatial axes allow the selection of document nodes by taking into account their spatial relationship with the context node. Spatial position functions allow the selection of document nodes by taking into account their spatial position on the Cartesian plane with respect to the context node. This is consistent with the fact that layout algorithms used by Web browsers view the area of the screen that will be used to visualize a Web page as a Cartesian plane. Therefore, the discovery of visual patterns on a Web page can be expressed as an SXPath query that exploits the relative spatial position of images and text. In SDOM spatial relationships between minimum bounding rectangles are encoded using the Rectangle Algebra [9]. However, since the relationships of the Rectangle Algebra are too fine-grained, the more intuitive relationships of the Rectangular Cardinal Relations model [62] and the Region Connection Calculus [61] are used in SXPath.

[55] also presents theoretical and implementation results concerning a subset of SXPath that appears to capture the most useful queries. Finally, this subset is experimentally evaluated regarding its usability for developing wrappers for Deep and Social Web sites.

6 Geospatial Extensions to RDF and SPARQL

Contrary to XML, much work has been done recently on extending RDF for representing geospatial information. This work has recently resulted in the defi-

³⁶ <http://www.georss.org/>

inition of the query language GeoSPARQL as an OGC standard [5]. This section surveys most of the relevant efforts in this area and it is organized as follows. First we discuss early proposals that paved the way for more mature languages such as stSPARQL (defined by our group) and GeoSPARQL. Then, these two languages are presented in more detail including examples.

6.1 Early Works on Representing Geospatial Information in RDF

Starting with the paper [58], a series of interesting contributions to the representation of geospatial (and temporal) information in the Semantic Web were done by Amit Sheth's group [59,57,60,58,30,70,7]. From these, the work closest to the topic of this survey paper is the Ph.D. dissertation of Matthew Perry [57] which proposed an extension of SPARQL, called SPARQL-ST, that allows one to query spatial and non-spatial data that change over time. The main idea of [57] is to incorporate geospatial information to the temporal RDF graph model of [25]. [25] presents an extension of RDF which allows one to represent when a triple is valid in reality. The main concept for achieving this is that of a *temporal triple* which is an RDF triple with an additional temporal label (a time point represented by a natural number). The corresponding notation used is, $(s, p, o)[t]$ which denotes the fact that triple (s, p, o) is valid at time t . Triples valid at time intervals are then defined by sets of temporal triples valid at time points. Finally, a *temporal RDF graph* is defined as a set of temporal RDF triples.

Geospatial information in [57] is captured using an upper ontology for modeling theme, space and time expressed in RDFS. The spatial part of this upper ontology uses the class `geo:SpatialRegion` and its subclasses (e.g., `geo:Polygon`) defined using the GeorSS GML profile in order to model spatial geometries (e.g., polygons). A geographic object (e.g., a town) can then be connected to its spatial geometry (e.g., a polygon) using the property `stt:located_at`. Spatial geometries in [57] are specified by sets of RDF triples that give various details of the geometry depending on its type (e.g., for a 2-dimensional polygonal area, they give the coordinates of its boundary and the relevant coordinate reference system).

The query language SPARQL-ST introduces two new types of variables, namely spatial and temporal ones, in addition to the standard SPARQL variables. Spatial variables (denoted by a % prefix) represent complex spatial features rather than a simple URI, and the concept of SPARQL mappings has been extended in [57] to map a spatial variable to a set of triples that represent the required spatial information. Similarly, temporal variables (denoted by a # prefix) are mapped to time intervals and can appear in the fourth position of a temporal triple pattern in the style of [25]. 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 (OGC Simple Feature Access topological relations and distance for the spatial part, and Allen's interval calculus [6] for the temporal part). In order to enable the realization of this query language, the used spatial and temporal operators need to be implemented. Both spatial and temporal operators are implemented using Oracle's

extensibility framework, while the strictly RDF concepts are implemented using Oracle’s RDF storage and inferencing capabilities. For more details of the implementation see [57,60].

In parallel with the work by Sheth’s group we surveyed above, Dave Kolas and his colleagues at Raytheon BBN Technologies studied similar questions in [37,38,36]. In [38], the use of RDF for representing spatial data in the Semantic Web is proposed in the context of the prototype system SPAUK (Spatially Augmented Knowledge Base). In SPAUK, geometric attributes of a resource (e.g., location of a gas station) are represented in RDF by introducing a blank node for the geometry, specifying the geometry using GML vocabulary [1], and associating the blank node with the resource using GeoRSS vocabulary. Queries are expressed in the SPARQL query language utilizing appropriate geometric vocabularies and ontologies (e.g., the topological relationships of RCC [16]). The main assumption of this work is that SPARQL should not be extended with new features for querying spatial data; instead, the existing features of SPARQL together with spatial vocabularies should be utilized. SPAUK has been implemented by storing RDF triples in Jena and using an in-memory grid file to index the geometries.

Kolas later revisited the problem of defining a Semantic Web data model and query language for spatial data in [36]. This paper assumes the RDF-based spatial data representation of [38] and discusses various ways to exploit what is already available in SPARQL to pose queries. The options compared are: (i) to use SPARQL as in [38], (ii) to introduce a new PREMISE clause in SPARQL that could be used to introduce spatial geometries that can be used in a query, and (iii) to use some form of the DESCRIBE query form of SPARQL for asking queries about geometries.

Our group entered the scene in 2010 with the paper [40] where the language stSPARQL was presented. [40] was critical (of earlier works such as [57,38,36] for their lack of theoretical foundations (especially a formal semantics) and proposed to introduce time and space in the Semantic Web formally by following the theoretically elegant paradigm of spatial and temporal constraint databases [34,65,64,39]. [40] represents spatial geometries by semi-linear point sets in the n -dimensional space \mathbb{Q}^n , i.e., sets that can be defined by quantifier-free formulas in the first-order logic of linear equations and inequalities over \mathbb{Q}^n . Semi-linear sets can capture a great variety of spatial geometries, e.g., points, lines, line segments, polygons, k -dimensional unions of convex polygons possibly with holes, thus they allow a lot of expressive power [75]. Similarly, the valid times of a triple is represented using temporal constraints (a very restricted class of linear constraints).

Following the approach of Dédale [66] and CSQL [47], we developed a constraint-based extension of RDF, called stRDF, that can be used to represent thematic and spatial data that might change over time [40]. Technically, the standard RDF notion of a triple is extended in stRDF, so that the object of a triple is allowed to be a quantifier-free formula with linear constraints (i.e., a finite representation of a semi-linear subset of \mathbb{Q}^n). In terms of the W3C spec-

ification of RDF, this spatial extension can be realized with the introduction of a new kind of *typed literals*: quantifier-free formulas with linear constraints. [40] had introduced the data type `strdf:SemiLinearPointSet` for this purpose. The values of this datatype are typed literals (called *spatial* literals) that encode geometric objects using Boolean combinations of linear constraints in \mathbb{Q}^2 . For example,

$$(x \geq 0 \wedge y \geq 0 \wedge x + y \leq 1) \vee (x \leq 0 \wedge y \leq 0 \wedge x + y \geq -1)$$

is such a literal encoding the union of two polygons in \mathbb{Q}^2 . To capture the validity time of a triple, [40] followed [25] by adding a fourth component to each RDF triple (i.e., similarly to [57]). Diverging from [25], this component is also a quantifier-free formula that expresses a temporal constraint. The solution set of this temporal constraint gives us the times that the triple is valid.

[40] also presented an extension of SPARQL, called stSPARQL, to query spatial and temporal data expressed in stRDF, in a declarative way. stSPARQL was introduced by example and a detailed semantics using the algebraic approach pioneered for SPARQL in [56] was presented. Later on, the computational complexity of evaluating stSPARQL was also studied in [44]. From a constraint databases perspective, stSPARQL follows closely the ideas of CSQL [47] and to a lesser extent the ideas of Dédale [66]; this allows us to have a useful language for expressing spatial and temporal queries while maintaining closure (i.e., staying within the realm of semi-linear point sets).

The model stRDF and the query language stSPARQL have been implemented by our group in the system Strabon³⁷, and used as the data model and query language of the registry service of the Semantic Sensor Web infrastructure developed by European FP6 project SemsorGrid4Env³⁸ (see [23] for the infrastructure and [49] for the implementation of Strabon).

In SemsorGrid4Env, we quickly realized that the theoretical elegance of stRDF and stSPARQL was not enough for achieving end-user acceptance in an application world ruled by OGC standards. Thus, starting in SemsorGrid4Env and continuing in FP7 project TELEIOS, we developed a new version of stRDF which respects our initial design decisions (especially the decision to represent geometries using literals), but uses OGC standards for the representation of geospatial data. In addition, we developed a new version of stSPARQL which extends SPARQL 1.1 for querying stRDF data [43]. Our work on this new version of stSPARQL has commonalities with the recent OGC work on GeoSPARQL as we will see below³⁹.

³⁷ <http://www.strabon.di.uoa.gr/>

³⁸ <http://www.sensorsgrid4env.eu/>

³⁹ In fact, the design of the new version of stSPARQL has partly been inspired by the presentation [52] which was kindly made available to us by Xavier Lopez of Oracle who participates in the relevant OGC working group, after he attended the presentation of [40] at ESWC 2010 (of course, the main inspiration has been our independent work in SemsorGrid4Env discussed above, and especially the suggestions of project members working with OGC standards for a long time). We also became members

6.2 The New Versions of stRDF and stSPARQL

In the new version of stRDF [43] we use OGC standards for the representation of geospatial data. The datatypes `strdf:WKT` and `strdf:GML` are introduced to represent geometries serialized using the OGC standards WKT and GML which were presented in Section 3.4.

Given the OGC specification for WKT, the datatype `strdf:WKT`⁴⁰ is defined as follows. The lexical space of this datatype includes finite-length sequences of characters that can be produced from the WKT grammar defined in the WKT specification, optionally followed by a semicolon and a URI that identifies the corresponding CRS. The default case is considered to be the WGS84 coordinate reference system. The value space is the set of geometry values defined in the WKT specification. These values are a subset of the union of the powersets of \mathbb{R}^2 and \mathbb{R}^3 .

The lexical and value space for `strdf:GML` are defined similarly. In this case, since the GML grammar allows us to state coordinate reference systems for the geometries we define, we do not have a separate component for them in `strdf:GML` literals as we do for `strdf:WKT` literals.

The datatype `strdf:geometry` is also introduced to represent the serialization of a geometry independently of the serialization standard used. The datatype `strdf:geometry` is the union of the datatypes `strdf:WKT` and `strdf:GML`, and appropriate relationships hold for their lexical and value spaces.

Both the original [40] and the new version of stRDF presented in this paper impose minimal new requirements to Semantic Web developers that want to represent spatial objects with stRDF; all they have to do is utilize a new literal datatype. These datatypes (`strdf:WKT`, `strdf:GML` and `strdf:geometry`) can be used in the definition of geospatial ontologies needed in applications, e.g., ontologies similar to the ones defined in [57]. The same approach based on spatial literals has also recently been used independently in the paper [12] and also in the GeoSPARQL standard [5] as we will see in Section 6.3 below. The datatype `strdf:geometry` can be used in the definition of geospatial ontologies that would like to use stRDF, e.g., ontologies similar to the ones defined in [57,5]. Let us now give some examples of modeling thematic and spatial data in stRDF (Turtle notation is used).

Example 1. stRDF triples derived from GeoNames⁴¹ that represent information about the Greek towns Olympia and Zacharo including an approximation of their geometries.

```
geonames:264637 geonames:name "Olympia";
                owl:sameAs dbpedia:Olympia_Greece;
```

of OGC in January 2011 and have, since then, followed the work on GeoSPARQL closely.

⁴⁰ <http://strdf.di.uoa.gr/ontology>

⁴¹ <http://www.geonames.org/>

```

        rdf:type noa:Town;
        strdf:hasGeometry "POLYGON((21.5 18.5, 23.5 18.5,
                                   23.5 21, 21.5 21, 21.5 18.5));
        <http://www.opengis.net/def/crs/EPSSG/0/4326>"
        ^^strdf:WKT.

geonames:251283 geonames:name "Zacharo";
        rdf:type noa:Town;
        strdf:hasGeometry "POLYGON((19 19, 21 19, 21 21,
                                   19 21, 19 19));
        <http://www.opengis.net/def/crs/EPSSG/0/4326>"
        ^^strdf:WKT.

```

The above triples represent some information about the Greek towns Olympia and Zacharo including an approximation of their geometry modified for the purposes of our example. GeoNames normally utilizes the W3C Basic Geo vocabulary⁴² which is a basic ontology and OWL vocabulary for representing geospatial properties for Web resources. Instead of this, we use a typed literal of the data type `strdf:WKT` to define a polygon approximation of the geometry of the town encoded in WKT. The data type `strdf:WKT` (resp. `strdf:GML`) is similar to the data type `strdf:geometry` but is used to represent spatial objects using the WKT (resp. GML) serialization. Notice that a URI is used to denote that the coordinates of these geometries are given in the WGS84 using the syntax that we discussed above. This is also considered to be the default case in our approach, thus in this example the presence of this URI is optional.

Example 2. stRDF triples produced from the processing chain of the National Observatory of Athens (see Section 2) that represent burnt areas.

```

noa:BA1 a noa:BurntArea;
        noa:hasConfirmation noa:verified;
        strdf:hasGeometry "POLYGON((20 20, 20 22,
                                   22 22, 22 20, 20 20))"^^strdf:WKT.

noa:BA2 a noa:BurntArea;
        noa:hasConfirmation noa:verified;
        strdf:hasGeometry "POLYGON((23 18, 24 19,
                                   23 19, 23 18))"^^strdf:WKT.

noa:BA3 a noa:BurntArea.
        noa:hasConfirmation noa:verified;
        strdf:hasGeometry "POLYGON((20 15, 21 15,
                                   21 16, 20 15))"^^strdf:WKT.

```

The above triples describe burnt areas that have been generated by the National Observatory of Athens and will be utilized in the following examples of this section.

⁴² <http://www.w3.org/2005/Incubator/geo/XGR-geo/>

Let us now present the main features of the new version of stSPARQL. stSPARQL is an extension of SPARQL 1.1 with functions that take as arguments spatial terms and can be used in the SELECT, FILTER, and HAVING clause of a SPARQL 1.1 query. A *spatial term* is a spatial literal (i.e., a typed literal with datatype `strdf:geometry`), a query variable that can be bound to a spatial literal, the result of a set operation on spatial literals (e.g., union), or the result of a geometric operation on spatial terms (e.g., buffer).

The new version of stSPARQL extends SPARQL 1.1 with the machinery of the OpenGIS Simple Feature Access standard which was presented in Section 3. We achieve this by defining a URI for each of the SQL functions defined in the standard and use them in SPARQL queries. For example, for the function `ST_IsEmpty` defined in the OGC-SFA standard, we introduce the SPARQL extension function

```
xsd:boolean strdf:isEmpty(strdf:geometry g)
```

which takes as argument a spatial term `g`, and returns `true` if `g` is the empty geometry. Similarly, we have defined a Boolean SPARQL extension function for each topological relation defined in OGC-SFA (topological relations for simple features), [17] (Egenhofer relations) and [16] (RCC-8 relations). In this way stSPARQL supports multiple families of topological relations our users might be familiar with. Using these functions stSPARQL can express *spatial selections*, i.e., queries with a FILTER function with arguments a variable and a constant (e.g., `strdf:contains(?geo, "POINT(1 2)"^^strdf:WKT)`), and *spatial joins*, i.e., queries with a FILTER function with arguments two variables (e.g., `strdf:contains(?geoA, ?geoB)`).

The SPARQL extension functions corresponding to the SQL functions of the OpenGIS Simple Feature Access standard can be used in the SELECT clause of a SPARQL query. As a result, new spatial literals can be generated on the fly during query time based on pre-existing spatial literals. For example, to obtain the buffer of a spatial literal that is bound to the variable `?GEO`, we would use the following select expression:

```
SELECT (strdf:buffer(?geo,0.01) as ?geobuffer)
```

One of the new features in SPARQL 1.1 is support for aggregate functions. In stSPARQL we have introduced the following three aggregate functions that deal with geospatial data:

- `strdf:geometry strdf:union(set of strdf:geometry a)`, returns a geometric object that is the union of the set of input geometries.
- `strdf:geometry strdf:intersection(set of strdf:geometry a)`, returns a geometric object that is the intersection of the set of input geometries.
- `strdf:geometry strdf:extent(set of strdf:geometry a)`, returns a geometric object that is the minimum bounding box of the set of input geometries.

More aggregate functions may be added in the future if we identify a need for them in applications. stSPARQL also supports update operations (insertion, deletion, and update of stRDF triples) conforming to the declarative update

language for SPARQL, SPARQL Update 1.1, which is a current proposal of W3C.

The following examples demonstrate the functionality of stSPARQL.

Example 3. Return the names of towns that have been affected by fires.

```
SELECT ?name
WHERE { ?town a noa:Town;
        geonames:name ?name;
        strdf:hasGeometry ?townGeom.
        ?ba a noa:BurntArea;
        strdf:hasGeometry ?baGeom.
        FILTER(strdf:overlap(?townGeom,?baGeom))}
```

The result of the query evaluated on the data sets of Example 1 and 2 is displayed below:

?name
"Olympia"
"Zacharo"

The query above demonstrates how to use a topological function in a query. The results of this query are the names of the towns whose geometries “spatially overlap” the geometries corresponding to areas that have been burnt.

Example 4. Isolate the parts of the burnt areas that lie in coniferous forests.

```
SELECT ?ba (strdf:intersection(?baGeom,strdf:union(?fGeom)) AS ?burnt)
WHERE { ?ba a noa:BurntArea. ?ba strdf:hasGeometry ?baGeom.
        ?f a noa:Area. ?f noa:hasLandCover noa:coniferousForest.
        ?f strdf:hasGeometry ?fGeom.
        FILTER(strdf:intersects(?baGeom,?fGeom)) }
GROUP BY ?ba ?baGeom
```

The query above tests whether a burnt area intersects with a coniferous forest. If this is the case, groupings are made depending on the burnt area. The geometries of the forests corresponding to each burnt area are unioned, and their intersection with the burnt area is calculated and returned to the user. Note that only `strdf:union` is an aggregate function in the `SELECT` clause; `strdf:intersection` performs a computation involving the result of the aggregation and the value of `?baGeom` which is one of the variables determining the grouping according to which the aggregate computation is performed.

The result of the query evaluated on the data sets of Example 1 and 2 is displayed below:

?ba	?burnt
geonames:264637	"POLYGON ((20 21, 20 22, 22 22, 22 21, 21.5 21, 21.5 20, 21 20, 21 21, 20 21))"^^strdf:WKT
geonames:251283	"MULTIPOLYGON (((23.5 18.5, 23 18, 23 18.5, 23.5 18.5)), ((23.5 19, 24 19, 23.5 18.5, 23.5 19)))"^^strdf:WKT

6.3 GeoSPARQL

GeoSPARQL is a recently proposed OGC standard for a SPARQL-based query language for geospatial data expressed in RDF [5]. GeoSPARQL defines much of what is required for such a query language by providing vocabulary (classes, properties, and functions) that can be used in RDF graphs and SPARQL queries to represent and query geospatial data. GeoSPARQL follows the modular design typical of OGC standards and consists of the following components:

- *Core.* This component defines top level classes that provide users with vocabulary for modeling geospatial information. The classes offered by this component are `geo:SpatialObject` and `geo:Feature`. `geo:SpatialObject` is the top class defined by GeoSPARQL and has as instances everything that can have a spatial representation. `geo:Feature` is the class of all the features and is the superclass of all classes of features users might want to define. `geo:SpatialObject` is a superclass of `geo:Feature` and `geo:Geometry` (to be defined in the geometry extension component below).
- *Geometry extension.* This component provides vocabulary for asserting and querying information about geometries. A crucial design decision of GeoSPARQL is to use *literal values* to encode geometries as a single unit (similar to stSPARQL), and introduce two RDF datatypes `geo:wktLiteral` and `geo:gmlLiteral` for these literals. The extension is parameterized by the serialization standard of OGC to be used for encoding geometry literals (WKT or GML) and the version of the relevant standard. Literals of type `geo:wktLiteral` consist of an optional URI identifying the co-ordinate reference system followed by the WKT encoding of a geometry. Similarly, literals of type `geo:gmlLiteral` consist of a valid GML element that implements a subtype of type `GM.Object`. The following are three examples of such literals:

```
"POINT(-83.38 33.95)"^^geo:wktLiteral
```

```
"<http://www.opengis.net/def/crs/EPSSG/0/4326>  
POINT(33.95 -83.38)"^^geo:wktLiteral
```

```
"<gml:Point  
  srsName=\"http://www.opengis.net/def/crs/OGC/1.3/CRS84\"  
  xmlns:gml=\"http://www.opengis.net/gml\">  
  <gml:pos>-83.38 33.95</gml:pos>  
</gml:Point>"^^geo:gmlLiteral
```

The geometry extension defines the class `geo:Geometry` as the superclass of all geometry classes. It also defines properties for representing metadata of geometries (e.g., `geo:dimension` that captures the topological dimension), for associating features with geometries (e.g., `geo:hasGeometry`) and for associating geometries with their literal serializations (`geo:asWKT` and `geo:asGML`). In addition, this extension defines functions for performing non-topological operations on geometries (e.g., `geof:distance`, `geof:buffer`,

`geof:convexHull`, etc.). These functions are the same as the ones defined for simple features defined in the ISO 19125 specification presented earlier in Section 3 [4].

- *Topology vocabulary extension.* This component provides vocabulary for asserting and querying topological relations between spatial objects. The extension is parameterized⁴³ by the family of topological relations supported. These can be the topological relations for simple features defined in the ISO 19125 specification presented in Section 3 [4] (e.g., `geo:sfTouches`), the Egenhofer relations defined in [20] (e.g., `geo:ehMeet`), or the RCC-8 relations defined in [61] (e.g., `geo:rcc8ec`). An important point here is that these topological relations can relate not just geometries but also features. They can be asserted by a triple of an RDF graph [51,67] (e.g., `dbpedia:Athens geo:sfWithin dbpedia:Greece`) but also used in a triple pattern of a SPARQL query (e.g., `?x geo:sfWithin dbpedia:Greece`). The query rewriting extension discussed below provides a mechanism to derive topological relations between features from the relationships that are satisfied by their corresponding geometries. In this way GeoSPARQL caters to users that are more interested in qualitative spatial reasoning applications [63,15] but also to more traditional GIS or DBMS users interested in geospatial computations.
- *Geometry topology extension.* This component provides a collection of Boolean functions that operate on geometries and check whether given topological relationships hold for these geometries. The extension is parameterized by the family of topological relations used, the serialization standard for the geometries and the version of the serialization standard. The extension defines Boolean functions that correspond to each of the topological relations of the topology vocabulary extension presented above (simple features, Egenhofer, and RCC-8) and can be applied to geometry literals encoded in WKT or GML (e.g., the Boolean function `geof:ehMeet` that corresponds to the Egenhofer topological relation `geo:ehMeet` mentioned above). In addition, the extension defines the general function `geof:relate` that can be used to specify any topological relationship one might be interested in that can be expressed using the dimensionally extended 9-intersection model of [20].
- *RDFS entailment extension.* This component provides a mechanism for realizing the RDFS entailments that follow from the geometry class hierarchies that are defined by the WKT and GML standards used as serializations of geometry literals, and the properties introduced by GeoSPARQL. If this extension is supported by a system then the system should use an implementation of RDFS entailment to allow the derivation of new triples from those already in a graph. An example is the triple `ex:f1 rdf:type geo:Feature` that follows from the triples `ex:f1 geo:hasGeometry ex:g1` and `geo:hasGeometry`

⁴³ The topology extension (and other extensions discussed later) are parameterized so that implementations have a choice from the multiple possibilities allowed by the extension.

`rdfs:domain geo:Feature`. This extension is parameterized by the family of topological relations used, the serialization standard for the geometries and the version of the serialization standard.

- *Query rewrite extension*. This component provides a collection of RIF rules⁴⁴ that derive any of the topological relations of the topology vocabulary extension between two pairs of spatial objects (features or geometries), whenever the corresponding Boolean function of the geometry topology extension holds between the corresponding geometry literals. As an example, the component has a RIF rule named `geor:sfWithin` that can be used to derive the triple `dbpedia:Athens geo:sfWithin dbpedia:Greece` from the fact that the Boolean function `geof:sfWithin` returns true when evaluated with arguments the WKT encodings of the geometry literals corresponding to features `dbpedia:Athens` and `dbpedia:Greece`. The extension is parameterized by the family of topological relations used, the serialization standard for the geometries and the version of the serialization standard. The name of the extension (query rewrite) comes from one of its possible implementations which would be to re-write a given SPARQL query with a triple pattern involving, e.g., a topological relation between two features, into one that checks whether the corresponding Boolean function holds between the geometry literals corresponding to the features.

The above six components also define conformance classes which can be followed by implementations that wish to support only a subset of GeoSPARQL.

Let us now revisit Examples 1, 2 and 3 above, but this time follow the modelling and querying approach of GeoSPARQL.

Example 5. Triples representing the town of Olympia and a burnt area product produced by the processing chain of the National Observatory of Athens.

```
noa:Town rdfs:subClassOf geo:Feature.

geonames:264637 a noa:Town;
    geonames:name "Olympia";
    owl:sameAs dbpedia:Olympia_Greece;
    geo:hasGeometry ex:OlympiaPolygon.

ex:OlympiaPolygon a sf:Polygon;
    geo:asWKT "POLYGON((21.5 18.5, 23.5 18.5,
    23.5 21, 21.5 21, 21.5 18.5))"^^geo:wktLiteral.

noa:BurntArea rdfs:subClassOf geo:Feature.

noa:ba1 a noa:BurntArea;
    owl:sameAs dbpedia:Olympia_Greece;
    geo:hasGeometry ex:ba1Polygon.
```

⁴⁴ <http://www.w3.org/TR/rif-overview/>

```

ex:ba1Polygon a sf:Polygon;
               geo:asWKT "POLYGON((20 20, 20 22, 22 22, 22 20,
               20 20))"^^geo:wktLiteral.

```

The above triples are a subset of the ones previously mentioned in Examples 1 and 2, slightly modified to use the vocabularies of GeoSPARQL.

We will now give an example of a GeoSPARQL query.

Example 6. Return the names of towns that have been affected by fires.

```

SELECT ?name
WHERE { ?town a noa:Town;
         geonames:name ?name;
         geo:hasGeometry ?townpoly.
        ?townpoly geo:asWKT ?townGeo.
        ?ba a noa:BurntArea;
         geo:hasGeometry ?bapoly.
        ?bapoly geo:asWKT ?baGeo.
        FILTER(geof:sfIntersects(?townGeo,?baGeo))
}

```

The above query is the same query with the one of Example 3, but now it is expressed in GeoSPARQL using the geometry topology extension of GeoSPARQL. This query checks whether the Boolean function `geof:sfIntersects` holds between the geometry literals related to towns and burned areas.

Let us now illustrate the modeling possibilities offered by the topology extension of GeoSPARQL with an example.

Example 7. Triples representing administrative geography information about the community of Olympia, the municipality of Olympia and the region of West Greece.

```

gag:Olympia rdf:type gag:Community;
             rdfs:label "Ancient Olympia".

gag:OlympiaMunicipality rdf:type gag:Municipality;
                       rdfs:label "Municipality of Ancient Olympia".

gag:WestGreece rdf:type gag:Region;
               rdfs:label "Region of West Greece".

gag:OlympiaMunicipality geo:sfContains gag:Olympia .

gag:WestGreece geo:sfContains gag:OlympiaMunicipality .

```

According to the Greek Administrative Geography⁴⁵ (namespace `gag` in the example), Greece is divided into 7 decentralized administrations, 13 regions and 325 municipalities. Communities, such as Ancient Olympia, are parts of municipalities (in this case the municipality with the same name). The municipality of Ancient Olympia is part of the region of West Greece. The last two of the above triples are used to assert the topological relations that hold between these three administrative divisions of Greece using vocabulary from the topology extension of GeoSPARQL.

Let us now consider the following query.

Example 8. Find the administrative region that contains the community of Ancient Olympia.

```
SELECT ?d
WHERE {
    ?d rdf:type gag:Region.
    ?d geo:sfContains geonames:Olympia.
}
```

The answer to the previous query is displayed below:

?d
gag:WestGreece

GeoSPARQL does not tell us how to compute this answer which needs reasoning about the transitivity of relation `geo:sfContains`. One option would be to have rules that express the transitivity of RCC-8 relations [52] while another would be to consult an RCC-8 reasoner based on constraint networks as in description logic reasoners PelletSpatial [78] and RacerPro [71].

6.4 Comparing stSPARQL with GeoSPARQL

Let us now compare the two languages stSPARQL and GeoSPARQL presented earlier. Both languages share the same basic assumption that a literal of an appropriately defined datatype should be used to encode all information about a geometry. The datatypes used in the two languages are exactly the same (although they have slightly different names), and enable serializations of geometries according to the OGC standards WKT and GML. The same choice has been made by the implementation of [12,49,10] and the systems Virtuoso and uSeekM to be discussed in Section 7. Note that this is not the only choice the literature offers us regarding the representation of a geometry (see also the discussion in Section 6.5). Others have argued for using a set of triples and appropriate vocabulary (e.g., from GeoRSS) to represent a geometry, such as [38,57] and the systems AllegroGraph and OWLIM.

With the exception of the relevant datatypes, stSPARQL offers nothing else in terms of vocabulary for modeling geospatial information; it is left to the users

⁴⁵ http://en.wikipedia.org/wiki/Kallikratis_reform

to define their own classes and properties. On the contrary, GeoSPARQL offers some basic classes and properties that encode well-known GIS and OGC terminology and all the relevant inferences through its RDFS entailment extension. This is a positive thing for users already familiar with this terminology, and an invitation for others to become familiar with it and use it in their modeling. Naturally, these GeoSPARQL classes and properties can easily be adopted by stSPARQL. In the future, one can imagine that user communities will develop more specialized ontologies that extend the basic ontologies of GeoSPARQL with relevant geospatial concepts from their own application domain (e.g., the NeoGeo vocabulary in Section 6.5 below).

When it comes to functions for manipulating geometries, stSPARQL and GeoSPARQL have almost the same expressive power since they choose to support the same set of functions i.e., the functions of the geometry and geometry topology extension of GeoSPARQL (again, different names are being used but this is of no consequence). stSPARQL also offers aggregate functions that have not been discussed by GeoSPARQL but can be easily included in its geometry extension.

GeoSPARQL goes beyond stSPARQL with the functionality offered by its topological extension and the related query rewrite extension. This opens up the possibility of useful forms of topological reasoning not covered by stSPARQL. In our group similar issues have been discussed in the more general area of “incomplete information in RDF” of which a preliminary overview with emphasis on geospatial data is given in [42].

6.5 Other recent efforts in defining ontologies and vocabularies for geospatial data

In addition to the published approaches discussed above (Sections 6.1 to 6.3), there have been other recent efforts to define vocabularies for the modeling of geospatial information in RDF that have not yet been reported in published papers. One such effort that we would like to mention has been organized around VoCamps⁴⁶ concentrating on vocabularies for geospatial information, and has resulted in the proposal of the NeoGeo vocabulary⁴⁷. Similarly to GeoSPARQL, the NeoGeo ontology distinguishes between features and geometries. The class `geom:Geometry` is the superclass of all geometry classes and has subclasses `geom:Point`, `geom:LineString` etc. Similarly, there is a `geom:GeometryCollection` class with subclasses `geom:MultiPoint`, `geom:MultiLineString` etc. Thus, the idea is to make available in RDF the class hierarchy of the OGC Simple Feature Access Standard presented in Figure 2. The property `geom:geometry` is available for connecting a feature to its geometry. In addition, NeoGeo provides the vocabulary to assert topological relations between features using RCC-8 relations. These relations are mapped to RDF properties (e.g., `spatial:NTPP`) and they are used to connect instances of

⁴⁶ http://vocamp.org/wiki/Main_Page

⁴⁷ <http://geovocab.org/doc/neogeo/>

the class `spatial:Feature`. Unlike stSPARQL and GeoSPARQL, which model geometries as a single literal, the NeoGeo effort opens up the possibility to have alternative representations of a geometry explicitly represented in RDF (e.g., for a line string, we also give a list of its points using appropriate vocabulary). This is an interesting idea since applications might need a more fine-grained access to the components of a geometry than the one offered by stSPARQL and GeoSPARQL through literals. To the best of our knowledge, NeoGeo is an effort still in progress and has not offered complete solutions to these interesting issues yet.

7 Geospatial Data Modeling and Querying in Existing RDF Stores

In parallel with the development of RDF/SPARQL-based data models and query languages discussed above, existing RDF stores started to add support for geospatial data. In addition, new systems were built that were aimed explicitly at the implementation of the new modeling and querying proposals for geospatial data. In this section we present a survey of these systems concentrating only on the data model and query language they support for geospatial data. Details having to do with system architecture, optimization techniques, indexing structures, etc. are omitted in most cases.

The system Strabon⁴⁸, under development in our group since 2009, is a storage and query evaluation module for stRDF/stSPARQL [49]. Strabon extends the well-known RDF store Sesame, allowing it to manage both thematic and spatial data expressed in stRDF and stored in the PostGIS spatially enabled DBMS. The current version of Strabon (Strabon 3.0) fully implements stSPARQL. In addition, Strabon 3.0 implements fully the GeoSPARQL Core, Geometry extension and Geometry topology extension components. The implementation of this subset of GeoSPARQL in Strabon was straightforward given the close relationship between stSPARQL and GeoSPARQL.

The RDF store Parliament⁴⁹ developed by Rob Battle and Dave Kolas at Raytheon BBN Technologies fully implements most of the functionality of GeoSPARQL except the query rewriting extension which has been promised for a forthcoming version [10].

AllegroGraph⁵⁰ is one of the first RDF stores that provided support for geospatial data. It can only store geometries that are points in a two-dimensional space. Support is provided both for Cartesian coordinate systems and for spherical coordinate systems. Geometries are assigned to geometric objects through the use of property `<http://franz.com/ns/allegrograph/3.0/geospatial/pos>`. For querying, AllegroGraph introduces a GEO operator for expressing geospatial query

⁴⁸ <http://www.strabon.di.uoa.gr/>

⁴⁹ <http://parliament.semwebcentral.org/>

⁵⁰ <http://www.franz.com/agraph/allegrograph/>

patterns in SPARQL. GEO has a syntax similar to the GRAPH operator of standard SPARQL. Whenever a geospatial query is posed in AllegroGraph, the bindings of the traditional WHERE clause of the query are evaluated at a first step. These bindings are then shared with the geospatial part of the query defined by the GEO clause and are used to define an area with the use of either a buffer (a variation exists for spherical and Cartesian coordinate systems respectively) or a bounding box operator. The area defined is used to filter the initial bindings, returning the ones lying within it. Results can also be filtered by utilizing AllegroGraph's functions that compute the Cartesian or the Haversine distance of two points, as well as functions returning a point's coordinates in order to perform numeric comparisons between the coordinates of different points. This syntax is neither as flexible as the syntax of GeoSPARQL and stSPARQL, nor SPARQL compliant.

OWLIM⁵¹ is another semantic repository enhanced with geospatial capabilities. It allows the representation of geometries that are points represented using the W3C Basic Geo vocabulary. OWLIM introduces a series of property functions⁵² extending SPARQL to enable the expression of queries over WGS84 point geometries. These functions are restricted to point-within-polygon and buffer operations. A SPARQL filter function computing the distance between two points is also available.

OpenLink Virtuoso⁵³, like OWLIM, provides support for the representation and querying of two-dimensional point geometries. However, Virtuoso allows geometries to be expressed in other coordinate reference systems besides WGS84. Virtuoso models geometries by typed literals like stSPARQL and GeoSPARQL. The datatype `<http://www.openlinksw.com/schemas/virtrdf#Geometry>` has been introduced for this purpose. The value of such a literal is the WKT serialization of the spatial object's geometry. Virtuoso offers vocabulary for a subset of the ISO 13249 SQL/MM standard to perform geospatial queries using SPARQL. The SQL/MM functions supported are realized by this vocabulary which can be used in the SELECT and FILTER clause of a SPARQL query. In the case of FILTER for example, the user can test for relations between two geometries by using SPARQL functions corresponding to the `st_intersects`, `st_contains`, and `st_within` SQL/MM functions. Thus, the query language of Virtuoso makes similar design decisions with stSPARQL and corresponds only to a part of the functionality envisioned by GeoSPARQL.

The well-known Oracle DBMS also offers support for representing and querying geospatial data in RDF through its Semantic Technologies product suite (version 11g, Release 2)⁵⁴. Similarly to stSPARQL and GeoSPARQL, geometries are modelled using typed literals utilizing the datatype `http://xmlns.oracle.com/rdf/geo/WKTLiteral`. The value of such a literal begins with an optional coordinate reference system URI, followed by white space and the WKT

⁵¹ <http://www.ontotext.com/owlim/>

⁵² http://www.w3.org/wiki/SPARQL/Extensions/Computed_Properties/

⁵³ <http://virtuoso.openlinksw.com/>

⁵⁴ <http://www.oracle.com/technetwork/database/options/semantic-tech/>

serialization of the geometry. In the default case, WGS84 is used as the coordinate reference system in which the geometry is represented, and the URI can be omitted. Oracle 11g defines a series of SPARQL extension functions that can be used with its SQL-based RDF querying functionalities exposing a subset of the functionality of the OpenGIS Simple Feature Access Standard specification. Other SPARQL extension functions such as *nearestNeighbor* and *centroid* are offered as well.

OpenSahara uSeekM⁵⁵ is an add-on library for semantic repositories utilizing the Sesame Java interface. uSeekM initially focused on providing full text search functionality. Newer versions have been enhanced with geospatial capabilities as well. Specifically, uSeekM follows an approach very close to stSPARQL and GeoSPARQL, enabling the representation not only of point geometries like most previous systems, but also of complex geometries such as polygons and line strings, and allowing queries through the use of SPARQL extension functions that incorporate the OpenGIS Simple Feature Access standard functionality.

8 Summary

In this paper, we presented a survey of related work in data models and query languages for linked geospatial data. We believe that this area has now matured enough to allow interested researchers to concentrate on the other important topics we mentioned in Section 1 that are currently open (most importantly, on the implementation of scalable systems for linked geospatial data). From a query language design point of view, two interesting theoretical questions remain also open. How do we give formal semantics for stSPARQL and GeoSPARQL (in the spirit of the W3C SPARQL semantics or the paper [56])? What is the computational complexity of query processing for these languages? We invite interested researchers to join us in pursuing these open problems.

Acknowledgements

This work was supported in part by projects TELEIOS (<http://www.earthobservatory.eu/>) and SemsorGrid4Env (<http://www.sensorgrid4env.eu/>) funded by the European Commission. We thank all the participants and the reviewers of these projects for many interesting discussions on the area covered by this survey paper.

References

1. Open Geospatial Consortium. Geography Markup Language(GML) Encoding Standard. OpenGIS Implementation Standard (2007), available from: http://portal.opengeospatial.org/files/?artifact_id=20509

⁵⁵ <https://dev.opensahara.com/projects/useekm/>

2. Open Geospatial Consortium. Geography Markup Language (GML) simple features profile. OpenGIS Implementation Standard Profile (2010), available from: http://portal.opengeospatial.org/files/?artifact_id=39853
3. Open Geospatial Consortium. OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option. OpenGIS Implementation Standard (2010), available from: http://portal.opengeospatial.org/files/?artifact_id=25354
4. Open Geospatial Consortium. OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common Architecture. OpenGIS Implementation Standard (2010), available from: http://portal.opengeospatial.org/files/?artifact_id=25355
5. Open Geospatial Consortium. OGC GeoSPARQL - A geographic query language for RDF data. OGC Candidate Implementation Standard (2012)
6. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(11), 832–843 (1983)
7. Arpinar, B.I., Sheth, A., Ramakrishnan, C., Usery, Azami, M., Kwan, M.P.: Geospatial Ontology Development and Semantic Analytics. *Transactions in GIS* 10(4), 551–575 (2006)
8. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData: Adding a Spatial Dimension to the Web of Data. In: *International Semantic Web Conference. Lecture Notes in Computer Science*, vol. 5823, pp. 731–746. Springer (2009)
9. Balbiani, P., Condotta, J.F., del Cerro, L.F.: A new tractable subclass of the rectangle algebra. In: *IJCAI*. pp. 442–447 (1999)
10. Battle, R., Kolas, D.: Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL (2011), under review in “Semantic Web Interoperability, Usability, Applicability” journal, available from <http://www.semantic-web-journal.net/content/enabling-geospatial-semantic-web-parliament-and-geosparql>.
11. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. *International Journal on Semantic Web and Information Systems* 5(3), 1–22 (2009)
12. Brodt, A., Nicklas, D., Mitschang, B.: Deep integration of spatial query processing into native RDF triple stores. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2010)
13. Clementini, E., Felice, P.D., Oosterom, P.v.: A small set of formal topological relationships suitable for end-user interaction. In: *Proceedings of the Third International Symposium on Advances in Spatial Databases*. pp. 277–295. SSD '93, Springer-Verlag, London, UK, UK (1993)
14. Clementini, E., Sharma, J., Egenhofer, M.J.: Modelling topological spatial relations: Strategies for query processing. *Computers & Graphics* 18(6), 815 – 822 (1994)
15. Cohn, A.G.: Qualitative Spatial Representation and Reasoning Techniques. In: *KI. Lecture Notes in Computer Science*, vol. 1303, pp. 1–30. Springer (1997)
16. Cohn, A.G., Bennett, B., Gooday, J., Gotts, N.M.: Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *Geoinformatica* 1(3), 275–316 (1997)
17. Egenhofer, M.J.: A Formal Definition of Binary Topological Relationships. In: *FODO. Lecture Notes in Computer Science*, vol. 367, pp. 457–472. Springer (1989)
18. Egenhofer, M.J.: Spatial SQL: A Query and Presentation Language. *IEEE Trans. Knowl. Data Eng.* 6(1), 86–95 (1994)
19. Egenhofer, M.J.: Toward the semantic geospatial web. In: *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*. pp. 1–4. GIS '02, ACM, New York, NY, USA (2002)

20. Egenhofer, M.J., Franzosa, R.D.: Point-Set Topological Spatial Relations. In: Proceedings of International Journal of Geographical Information Systems. vol. 5(2), pp. 161–174 (1991)
21. Geerts, F.: Constraint query languages. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 454–458. Springer US (2009)
22. Goodwin, J., Dolbear, C., Hart, G.: Geographical Linked Data: The Administrative Geography of Great Britain on the Semantic Web. *Transaction in GIS* 12(1), 19–30 (2009)
23. Gray, A.J.G., Garcia-Castro, R., Kyzirakos, K., Karpathiotakis, M., Calbimonte, J.P., Page, K.R., Sadler, J., Frazer, A., Galpin, I., Fernandes, A.A.A., Paton, N.W., Corcho, Ó., Koubarakis, M., Roure, D.D., Martinez, K., Gómez-Pérez, A.: A Semantically Enabled Service Architecture for Mashups over Streaming and Stored Data. In: ESWC. Lecture Notes in Computer Science, vol. 6644, pp. 300–314. Springer (2011)
24. Grumbach, S., Su, J.: Queries with arithmetical constraints. *Theor. Comput. Sci.* 173(1), 151–181 (1997)
25. Gutierrez, C., Hurtado, C., Vaisman, A.: Introducing Time into RDF. *IEEE Transactions on Knowledge and Data Engineering* 19(2), 207–218 (2007)
26. Guting, R.H.: An introduction to spatial database systems. *VLDB Journal* 3, 357–399 (1994)
27. Guting, R.H., Schneider, M.: Realms: A foundation for spatial data types in database systems. In: 3rd International Symposium on Advances in Spatial Databases. pp. 14–35. Springer-Verlag (1993)
28. Guting, R.H., Schneider, M.: Moving Objects Databases. Morgan Kaufmann Publishers Inc. (2005)
29. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: International Conference on Management of Data. pp. 47–57. ACM (1984)
30. Hakimpour, F., Aleman-meza, B., Perry, M., Sheth, A.: Data Processing in Space, Time and Semantics Dimensions. In: Terra Cognita 2006 - Directions to the Geospatial Semantic Web, in conjunction with the Fifth International Semantic Web Conference (2006)
31. Hausenblas, M., Halb, W., Raimond, Y., Feigenbaum, L., Ayers, D.: Scovo: Using statistics on the web of data. In: ESWC. Lecture Notes in Computer Science, vol. 5554, pp. 708–722. Springer (2009)
32. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. Research report, Max-Planck-Institut für Informatik (2010)
33. Ivanova, M., Kersten, M., Manegold, S.: Data vaults: a symbiosis between database technology and scientific file repositories. In: 24th International Conference on Scientific and Statistical Database Management (2012)
34. Kanellakis, P., Kuper, G., Revesz, P.: Constraint Query Languages. In: Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. pp. 299–313 (1990)
35. Kersten, M.L., Zhang, Y., Ivanova, M.: SciQL, a query language for science applications. In: Proceedings of EDBT-ICDT, Workshop on Array Databases (2011)
36. Kolas, D.: Supporting Spatial Semantics with SPARQL. In: Terra Cognita Workshop. Karlsruhe, Germany (2008)
37. Kolas, D., Hebel, J., Dean, M.: Geospatial Semantic Web: Architecture of Ontologies. In: GeoSpatial Semantics, Lecture Notes in Computer Science, vol. 3799, pp. 183–194. Springer (2005)

38. Kolas, D., Self, T.: Spatially Augmented Knowledgebase. In: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007). Lecture Notes in Computer Science, vol. 4825, pp. 785–794. Springer Verlag (2007)
39. Koubarakis, M.: Database models for infinite and indefinite temporal information. *Information Systems* 19, 141–173 (1994)
40. Koubarakis, M., Kyzirakos, K.: Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL. In: ESWC. Lecture Notes in Computer Science, vol. 6088, pp. 425–439. Springer (2010)
41. Koubarakis, M., Kyzirakos, K., Karpathiotakis, M., Nikolaou, C., Vassos, S., Garbis, G., Sioutis, M., Bereta, K., Michail, D., Kontoes, C., Papoutsis, I., Herekakis, T., Manegold, S., Kersten, M., Ivanova, M., Pirk, H., Zhang, Y., Datcu, M., Schwarz, G., Dumitru, O., Molina, D., Molch, K., Giammatteo, U.D., Sagona, M., Perelli, S., Reitz, T., Klien, E., Gregor, R.: Building Earth Observatories using Scientific Database and Semantic Web Technologies (2012), unpublished manuscript.
42. Koubarakis, M., Kyzirakos, K., Karpathiotakis, M., Nikolaou, C., Sioutis, M., Vassos, S., Michail, D., Herekakis, T., Kontoes, C., Papoutsis, I.: Challenges for Qualitative Spatial Reasoning in Linked Geospatial Data. In: Proceedings of IJCAI 2011 Workshop on Benchmarks and Applications of Spatial Reasoning (2011)
43. Koubarakis, M., Kyzirakos, K., Nikolaou, B., Sioutis, M., Vassos, S.: A data model and query language for an extension of rdf with time and space. Deliverable D2.1, TELEIOS (2011), available from: http://www.earthobservatory.eu/deliverables/FP7-257662-TELEIOS-D2.1_revised.pdf
44. Koubarakis, M., Nikolaou, C., Fisikopoulos, V.: Theoretical results on query processing for RDF/SPARQL with time and space. Deliverable D2.3, European FP7 project TELEIOS (2011), available from: <http://www.earthobservatory.eu/deliverables/FP7-257662-TELEIOS-D2.3.pdf>
45. Koubarakis, M., Sellis, T.K., Frank, A.U., Grumbach, S., Güting, R.H., Jensen, C.S., Lorentzos, N.A., Manolopoulos, Y., Nardelli, E., Pernici, B., Schek, H.J., Scholl, M., Theodoulidis, B., Tryfona, N. (eds.): Spatio-Temporal Databases: The CHOROCHRONOS Approach, Lecture Notes in Computer Science, vol. 2520. Springer (2003)
46. Kuijpers, B.: Linear versus polynomial constraint databases. In: Encyclopedia of GIS, pp. 612–615 (2008)
47. Kuper, G., Ramaswamy, S., Shim, K., Su, J.: A Constraint-based Spatial Extension to SQL. In: Proceedings of the 6th International Symposium on Advances in Geographic Information Systems (1998)
48. Kuper, G.M., Libkin, L., Paredaens, J. (eds.): Constraint Databases. Springer (2000)
49. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Developing Registries for the Semantic Sensor Web using stRDF and stSPARQL (short paper). In: Proceedings of the 3rd International Workshop on Semantic Sensor Networks (SSN10) . vol. 668 (2010)
50. León, A.d., Saquicela, V., Vilches, L.M., Villazón-Terrazas, B., Priyatna, F., Corcho, O.: Geographical Linked Data: a Spanish Use Case. In: I-SEMANTICS. ACM (2010)
51. Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W.: Geographic Information Systems and Science. John Wiley & Sons (2005)
52. Lopez, X.: Querying Spatial Data with SPARQL. ORACLE presentation sent to NKUA group by email on June 18, 2010

53. Manola, F., Orenstein, J.A.: Toward a General Spatial Data Model for an Object-Oriented DBMS. In: VLDB. pp. 328–335 (1986)
54. Melton, J., Eisenberg, A.: SQL Multimedia and Application Packages (SQL/MM). SIGMOD Record 30(4), 97–102 (2001)
55. Oro, E., Ruffolo, M., Staab, S.: SXPath: extending XPath towards spatial querying on web documents. Proceedings VLDB Endowment 4, 129–140 (2010)
56. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. In: Proceedings of the 5th International Semantic Web Conference. pp. 30–43 (2006)
57. Perry, M.: A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data. Ph.D. thesis, Wright State University (2008)
58. Perry, M., Hakimpour, F., Sheth, A.P.: Analyzing Theme, Space, and Time: an Ontology-based Approach. In: Proceedings of the 14th annual ACM International Symposium on Advances in Geographic Information Systems (GIS'06). pp. 147–154 (2006)
59. Perry, M., Jain, P., Sheth, A.P.: SPARQL-ST: Extending SPARQL to Support Spatiotemporal Queries. In: Geospatial Semantics and the Semantic Web. Semantic Web And Beyond Computing for Human Experience, vol. 12, pp. 61–86. Springer (2011)
60. Perry, M., Sheth, A.P., Hakimpour, F., Jain, P.: Supporting Complex Thematic, Spatial and Temporal Queries over Semantic Web Data. In: Proceedings of the Second International Conference on GeoSpatial Semantics (GeoS). pp. 228–246 (2007)
61. Randell, D.A., Cui, Z., Cohn, A.G.: A Spatial Logic based on Regions and Connection. In: Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning (1992)
62. Renz, J.: Qualitative spatial reasoning with topological information. Springer (2002)
63. Renz, J., Nebel, B.: Efficient methods for qualitative spatial reasoning. Journal of Artificial Intelligence Research (JAIR) 15, 289–318 (2001)
64. Revesz, P.Z.: Introduction to Constraint Databases. Springer (2002)
65. Rigaux, P., Scholl, M., Voisard, A.: Introduction to Spatial Databases: Applications to GIS. Morgan Kaufmann, Reading, MA (2000)
66. Rigaux, P., Scholl, M., Segoufin, L., Grumbach, S.: Building a constraint-based spatial database system: model, languages, and implementation. Information Systems 28(6), 563–595 (2003)
67. Rigaux, P., Scholl, M., Voisard, A.: Spatial databases - with applications to GIS. Elsevier (2002)
68. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers Inc. (2006)
69. Scholl, M., Voisard, A.: Object-Oriented Database Systems for Geographic Applications: an Experiment with O_2 . pp. 103–137. Morgan Kaufmann (1992)
70. Sheth, A., Perry, M.: Traveling the Semantic Web through Space, Time, and Theme. IEEE Internet Computing 12(2), 81–86 (2008)
71. Stocker, M., Sirin, E.: PelletSpatial: A hybrid RCC-8 and RDF/OWL reasoning and query engine. In: OWLED (2009)
72. Stolze, K.: SQL/MM Spatial - The Standard to Manage Spatial Data in a Relational Database System. In: BTW. pp. 247–264 (2003)
73. Suarez-Figueroa, M.C., Gomez-Perez, A.: NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology. In: Proceedings of the International Conference on Software, Services & Semantic Technologies (2009)

74. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: 16th international World Wide Web conference (WWW 2007). ACM Press (2007)
75. Vandeurzen, L., Gyssens, M., Gucht, D.V.: On the expressiveness of linear-constraint query languages for spatial databases. *Theoretical Computer Science* (2001)
76. Vilches-Blázquez, L.M., Ramos, J.A., López-Pellicer, F.J., Corcho, O., Nogueras-Iso, J.: An approach to comparing different ontologies in the context of hydrographical information. In: *Information fusion and geographic information systems* (2009)
77. Vilches-Blázquez, L.M., Villazón-Terrazas, B., Saquicela, V., de León, A., Corcho, O., Gómez-Pérez, A.: GeoLinked data and INSPIRE through an application case. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. pp. 446–449. GIS '10, ACM (2010)
78. Wessel, M., Moller, R.: Flexible software architectures for ontology-based information systems. *Journal of Applied Logic* 7(1), 75–99 (2009)