

Article

# An Ontology-Based Framework for Geospatial Integration and Querying of Raster Data Cube Using Virtual Knowledge Graphs

Younes Hamdani <sup>1,\*</sup>, Guohui Xiao <sup>2,3,4</sup>, Linfang Ding <sup>5</sup> and Diego Calvanese <sup>1,4,6</sup>

- <sup>1</sup> Department of Computing Science, Umeå University, 901 87 Umeå, Sweden; diego.calvanese@umu.se  
<sup>2</sup> Department of Information Science and Media Studies, University of Bergen, 5007 Bergen, Norway; guohui.xiao@uib.no  
<sup>3</sup> Department of Informatics, University of Oslo, 0373 Oslo, Norway  
<sup>4</sup> Ontopic S.r.l., 39100 Bolzano, Italy  
<sup>5</sup> Department of Civil and Environmental Engineering, Norwegian University of Science and Technology, 7491 Trondheim, Norway; linfang.ding@ntnu.no  
<sup>6</sup> KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, 39100 Bolzano, Italy  
\* Correspondence: younes.hamdani8@gmail.com

**Abstract:** The integration of the raster data cube alongside another form of geospatial data (e.g., vector data) raises considerable challenges when it comes to managing and representing it using knowledge graphs. Such integration can play an invaluable role in handling the heterogeneity of geospatial data and linking the raster data cube to semantic technology standards. Many recent approaches have been attempted to address this issue, but they often lack robust formal elaboration or solely concentrate on integrating raster data cubes without considering the inclusion of semantic spatial entities along with their spatial relationships. This may constitute a major shortcoming when it comes to performing advanced geospatial queries and semantically enriching geospatial models. In this paper, we propose a framework that can enable such semantic integration and advanced querying of raster data cubes based on the virtual knowledge graph (VKG) paradigm. This framework defines a semantic representation model for raster data cubes that extends the GeoSPARQL ontology. With such a model, we can combine the semantics of raster data cubes with features-based models that involve geometries as well as spatial and topological relationships. This could allow us to formulate spatiotemporal queries using SPARQL in a natural way by using ontological concepts at an appropriate level of abstraction. We propose an implementation of the proposed framework based on a VKG system architecture. In addition, we perform an experimental evaluation to compare our framework with other existing systems in terms of performance and scalability. Finally, we show the potential and the limitations of our implementation and we discuss several possible future works.

**Keywords:** raster data cube; ontology; virtual knowledge graphs; geospatial data integration; knowledge querying; SPARQL



**Citation:** Hamdani, Y.; Xiao, G.; Ding, L.; Calvanese, D. An Ontology-Based Framework for Geospatial Integration and Querying of Raster Data Cube Using Virtual Knowledge Graphs. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 375. <https://doi.org/10.3390/ijgi12090375>

Academic Editors: Wolfgang Kainz and Sisi Zlatanova

Received: 23 June 2023

Revised: 16 August 2023

Accepted: 3 September 2023

Published: 8 September 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Recent advances in geospatial data acquisition, particularly in the areas of remote sensing as well as positioning and communication technologies, have led to the accumulation of an enormous amount of geospatial data with temporal properties. One widely used spatiotemporal data structure is the raster data cube, which describes the continuous distribution of a variable in space and its change over time. It can be seen as a special case of a multi-dimensional array with three dimensions: two spatial and one temporal [1,2]. Such a form of spatiotemporal data (i.e., a data cube) raises considerable challenges when it comes to managing and portraying it using a knowledge graph. The smart integration of data cubes using knowledge graphs can play an invaluable role in handling the heterogeneity of geospatial data and linking them with other enhanced semantics and knowledge.

Knowledge graphs manage data and knowledge in the form of a semantic directed graph that allows the capture of relationships between instances of different concepts and offers a quite flexible structure compared to the relational structure [3]. In addition, its reasoning capabilities can facilitate the inference of more implicit facts and knowledge, and most importantly, it can verify the consistency of the data/knowledge as a whole, which can contribute to the intelligent integration of different data sources and forms.

Many thoughtful and recent approaches have been tried to address the representation of raster data cubes using knowledge graphs [4–6]. However, these solutions are not yet mature. They either lack strong formal development or focus only on the management of this form (i.e., a raster data cube) without combining it with other data forms that represent discrete space with a set of spatial features along with their spatial relationships. This can present a major shortcoming when it comes to performing advanced geospatial queries and semantically enriching geospatial models. One feasible solution would be to translate the raster data cubes into a linked dataset (i.e., a set of RDF triples), manage them using a knowledge graph, and then combine them with other geospatial knowledge graphs to perform more advanced geospatial queries [7–9]. However, such a solution implies the materialization of the data into RDF triples. This directly leads to unnecessary high redundancy, and then high storage costs and scalability issues when the data size increases. One alternative approach to avoiding materialization is to use an ontology to define a semantic layer of the data cube and link it to the data source through mapping. Thereafter, and only during the query process, the raster knowledge graph (i.e., the RDF triples representing the raster data cubes) could be generated. This virtual knowledge graph (VKG) [10] seems to be promising to link different forms of geospatial data and manage spatiotemporal semantics at a high level of abstraction, which could significantly reduce the high redundancy that materialization would entail.

In this paper, we propose a framework that can enable such semantic integration and advanced querying of raster data cubes based on VKG. We define a semantic representation model for raster data cubes in this framework by extending the GeoSPARQL ontology. With such a model, we can combine the semantics of raster data cubes with a vector data model that represents features using geometries and attributes. We propose an implementation of our framework and confront it with a set of scenarios that take the form of basic and advanced semantic geospatial queries. With this implementation, one can formulate spatiotemporal queries using SPARQL in a natural way by using ontological concepts at an appropriate level of abstraction. We then perform an experimental evaluation to compare our framework with other existing systems in terms of performance and scalability and show the potential as well as the limitations of our implementation.

The rest of the paper is organized as follows. Section 2 explores the literature regarding the integration of raster data cubes and their link to knowledge graphs and summarizes the different approaches that address this issue. Section 3 explains our proposed framework at the conceptual level by proposing a semantic representation model as well as a taxonomy of semantic geospatial queries. Section 4 presents the implementation aspect of the proposed framework and how, concretely, the raster data cube could be linked virtually to the knowledge graphs with an appropriate architecture and data modeling process. Section 5 evaluates the semantic query capability of our implemented framework by proposing a use case with its datasets, and formulating basic and advanced geospatial queries using SPARQL. In addition, this section evaluates the query performance of our approach and makes a comparison with other systems. Finally, Section 6 concludes the results, highlighting the advantages and disadvantages of our framework, and then provides insights for future work.

## 2. Related Works

The smart integration of earth observation (EO) raster data cubes has been extensively investigated from multiple aspects. While many works have focused on providing an accurate formalization and algebra to properly represent and manipulate this concept,

considering it a special case of multidimensional arrays [1,2], other works have focused on effective processing and analyzing raster data cubes. These works started with the proposal of array database systems such as SciDB [11] or Rasdaman [12], up to the development of an appropriate advanced infrastructure capable of supporting such a task with high performance and appropriate scalability on a large amount of data [13–16].

Further solutions have been proposed to facilitate the efficient integration and querying of raster data cubes. One of the most important advancements in this area was the creation of the Web Coverage Processing Service (WCPS) [17]. This standard enhances and expands the basic access to geospatial raster coverage data by providing a web interface in the form of a retrieval language that can handle on-the-fly processing and filtering of multi-dimensional raster coverages. It also enables the execution of more complex queries. The Open Geospatial Consortium (OGC) has been responsible for maintaining this language, and it has been further improved by the Petascope implementation of WCPS [18]. This implementation enables the translation of queries from the aforementioned language into Rasdaman array DBMS queries. However, despite the existence of such powerful solutions, there has not been coupling between raster data cubes and conventional linked data technologies.

Several efforts were initiated to address such a coupling, starting first at the conceptual level by proposing a semantic model in the form of ontologies to capture the data cube properties. The main efforts culminated in the proposal of the RDF Data Cube (QB) ontology by the World Wide Web Consortium (W3C) in 2017 [19]. This ontology was originally conceived to handle multi-dimensional statistical data, and subsequently adopted for representing Earth Observation (EO) data that are conceived as a data cube with the three dimensions of latitude, longitude, and time. The main benefit of QB ontology emanates from the combination of several standard vocabularies, such as Simple Knowledge Organization System (SKOS), sensor network (SSN), OWL-time, and Provenance Ontology (PROV-O) [19]. An auxiliary ontology designated as the RDF Data Cube extensions for spatiotemporal components (QB4ST), was meticulously formulated by the W3C [20]. The objective was to enable a flexible adaptation of data cubes to various dimensions, with particular emphasis on spatiotemporal aspects, allowing the formation of support coverage that encompasses dimensions ranging from one to four [21]. Nevertheless, despite the importance of these conceptual efforts, they have not been able to address implementation considerations regarding the efficient querying of Earth Observation (EO) data using SPARQL. In contrast, other initiatives have focused more on this facet of implementation, resulting in two main solutions, namely SciSPARQL [4,22] and GeoSPARQL+ [23]. The goal of these solutions was to provide SPARQL support for scientific array data and raster coverages by defining new data types, algebras, and operators for raster arrays. While these solutions offered computational benefits, they only considered the static aspect of raster coverages and did not adequately handle the temporal dimension. Furthermore, users needed to familiarize themselves with a set of new raster data operators, in addition to the underlying SPARQL syntax, in order to perform geospatial queries.

Several approaches have been proposed to achieve the integration of raster data cubes, including their temporal dimension, while establishing connections with knowledge graph and semantic web technologies. These approaches can be categorized into three main types. The first approach focuses on feature-based representation, placing emphasis on the selection or derivation of ontological features through the aggregation of multiple raster cells. Subsequently, instead of querying and analyzing the data cubes as a whole, the system directly queries these specific features that emerge as abstractions of the underlying data [24–26]. This approach has been significantly enhanced by the research conducted by [9,27], which proposed a semantic and configurable extract, transform, load (ETL) mechanism. This ETL process serves as a pipeline responsible for extracting raster cells, recognized as observations, and establishing associations with geometric entities denoted as territorial units. However, despite the significance of this work, a rigorous

formalism and representation for the semantic model, as well as the mapping process between datasets and ontological features, are still lacking.

The second type of approach has centered on converting the entire geospatial dataset into the RDF data format and then storing and managing it in a spatial database. One of the main systems developed for this purpose was Strabon [28]. Strabon has been extended to be able to manage and query spatiotemporal data, and more particularly, time-stamped satellite images in their RDF format [7]. Upon that, and to assist in the translation between common geospatial formats and linked data, an open-source tool was developed to facilitate this task [8]. The main advantage of this approach is that the integrated data are gathered from a unified source; thus, the process of rewriting queries is not necessary, and the processing of queries is both centralized and prompt. However, these materialized approaches have major drawbacks, including the necessity of extra storage since the data are replicated, which implies an added storage cost as well as a maintenance cost. In addition, materialized data can quickly become stale if the data source is frequently updated.

The third approach consists of virtually linking a knowledge graph to data cubes without materializing it in the form of a set of RDF triples. Such an approach is essentially based on the virtual knowledge graph (VKG) paradigm, also known as an OBDA (ontology-based data access) [29–31]. A VKG consists of three main components: (1) a collection of data sources, (2) an ontology, and (3) mapping between the two. The ontology provides a high-level specification of the domain of interest and is semantically linked to the data sources by a mapping consisting of a series of mapping statements, where the standard mapping language is R2RML [32]. The ontology and mapping, called the VKG specification, expose the underlying data sources as a virtual RDF graph that can be accessed at query time using the standard W3C SPARQL language. When the user expresses SPARQL queries, the VKG system, e.g., Ontop [33], translates these queries into ones that are directly evaluated by the underlying database engine (e.g., SQL) without having to convert and materialize the original data into RDF and then store them in a triple store. The VKG system Ontop has implemented most of the GeoSPARQL functions [10] elaborated by the Open Geospatial Consortium (OGC), which is specifically designed as a geographic query language that extends SPARQL [34]. Using VKG, one can perform geospatial queries on relational data sources to query features, geometries, attributes, and their topological relationships using SPARQL [35,36]. Additional efforts have been made to manage knowledge consistency when querying geospatial data through VKG [37]. Also, work has been performed to show the potential of VKG to support some spatial-temporal data queries [38]. However, no work has been carried out on harnessing VKG to properly integrate and query raster data cubes.

Another solution aligned with the third approach involved the creation of a knowledge hypergraph system known as Onto-KIT, which aimed to integrate and manage raster data cubes with their heterogeneous formats [5,39]. This model presents a promising approach, as a hypergraph structure encompasses a generalization of the conventional graph, allowing edges to connect more than two nodes. Leveraging the hypergraph-based model enables the integration and semantic linking of data from multiple sources, facilitating enhanced information extraction in terms of relationship abundance. The hypergraph structure serves the dual purpose of representing the semantic model and virtually managing the population process. Notably, Onto-KIT incorporates a query processing engine based on hypergraphs, enabling the efficient rewriting and execution of fundamental geospatial queries on raster data cubes.

Another notable solution in the third category is the development of GeoLD, a system that offers a dedicated GeoSPARQL query engine to handle queries involving raster data cubes [6]. This system leverages the Rasdaman array database for efficient data storage and utilizes the Petascope implementation of WCPS to facilitate web access. To establish mapping between the ontological domain and the dataset, a novel mapping process called Coverage to RDF (C2RML) is proposed, wherein the conceptual model extends the

GeoSPARQL feature class with the coverage class. The system also incorporates a query engine that efficiently processes and rewrites the geospatial components of the SPARQL query into the WCPS language. The request is then transformed into the Rasdaman query language (Rasql) and executed on the Rasdaman system to obtain the desired data output. Through the mapping process, the entities in the ontological coverage model are populated, and the result is returned as an RDF dataset. One considerable advantage of this system is that it does not introduce a new collection of data operators to the SPARQL syntax, thus making it more accessible for regular SPARQL users to conduct geospatial queries on data cubes. Additionally, GeoLD demonstrates remarkable performance in executing spatiotemporal queries on data cubes, outperforming other systems like Geospark+ or Strabon. However, it should be noted that the system currently lacks the capability to perform aggregation operations, which are crucial in the geospatial query process, necessitating further development in this aspect.

Despite the valuable capabilities offered by both Onto-KIT and GeoLD, they have been limited to querying raster data cubes without supporting their combination with vector features along with their relationships. Such a combination would greatly improve the integration and semantics of geospatial querying.

In this work, we will rely on the virtual knowledge graph (VKG) paradigm as the foundation for proposing a comprehensive framework that facilitates the seamless integration of raster data cubes. Our objective is to investigate the potential of this approach (i.e., VKG) and evaluate its capacity to handle a diverse range of queries. These queries aim to retrieve information from the data cube while simultaneously combining it with another semantic layer, specifically feature-based semantics. This integration would enable the formulation of spatiotemporal queries in a more intuitive manner, leveraging ontological concepts at an appropriate level of abstraction. By exploring the limits of the VKG approach, we aim to demonstrate its efficacy in addressing the challenges associated with geospatial querying and enhancing the overall integration of raster data cubes.

### 3. An Ontology-Based Framework for Integrating Raster Data Cubes

In this section, we explain the conceptual aspects of our proposed framework. This mainly includes the modeling process that led us to the ontological representation of a semantic raster data cube using virtual graph knowledge.

In our ontological representation, we have chosen to rely on and use the GeoSPARQL standard only instead of reusing the data cube RDF (QB) ontology. The reason for adopting this approach is that QB, when used to represent raster data cubes, employs the concept of coverage as its basic element. This concept of coverage consists of a numerical array characterized by its spatial and temporal dimensions. However, the current landscape of OBDA systems is limited in its ability to handle spatiotemporal queries that involve combining raster data cubes with geometric features. Adopting such a coverage model by OBDA systems requires the development of new operators and functions adapted to the processing of raster data cubes and the realization of adding geometries to support such spatial-temporal queries.

In our framework, we introduce a semantic representation approach for raster data cubes. This approach is based on the concept of a spatiotemporal entity or location that encapsulates the spatial and temporal dimensions of the geospatial data cube concept. Although data cubes can possess multiple dimensions (i.e.,  $n$ -dimensional), in the context of Earth observation they are often simplified to a three-dimensional model ( $x, y, t$ ) or may also be a four-dimensional model known as a voxel data cube ( $x, y, z, t$ ) [40]. Within our ontology, we label this spatiotemporal entity `SpaceTimeLocation`, and its dimensions can be adjusted based on the specific data cube type under consideration. For our focus, we narrowed the selection down to the most prevalent and simplified form of data cube, which is the three-dimensional raster data cube. The spatial aspect of the `SpaceTimeLocation` is acquired from the GeoSPARQL feature class for spatial reference. As for the temporal dimension, we adopt a straightforward linear time model with a single dimension. If required, the temporal

dimension can be further extended to support a more complex model of time (cyclic time, relative time, temporal relations, . . . , etc.) by establishing connections between the SpaceTimeLocation concept and the OWL-time ontology. However, this is not within the interest or scope of this paper. The advantage of such a semantic representation is its seamless alignment with a large number of OBDA systems that offer GeoSPARQL-compliant functionalities and operations.

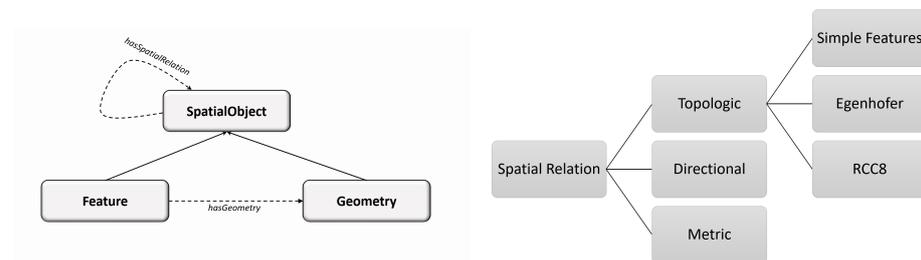
Additionally, with such a semantic representation, we would be able to formulate semantic geospatial queries that could combine discrete entities (i.e., features) with the properties of the continuous space represented by the data cube construct. For the modeling, we use the description logic, and at the same time, we combine it with the notation of other web semantics standards like RDF, RDFS, and GeoSPARQL. In addition, this framework includes a taxonomy of semantic geospatial queries.

### 3.1. Semantic Representation of Discrete Space

We consider our discrete space as a set of geospatial entities (i.e., features) associated with their geometries and forming spatial relationships between them. We model such a space as the quadruplet:

$$\langle E, T_y, S_R, F_g \rangle \quad \text{where :}$$

- $E$  represents the set of entities (i.e., features) that exist in the space and that inherit all their properties from the *Feature* class of the GeoSPARQL ontology (see Figure 1(left)), such as  $E \sqsubseteq \text{Feature}$ ;
- $T_y$  is a function that assigns to each entity  $e \in E$  its type, which we denote by the function:  $T_y: e \mapsto \text{type}$ . This function corresponds to the RDF property `rdf:type`. In this case, we try to simplify the model by assigning a unique type to each entity instance.
- $S_R$  consists of the set of spatial relations that can link two spatial entities,  $e_1$  and  $e_2$ , where  $S_R \subseteq E \times E$ . This spatial relation could be metric, directional, or topological (see Figure 1(right)). For the taxonomy of the topological relations between two entities, we adopt the one used by the GeoSPARQL standard (see Table 1).
- Finally,  $F_g$  is a function that assigns for each entity  $e \in E$  its geometrical representation, where  $F_g: E \rightarrow G$ . This function corresponds to the GeoSPARQL property `geo:hasGeometry`, where  $G \sqsubseteq \text{geo:Geometry}$ . We adopt the same taxonomy as the GeoSPARQL standard for this class, which is by default in compliance with OGC standards (see Figure 2).



**Figure 1.** An abstract overview of the classes and properties defined in the GeoSPARQL standard (**left sub-figure**) and the types of spatial relationships where only the topological ones are implemented by the GeoSPARQL standard (**right sub-figure**).

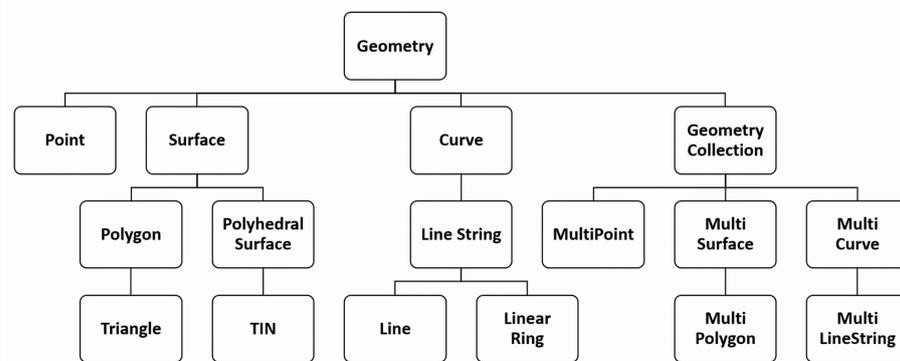
### 3.2. Semantic Representation of Continuous Space

Within a continuous space, a potentially infinite number of spatially distributed properties (e.g., temperature, pressure, humidity, etc.) can be identified at each location in that space. A location in geographic space can be determined using a two-dimensional coordinate system, represented by  $(x, y)$  coordinates, or a three-dimensional coordinate system, represented by  $(x, y, z)$  coordinates. Another alternative is to use geographic coordinates, which include latitude and longitude values, rather than projected coordinates.

The relationship or association between a location and its property's value is represented in the form of a function called a field [41,42]. A field can be seen as a set of observations where each one has been measured and associated with a geo-referenced component.

**Table 1.** Different topological function classifications implemented by GeoSPARQL standard [34].

OGC	Egenhofer	RCC8
geo:sfEquals	geo:ehEquals	geo:rcc8eq
geo:sfDisjoint	geo:ehDisjoint	geo:rcc8dc
geo:sfIntersects	geo:ehMeet	geo:rcc8ec
geo:sfTouches	geo:ehOverlap	geo:rcc8po
geo:sfWithin	geo:ehCoveredBy	geo:rcc8tppi
geo:sfContains	geo:ehContains	geo:rcc8tpp
geo:sfOverlaps	geo:ehInside	geo:rcc8ntpp
geo:sfCrosses	geo:ehCovers	geo:rcc8ntppi



**Figure 2.** GeoSPARQL geometry taxonomy in compliance with OGC standards [34].

In this section, we use the definition of the spatiotemporal field that has been developed in the works [43,44] and subsequently further specify it to yield a basic definition of the notion of a data cube, and finally propose an ontological model to capture the domain of the semantic data cube.

### 3.2.1. Definition of Spatiotemporal Field

In the works [43,44], the notion of the spatiotemporal field was defined as a function  $f$ :

$$f: S \times T \rightarrow \mathcal{P}(H) \text{ where}$$

- $S \times T$  consists of a spatiotemporal domain, such that  $S$  consists of a set of locations in geographic space that we denote as spatial atoms, and  $T$  is an ordered domain consisting of a sequence of timestamps:  $T = \{t_1, t_2, \dots, t_k, \dots, t_n\}$ .
- $H$  consists of a set of property-value pairs  $(M, V)$ , such that  $M$  is a set of properties and  $V$  is the set of numerical values associated with the property  $M$ .  $\mathcal{P}(H)$  represents the set of all subsets of  $H$  (i.e., the power set), which consists of the range of the field  $f$ .

The function  $f$  assigns for each spatiotemporal location  $l \in L$  a subset of property-value pairs  $(m, v) \in M \times V$ . This function will help us define the notion of the data cube by adding constraints to its domain.

### 3.2.2. Basic Definition of Data Cube

In our case, we use the previous definition in order to define the notion of the data cube. This notion is defined as a triplet as follows:

$$F_L = (S \times T, f, \mathcal{P}(H))$$

We consider the notion of a data cube as a special case of the spatiotemporal field  $f$  applying three constraints, which we express precisely as follows:

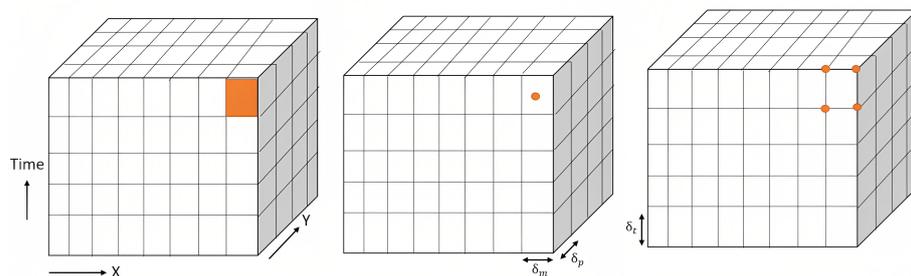
- $S$  consists of a set of sampled spatial atoms within a given spatial coverage. Each spatial atom  $s \in S$  is bidimensionally indexed by the indices  $(i, j) \in (m, p)$ , where  $m$  and  $p$  respectively denote the number of rows and columns. This uniform sample  $S$  is maintained across all timestamps  $t_k$  where every space–time location  $(s_{i,j}, t_k)$  has one and only one value from the range of the function  $f$ :

$$\forall s_{i,j} \in S, \forall t_k \in T \exists! h \in \mathcal{P}(H) \text{ such that } f(s_{i,j}, t_k) = h$$

- All time elements  $t_k \in T$  are temporally regular, which means that the temporal distance between each two consecutive times  $t_k$  and  $t_{k+1}$  corresponds to the duration constant  $\delta_t > 0$ . This constant is designated as the temporal resolution of a data cube.

$$\forall t_k, t_{k+1} \in T : (t_{k+1} - t_k) = \delta_t$$

- The spatial atom  $s \in S$  of a raster data cube could have many abstractions (i.e., spatial representation). As shown in Figure 3, a spatial atom can be either abstracted as a rectangle or as a point (excluding the center or the left/top corner of a rectangular raster cell). The spatial atoms in the raster data cube must be spatially regular, which means the size and shape of these spatial atoms (i.e., cells) should be consistent across the entire raster data cube coverage, forming a regular grid pattern. Additionally, the spatial arrangement and spacing between adjacent spatial atoms should be constant throughout the entire dataset.



**Figure 3.** Different abstraction of the data cube spatial atom.

In the case where the spatial atoms are abstracted as points, spatial regularity implies that vertically and horizontally, the contiguous spatial atoms are separated by the same consecutive distances  $\delta_m > 0$  and  $\delta_p > 0$ . We express this constraint as follows:

$$\forall s_{i,j}, s_{i,j+1}, s_{i+1,j} : (s_{i,j+1} - s_{i,j}) = \delta_p \text{ and } (s_{i+1,j} - s_{i,j}) = \delta_m$$

The pair  $(\delta_m, \delta_p)$  is designated as the spatial resolution of a data cube.

### 3.2.3. An Ontological Representation of Raster Data Cube

We propose an ontology in order to capture the key concepts along with their relationships and constraints pertaining to the notion of a semantic data cube. For this purpose, we employ description logic (DL), which is considered a fragment of first-order logic and serves as the formal expressive language of the ontological web language (OWL2). Using the expressiveness of DL, we formulate the set of axioms that bind higher-level concepts as well as the set of properties (i.e., roles) that would capture the notion of a semantic data cube. To do so, three main concepts that were evoked in the basic definition of a data cube will be semantically defined using DL, especially the main concepts  $S \times T$ ,  $f$ , and  $\mathcal{P}(H)$ .

Initially, we need to define the atomic concepts that constitute the building blocks on which our axioms will be based. The first concept consists of the spatial domain of

any raster data cube. We represent this concept with the class `dcb:FieldAtom`, which is defined as the union of the classes `geo:Point` and `geo:Polygon`. This class refers to the set  $S$  defined in the previous section.

$$FieldAtom \equiv Point \sqcup Polygon$$

In this definition, we assume that any spatial atom of a data cube could be represented either by a polygon, where a rectangle or a square is a special case of it, or by a point, which could represent the center or the left/right corner of each spatial atom of a data cube. The second concept consists of representing the set of all variable names (e.g., temperature, rainfall, humidity, etc.) with the class `dcb:FieldPropertyVar`. This class refers to the set  $M$  defined in Section 3.2. The third concept consists of the set of all values  $V$  that could be affected by the spatial atoms of a data cube. Often these values are of the numeric type and sometimes textual (e.g., a data cube that represents the variable of landcover types). However, since there is no need to restrict these values to only these types, we prefer to remain generic and represent the class `dcb:Value` as equivalent to the related generic type `rdfs:Literal`, which represents all data types in the system.

$$Value \equiv rdfs : Literal$$

The last concept consists of the time  $T$ , which we represent simply using the data property `DateTimestamp`, which represents the set of ordered instants.

The second step entails the derivation of key concepts by compounding the atomic ones and specifying the set of axioms that can be applied to any semantic data cube. In the previous section, we considered that a spatial atom in a data cube could be associated with one or a collection (i.e., a subset) of property-value pairs. This subset was represented by the set  $H = (M, V)$ . This leads us to represent the first key concept, which consists of the class `dcb:FieldPropValue` as follows:

$$\begin{aligned} FieldPropValue &\sqsubseteq = 1 \text{ hasFieldProp.FieldPropertyVar} \sqcap \\ &= 1 \text{ hasFieldValue.Value} \end{aligned}$$

The class `dcb:FieldPropValue` is subsumed by the intersections of two classes  $= 1\text{hasFieldProp.FieldPropertyVar}$  and  $= 1\text{hasFieldValue.Value}$ , where the former refers to all instances having exactly one relationship `dcb:hasFieldProp` with the class `dcb:FieldPropertyVar` and the latter refers to all instances having exactly one relationship `dcb:hasFieldValue` with the class `dcb:Value`. In other words, this description implies that each instance of the `dcb:FieldPropValue` class is composed of a unique pair made up of an instance of the `dcb:FieldProperty` class together with an instance of the `dcb:Value` class. The second key concept consists of defining the spatiotemporal domain ( $S \times T$ ) related to a semantic data cube, which is represented with the class `dcb:SpaceTimeLocation` and the following axiom:

$$\begin{aligned} SpaceTimeLocation &\equiv \exists \text{ hasSpatialRef.FieldAtom} \sqcap \\ &\quad \exists \text{ hasTemporalRef.TimeStamp} \sqcap \\ &\quad \forall \text{ hasField.FieldPropValue} \end{aligned}$$

In this axiom, we define the class `dcb:SpaceTimeLocation` as an equivalence of the intersection of three classes: the  $\exists \text{hasSpatialRef.FieldAtom}$ , which starts with the existential quantifier  $\exists$  that denotes the set of all instances connected to the class `dcb:FieldAtom` through the relation `dcb:hasSpatialRef`. The class  $\exists \text{hasTemporalRef.TimeStamp}$ , for which the same case applies and which denotes the set of all instances connected to the data type `TimeStamp` through the relation `dcb:hasTemporalRef`. And the third class,  $\forall \text{hasField.FieldPropValue}$ , which starts with the universal quantifier  $\forall$ , defines the set of all instances that have only been connected to the class `dcb:FieldPropValue` through the relationship `dcb:hasField`. This means that any instance that is a sub-

ject of the `dcb:hasField` property must necessarily be connected to the instance of the `dcb:FieldPropValue` class. However, such a definition does not necessarily imply that every instance of `dcb:SpaceTimeLocation` must be associated with an instance of the class `dcb:FieldPropValue`. In other words, for an instance to stand for a space–time location, it must at least be related to a spatial atom (i.e., a cell) represented by the `dcb:FieldAtom` class and have at least one temporal reference represented by the `TimeStamp` data type. Even if such a space–time location is not associated with a property-value pair, it is nevertheless still considered an instance of the `dcb:SpaceTimeLocation` class.

This last point leads us to define the notion of observation and to clearly differentiate it from the notion of spatiotemporal location. Thus, an observation could be seen as a subclass of `dcb:SpaceTimeLocation` with an obligation to be involved in the `dcb:hasField` relation along with the `dcb:FieldPropValue` class. In this case, we formulate this axiom as follows, using the intersection of two classes:

$$\begin{aligned} \textit{Observation} &\equiv \textit{SpaceTimeLocation} \sqcap \\ &\quad \exists \textit{hasField}.\textit{FieldPropValue} \end{aligned}$$

This implies that an observation must possess spatial and temporal references, and furthermore, it must be associated with at least one property-value pair that is an instance of the class `dcb:FieldPropValue` and this is necessarily through the relation `dcb:hasField`. This relationship represents the most important association that plays the role of linking the spatiotemporal domain of a data cube with its range. Precisely, this relation `dcb:hasField` represents the function  $f$ , which is considered the definition of the spatiotemporal field in the previous section. For this reason, we chose to give a definition of this relationship as follows:

$$\begin{aligned} \exists \textit{hasField}.\top &\sqsubseteq \leq 1 \textit{SpaceTimeLocation} \\ \top &\sqsubseteq \forall \textit{hasField}.\textit{FieldPropValue} \end{aligned}$$

The relationship `dcb:hasField` has as its domain the class `dcb:SpaceTimeLocation` and exclusively as its range the class `dcb:FieldPropValue`. Since there are no cardinality restrictions on the range of `dcb:hasField`, an instance of `dcb:SpaceTimeLocation` can have from zero to multiple instances related to the `dcb:FieldPropValue` class. This means that a space–time location might have a collection of property-value pairs. This last point was expressed in the previous section using the range of the function  $f$ , which is denoted by the power set  $\mathcal{P}(H)$ .

On the other hand, the existence of the cardinality notation  $\leq 1$  in the domain of the function `dcb:hasField` implies that this relation is inversely functional. This implies that each instance of the range related to the `dcb:hasField` relation can be associated with at most one spatiotemporal location. Such a characteristic leads us to specify the inverse function related to the `dcb:hasField` relation that we call `hasSTL`.

$$\textit{hasSTL} \equiv \textit{hasField}^{-}$$

This relation would therefore be functional and would allow one to easily find, for each instance of the class `dcb:FieldPropValue`, at most one associated instance of the `dcb:SpaceTimeLocation` class.

The last concept and axiom related to the design of our ontology concerns the notion of a data cube and how it is defined in relation to the other concepts. In this model, a data cube concept is a subclass of the intersection of three anonymous classes:

$$\begin{aligned} \textit{DataCube} &\sqsubseteq = 1 \textit{hasCoverageParams}.\textit{CoverageParams} \sqcap \\ &\quad \exists \textit{hasPropSet}.\textit{FieldPropertyVar} \sqcap \\ &\quad = 1 \textit{hasTimeExtent}.\textit{TimeExtent} \end{aligned}$$

- An instance of the `dcb:DataCube` class must first be associated with one and only one instance of the `dcb:CoverageParams` class. This class defines the main spatial and temporal parameters that shape the spatiotemporal domain of a semantic data cube. It defines exactly its spatial resolution (i.e.,  $(\delta_m, \delta_p)$ ) as well as its temporal resolution (i.e.,  $\delta_t$ ) and the coordinates of the point that constitutes the left corner of the data cube. With these parameters, we will be able to reconstruct the spatiotemporal domain of any given `dcb:DataCube` instance.
- Each instance of `dcb:DataCube` records the values of one or more distributed variables (e.g., temperature, evapotranspiration, cloud cover, etc.). Thus, for each instance of the `dcb:DataCube` class, there must be a relationship with at least one instance of the `dcb:FieldPropertyVar` class. This last class represents the set of all the distributed variables. Thus, the relation between the class `dcb:DataCube` and the other classes of our ontology is materialized by its relation to the class `dcb:FieldPropertyVar` through the relation `dcb:hasPropSet`.
- Any instance of the `dcb:DataCube` class embodies a unique temporal extent for which it holds. The temporal extent signifies the valid time interval between when the data cube starts recording its values and the end of that recording. Similarly to the constraint in the `dcb:CoverageParams` class, each instance of `dcb:DataCube` must possess only one and only one temporal extent.

Another property that could potentially be assigned to the `dcb:DataCube` class is the abstract relation `dcb:DataCubeBinaryRelation` which could specifically represent the set of topological and arithmetic operations between each pair of two instances of the `dcb:DataCube` class. A `dcb:DataCube` class can also be associated with a class that stores all of its metadata, which may include the type of sensors used, the owner of such data, and others. The design of the whole ontology, which mentions the characteristics of data cubes and their relationships with all other classes, is illustrated in Figure 4.

### 3.3. Ontology-Based Geospatial Queries

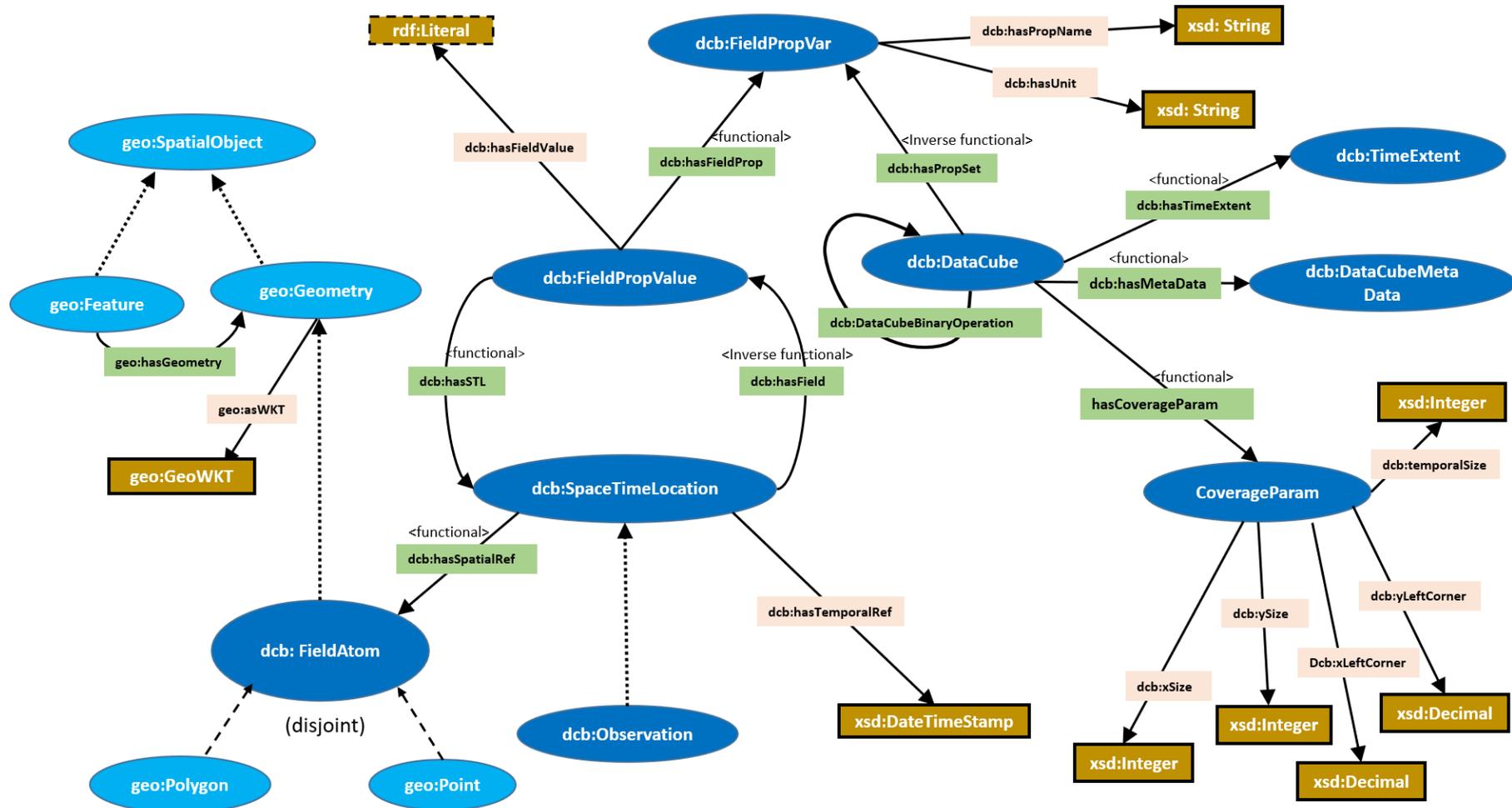
To evaluate the expressivity of the proposed semantic representation model, we carefully designed a set of geospatial queries. These queries can be classified into a taxonomy that captures most of their aspects. Thus, we present a taxonomy within which each query can find its semantic position. We then take a sample of scenarios and classify them according to this taxonomy.

#### 3.3.1. Taxonomy of Semantic Geospatial Queries

In this taxonomy, and at a high level of abstraction, we characterize any semantic geospatial query both by the nature of the operations that it involves and by the nature of the output that it might yield.

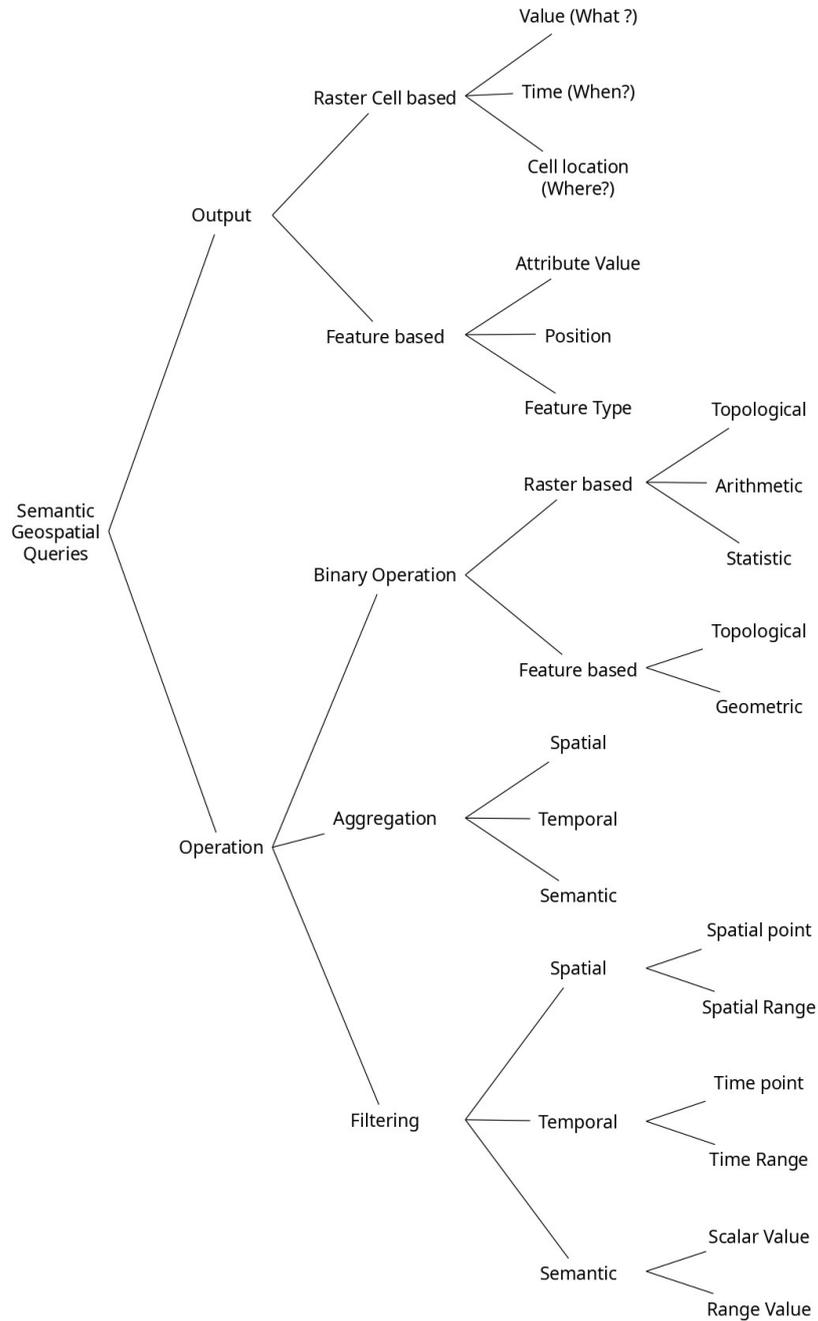
Regarding the nature of the operation, we distinguish three main ones: binary operation, aggregation, and filtering. The binary operation can be of raster type, which means that the two operands of such an operation are data cube/raster layers, or it could be feature-based, where the operands are features. These operations are principally topological, geometric, or arithmetic. For the aggregation operation, it could be spatial, which represents the operation of reducing variability within space; temporal, which, by analogy with space, is the operation of reducing variability over the time interval; or semantic, which represents the operation of reducing variability over the value range. The same goes for the filtering operation, which consists of the condition that the query must meet to give the final result. The filter used can also be spatial, temporal, or semantic (see Figure 5).

Note that there is no exclusivity between each type of operation. This implies that a geospatial query can, for example, include both aggregation and filtering operations. Similarly, aggregation can be spatial or temporal in the same query.



**Figure 4.** Diagram of the developed raster data cube vocabulary. Solid lines indicate object or data properties, whereas arrows indicate the direction of property relations. Dotted lines indicate subclass relations. Dashed lines without arrowheads indicate the connection between disjoint classes. The green rectangles indicate object properties, while the pink rectangles indicate data properties. In the case of object properties, their functional or inverse-functional nature is specified where applicable. Classes represented in dark blue correspond to the classes of the ontological vocabulary we have developed, while those in lighter blue represent classes reused from another ontology (in this case, the GeoSPARQL ontology).

In terms of the output of a semantic geospatial query, it consists of the nature of the answer sought by the query, which could be raster-based, where the target could be the value, time, or location of the raster cell, or it could be feature-based, where the target is the value of an attribute or the position or the type of the feature (see Figure 5).



**Figure 5.** Taxonomy used for the classification of geospatial semantic queries.

### 3.3.2. Scenarios and Classifications

We propose a set of geospatial queries that combine the information retrieval from the data cube with a set of features that are in the same spatial domain as the data cube (see Table 2). These queries represent a set of generic scenarios related to the study of any given geographic phenomenon. These queries are only a sample of the total number of scenarios we could obtain by using all the possibilities within our proposed geospatial query framework.

The first four queries are classified as basic since they are exclusively related to retrieving different patterns of information from data cubes over time without including any information about geospatial features. For instance, Q1 could be labeled a time range query (TRQ) since it is searching for the raster cells (i.e., spatial atoms) and their values over a time range. Q2 could be mostly termed as a field range query (FRQ) since it seeks to retrieve raster cells and their values that fall within a range of values related to a given distributed variable. Q3 might be considered mostly a spatial point query (SPQ) since it seeks to retrieve the raster cells and values related to a given spatial point. The last basic query, Q4, could be characterized as a spatial range query (SRQ) since it is concerned with retrieving the raster cells and their values that lie within a given spatial region. We give more accurate classifications regarding these four queries in both Tables 3 and 4.

**Table 2.** List of the semantic geospatial queries that should be tested as scenarios for the use case.

	Query Code	Textual Description
Basic	Q1	What are the raster cells and their values related to the variable $V$ between two times, $t_i$ and $t_j$ ?
	Q2	What are the raster cells and their values related to the variable $V$ where these values are greater than the value $v_i$ between two times, $t_i$ and $t_j$ ?
	Q3	What are the raster cells and their values related to the variable $V$ at the point $p_i$ between two times, $t_i$ and $t_j$ ?
	Q4	What are the raster cells and their values related to the variable $V$ inside the region $R$ between two times, $t_i$ and $t_j$ ?
Advanced	Q5	How is the spatial distribution of the maximum of the variable $V$ within a feature $f$ between two times, $t_i$ and $t_j$ ?
	Q6	How is the spatial distribution of the highest difference between two variables, $V_j$ and $V_k$ , in each raster cell within a feature $f$ during a period between $t_i$ and $t_j$ ?
	Q7	What and where is the maximum value of the variable $V$ inside the feature $f_a$ of the type $T_a$ holding the maximum number of features of the type $T_b$ between two times, $t_i$ and $t_j$ ?
	Q8	How does the maximum value of the variable $V$ evolve during the period between $t_i$ and $t_j$ ?
	Q9	When and where does the highest value of the variable $V$ occur during the period between $t_i$ and $t_j$ , and which features and their types does this value fall within?

**Table 3.** Classification of the set of spatiotemporal queries Queries based on the data cube and feature nature of output as well as their basic operations.

Queries Ref	Data Cube-Based Output		Data Cube Binary Operations		Feature-Based Outputs			Feature-Based Operations	
	Value	Location/Cell	Topological	Arithmetic/Statistics	Attribute Value	Position	Type	Geometric (e.g., Buffer, Area)	Topologic
Q1	×	×							
Q2	×	×							×
Q3	×	×							×
Q4	×	×							
Q5	×	×							×
Q6	×	×		×					
Q7	×	×				×			×
Q8	×	×							×
Q9	×	×			×	×	×		×

On the other side, we consider the last five queries advanced ones. This is mainly because all these queries combine information related to geospatial features with the semantic data cube. These queries also apply to the processes of spatial and temporal aggregation. We characterize these queries as follows:

- Queries Q5 and Q6, for instance, first apply temporal aggregation over a time interval. However, Q5 applies it to a distributed variable, while Q6 performs a difference operation between two given distributed variables, which could be considered a map algebra operation qualified as a local one.
- Q7 applies spatial and temporal aggregation to find the maximum value of a given distributed variable. This value must be within the feature  $f_i$  that contains the maximum number of features of the specific type  $T_b$ . Thus, this query additionally involves two spatial topological operations, one between the features of two entity types and the second between the selected feature and the data cube.
- Q8 applies only a spatial aggregation for each moment in a time interval over a given distributed variable  $V$ . This leads to a trajectory with a sequence of time-stamped value-point pairs.
- The last query, Q9, also applies spatial aggregation for each time point but also looks at which feature the trajectory of the highest value crosses and determines the type of these features. In addition, this query involves a spatial topology operation between the feature and the data cube.

**Table 4.** Classification of the set of spatiotemporal queries based on the nature of aggregation and used filter.

Queries Ref.	Aggregation-Based			Filter-Based					
	Spatial	Temporal	Semantic	Spatial		Temporal		Semantic	
				0-Dimension	2-Dimension	Time Point	Time Range	Scalar	Range
Q1							×		
Q2				×			×		
Q3					×		×		
Q4							×		×
Q5		×			×		×		
Q6		×			×		×		
Q7	×	×	×		×		×		
Q8	×				×		×		
Q9	×				×		×		

The precise positioning of these advanced queries can be seen in Tables 3 and 4. Additionally, each query mentioned in this table has a corresponding equivalent that effectively illustrates it through a specific example. The chosen illustration revolves around forest fire data, whereas raster data includes variables such as surface temperature and evapotranspiration. Similarly, features are represented using entities such as urban regions, territorial zones, and airports (see Section 5.1). Readers can refer to Appendix A, which covers queries Q1 to Q4. Similarly, for queries Q5 to Q9, they can refer to Section 5.2.

#### 4. The Implementation of the Proposed Framework

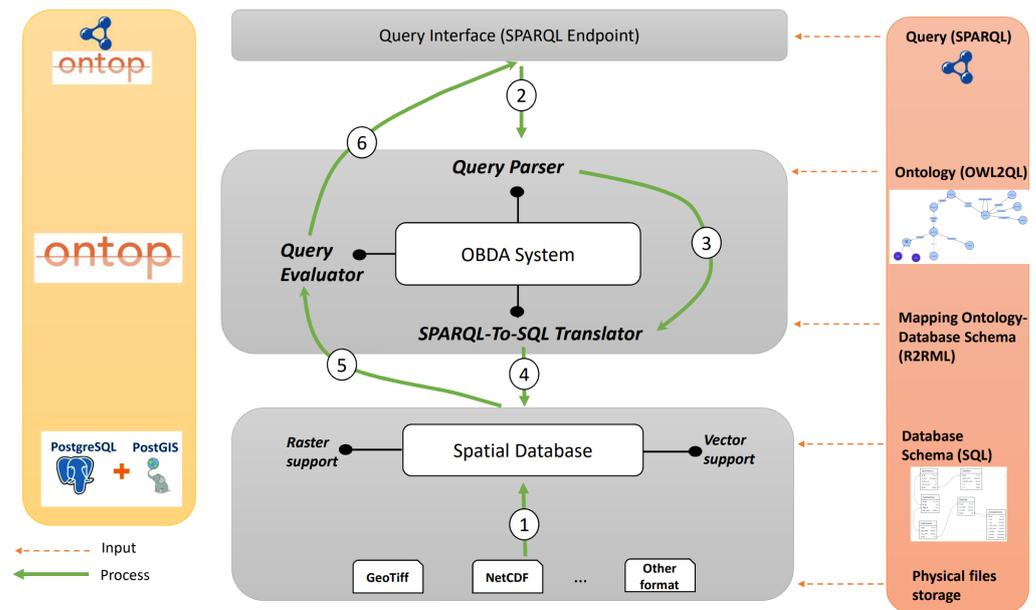
In this section, we explain the overall architecture that our framework uses to implement the underlying concepts developed. In addition, we describe the VKG specifications that depict the process through which the framework was implemented.

##### 4.1. System Architecture

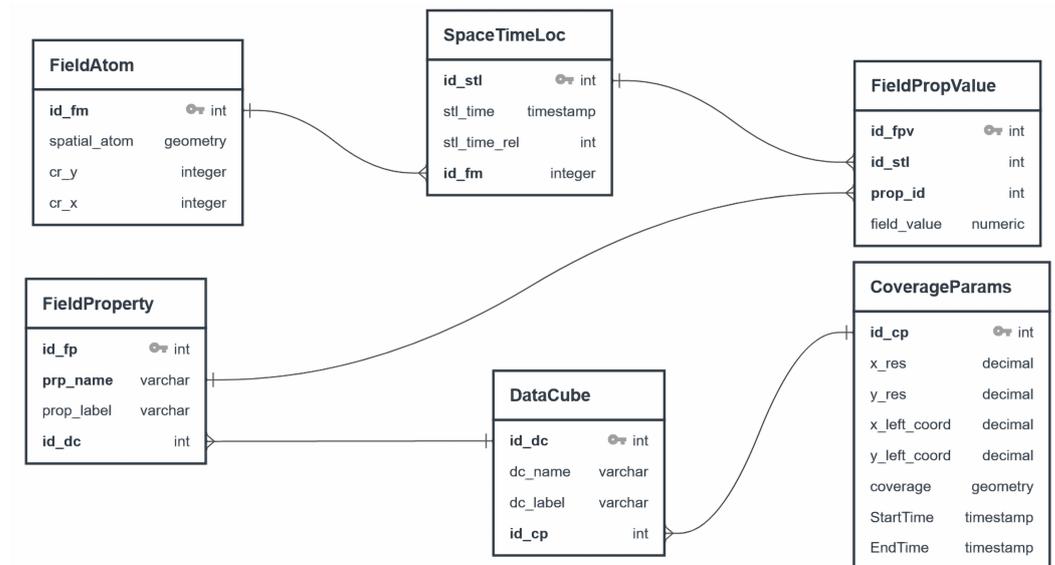
As shown in Figure 6, our system architecture is composed of three main layers:

- **Data integration layer.** This layer lies at the bottom of the architecture and involves integrating raw data from different geospatial sources, especially raster (e.g., Geotiff and NetCdf formats) and vector (e.g., Shapefiles and GML formats) data, within a spatial database that supports both raster and vector data (e.g., PostgreSQL/Post-

GIS, Oracle, etc.) using well-designed database schemas. In our case, after creating our database schemas in PostgreSQL, we extracted and loaded the data from their raw formats (ex. Netcdf, GeoTiff, etc.) along with their metadata into the selected PostgreSQL relational database. Then, we used PostGIS raster functions, mainly ST\_PixelAsPoint, to extract the center of each pixel related to the coverage of the data cube and ST\_Value to extract the value of such pixel. The aim of this process (i.e., process 1) was to fill the schema of our database and, in particular, the three tables of FieldAtom, SpaceTimeLoc, and FieldPropValue (see Figure 7).



**Figure 6.** System architecture is presented as three layers and six processes. The orange dashed arrows indicate the inputs to each system component. The green arrows indicate the processes involved in the query answer. The numbers indicate the order in which each process must be performed to obtain an answer to a query.



**Figure 7.** Database design for representing semantic data cube.

- Ontology-based data access (OBDA) layer.** This layer represents the core of the system. At this layer, a semantic model in the form of an ontology is defined in order to provide a formal and high-level representation of the domain of interest. In addition,

the mapping must be defined to link the classes and properties of the ontology to the tables and attributes of the database schemas. There are three main components in this layer that are responsible for four related processes:

- **Query parser:** this component is the upper part of the SPARQL query engine. Its main role is to receive SPARQL queries and perform a syntax check to verify their correctness and compliance with the query language syntax. Once queries have passed the syntax check, this component forwards them to the translation module for further processing (process 3).
- **SPARQL-to-SQL translator:** this component receives the analyzed SPARQL query as input and uses the VKG specification, which includes the ontology and its mapping. With the support of a reasoner that uses the axioms defined in the ontology, it transforms the query into an SQL equivalent. The reformulated query is then sent for execution to the spatial database engine (process 4).
- **Query Evaluator:** this component utilizes the output of an SQL query executed within the spatial database engine (process 5). It assesses the query result and converts it into a collection of RDF triples, without physically materializing the data. The resulting set of RDF triples includes the original RDF triples representing the queried records from the database as well as newly inferred RDF triples. The entire collection of RDF triples generated for a specific query is known as the virtual knowledge graph (VKG). Finally, the result of this translation is sent to the query interface for visualization (process 6).
- **Query interface for SPARQL layer.** This layer consists of a front-end interface that allows the interaction between users and the proposed system. Users can query the classes and relations of the ontology using the SPARQL query language (process 2) and then display the results of their queries.

#### 4.2. The VKG Specifications

As explained earlier, to implement our framework, we need to define the VKG specification triple  $(O, M, S)$  with three key ingredients: the ontology,  $O$ , which is defined in Section 3.2.3, the data source schema,  $S$ , which structures the data source into a database schema, and the mapping,  $M$ , which connects the classes and properties of the ontology to the tables and attributes of the database schema.

For the design of the data source schema, we define the database design model that is most suitable for our defined ontology. This data model is designed with respect to the normal form to benefit from all optimizations deployed in the VKG system as well as to avoid some redundancies and self-joining during query execution and evaluation. The logical data model (see Figure 7) is composed of six interrelated tables that correspond to the classes defined in our ontology.

For instance, the class `dcb:SpaceTimeLoc` is represented by the table `SpaceTimeLoc`, where each instance is characterized by a temporal component (i.e., `stl_time` attribute) and by a spatial component that is an instance of the spatial domain of a data cube. This spatial domain is modeled as a table named `FieldAtom`, which stores the geometries of all cells in a given data cube. Any given spatiotemporal location should have one and only one spatial component (i.e., a raster cell), and this spatial component could spawn or be the source of multiple spatiotemporal locations depending on the size of the temporal dimension. Thus, to represent such a functional relationship, we move the primary key of the table `FieldAtom` to the table `SpaceTimeLoc` and consider it a foreign key.

The data cube class is mainly defined on the basis of the relationship between a spatiotemporal location and a subset of property-value pairs. This subset is materialized by the table `FieldPropValue`. This table mainly contains a foreign key (`prop_id` attribute), which refers to the table `FieldProperty` containing the set of all distributed variable names (i.e., the field properties). Correspondingly, the table `FieldPropValue` contains the value (`field_value` attribute), which is associated with a specific property and characterizes a unique spatiotemporal location.

Finally, since a data cube can be mainly characterized by the set of its properties (i.e., variables) along with the set of parameters related to its spatial and temporal coverage, we therefore create a table named `DataCube` and we link it to the table `FieldProperty`, where each property of this table characterizes one and only one instance of the `DataCube` table. In a similar way, we create the table `CoverageParams`, which contains all the parameters that characterize the spatial and temporal extent of a given data cube. A data cube must be characterized by only one instance of the table `CoverageParams`.

In order to link the database schema  $S$  to the ontology  $O$ , we need to define a mapping  $M$  between them. Following the definition in [31,45], a mapping,  $M$ , consists of a set of assertions of the form:

$$m: \phi(a_1, a_2, a_3, \dots, a_n) \leftrightarrow \psi(a_1, a_2, a_3, \dots, a_n)$$

- In this case,  $\phi(a_1, a_2, a_3, \dots, a_n)$  represents a query on a data source schema,  $S$ , that selects the attributes  $(a_1, a_2, a_3, \dots, a_n)$  of all tables generated by such a query. In the case where the data source schema  $S$  is relational, such a query is expressed using SQL.
- $\psi(a_1, a_2, a_3, \dots, a_n)$  represents a RDF triplet statement specifying the way to use RDF terms constructed from database values to instantiate classes and properties. More precisely, such a model indicates either that an RDF term (representing an object) is an instance of a class, or that such a term is related by a property to another term representing an object or a literal value.

The standard language for representing the mapping,  $M$ , is defined by the W3C R2RML specification. For instance, in Table 5, we define a set of mapping assertions to associate our database schema,  $S$ , with our ontology terms. For example, in the mapping assertion called `spaceTimeLocation`, the source SQL query  $\phi$  outputs the attributes (`id_stl`, `time_index`, `stl_time`, `id_fm`), which are mapped to the RDF triple statement  $\psi$ . The statement  $\psi$  binds the identifier of the table `spaceTimeLoc` (i.e., `id_stl` attribute) with the URI of the class `dcb:SpaceTimeLocation` in order to uniquely identify each instance of this class. Then, in the same mapping, we associate the attribute `stl_time` with the data property `dcb:hasDateTime` and the foreign key `id_fm` with the data property `dcb:hasSpatialRef`, which refer respectively to the temporal and spatial components of each instance of the class `dcb:SpaceTimeLocation`.

Once our relational schemas are populated, we obtain the source database,  $D$ , which will serve as the population source of our ontology through the mapping,  $M$ . This population consists of a set of RDF assertions. Conceptually speaking, we call  $M(D)$  the set of RDF assertions on a database,  $D$ , through the mapping,  $M$ . Then, by applying the semantics and axioms related to our ontology, we could derive a new RDF graph, such as the relational data on the source, which is denoted  $G_{M,D}$ .

In practice, the ontology is populated only at the time of the SPARQL query execution. The SPARQL query is rewritten into a SQL query based on the mapping,  $M$ , and the result is transformed into a knowledge graph, as illustrated in Section 4.1.

**Table 5.** Mapping between the database schema and the ontology.

Mapping Assertions		
Mapping ID	Source: SQL Query	Target: Turtle Template
FieldAtom	<i>select id_fm, st_astext(field_loc) as field_loc, coord_rel_x, coord_rel_y from FieldAtom</i>	<b>dcb:fad-{id_fm} a dcb:FieldAtom;</b> <i>dcb:xRelCoord {coord_rel_x}^^xsd:integer;</i> <i>dcb:yRelCoord {coord_rel_y}^^xsd:integer;</i> <i>dcb:asFieldWKT {field_loc}^^geo:wktLiteral.</i>
SpaceTimeLocation	<i>select from SpaceTimeLoc</i>	<b>dcb:stl-{id_stl} a dcb:SpaceTimeLoc;</b> <i>dcb:hasRelTime {time_index}^^xsd:integer;</i> <i>dcb:hasDateTime {stl_time}^^dateTimeStamp;</i> <i>dcb:hasSpatialRef dcb:fad-{id_fm}.</i>
Field_prop_value	<i>select from FieldPropValue</i>	<b>dcb:fpval-{id_stlfv} a dcb:FieldPropValue;</b> <i>dcb:hasFieldValue {fvalue};</i> <i>dcb:hasVariableName {prop_name};</i> <i>dcb:hasFieldProp dcb:fpv-{prop_id};</i> <i>dcb:hasSTL dcb:stl-{id_stl}.</i>
DataCube_coverage	<i>select id_ps, ps_name, ps_label, dc.id_cp, upperleftx, upperlefty, dc_width, dc_hight from property_set dc, coverage_params cp where dc.id_cp = cp.id_cp</i>	<b>dcb:dc-{id_ps} a dcb:DataCube;</b> <i>dcb:dcHasName {ps_name}^^xsd:string;</i> <i>dcb:dcHasLabel {ps_label}^^xsd:string;</i> <i>dcb:hasCoverageParams dcb:cp-{id_cp}.</i> <b>dcb:cp-{id_cp} a dcb:CoverageParams;</b> <i>dcb:xLeftCorner {upperleftx}^^xsd:double;</i> <i>dcb:yLeftCorner {upperlefty}^^xsd:double;</i> <i>dcb:xspatialSize {dc_width}^^xsd:integer;</i> <i>dcb:yspatialSize {dc_hight}^^xsd:integer</i>
Field_Property	<i>select from FieldProperty</i>	<b>dcb:fpv-{id_prop} a dcb:FieldPropVar;</b> <i>dcb:hasPropName {prop_name}^^xsd:string;</i> <i>dcb:hasPropSet dcb:dc-{ps_id}.</i>

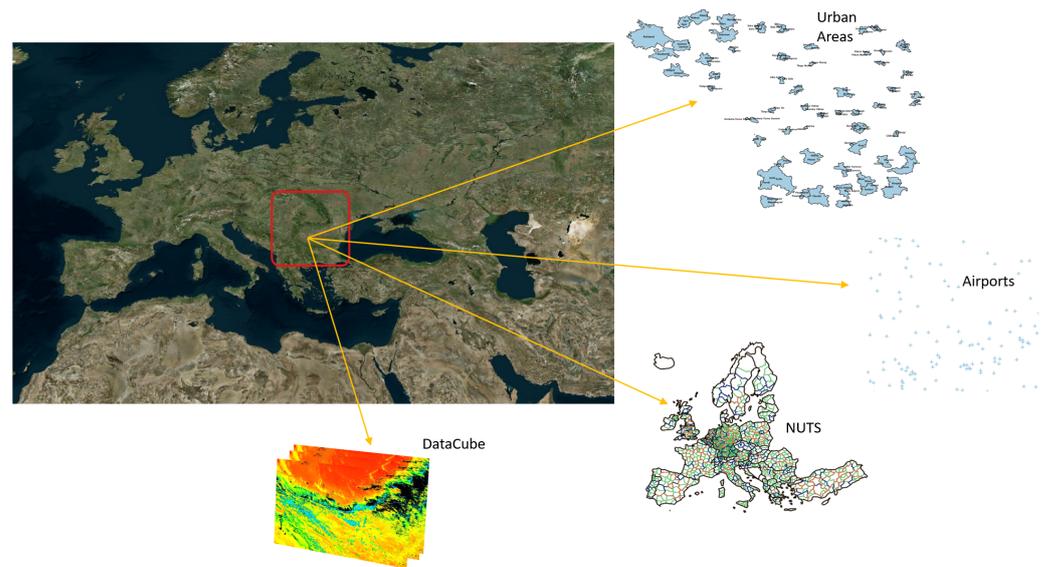
## 5. Use Cases and System Performances Evaluation

We apply our approach to a specific use case using a given dataset by formulating and executing nine scenarios in the form of spatiotemporal queries. We distinguish between basic queries that focus on retrieving information only from raster layers and advanced queries that combine raster layers with discrete features.

### 5.1. Use Case and Dataset

The modeling approach was tested and applied to the forest fire data, as future climate variability is expected to increase the risk and severity of forest fires in many regions, especially in southern Europe and the Mediterranean region (see Figure 8). Properly managing and exploring raster data cube variables, such as land surface temperature, in their spatial context can help scientists better recognize areas or regions most likely to experience forest fires in the future.

For this experiment, we use two main types of geospatial datasets (see Figure 8). The first type is feature-based data that consists of three vector layers, namely urban areas, airports, and NUTS [46]. NUTS (nomenclature of territorial units for statistics) is a hierarchical system dividing the economic territory of the EU and the UK into three levels known as NUTS 1 (major socio-economic regions), NUTS 2 (basic regions for the application of regional policies), and NUTS 3 (small regions for specific diagnoses).



**Figure 8.** Study area and dataset.

The second dataset type is data cubes, consisting of sequences of time-stamped raster layers of multiple variables. In our case, we select four variables that are resampled at a daily temporal resolution and with a spatial resolution of 1 km. These variables are land surface temperature during the day, land surface temperature at the night, leaf area index, and evapotranspiration. The spatial size of each temporal raster layer is  $117 \times 142$  pixels, whereas the temporal dimension we pick up starts from 15 March 2009 and ends on 3 May 2009, and consists of 50 temporal snapshots for each variable (daily temporal resolution).

### 5.2. SPARQL Query Formulation and Visualization

In this section, we express the set of queries that are presented in Table 2. We map these queries to our dataset to make them more specific and utilize SPARQL to formulate and answer such queries. We use the query editor extension developed by the Ontop system for the Protégé software [33]. We only display the SPARQL formulations relative to the advanced queries (i.e., from Q5 to Q9). However, the reader can find the SPARQL formulations relative to the first four queries in Appendix A. To reformulate queries, we need to define a list of prefixes that must be declared for each query being reformulated (see Table 6). For query execution, the Ontop system is employed. This system provides outcomes in either tabular or JSON format and concurrently generates a translation of the SPARQL query into an SQL query optimized for PostGIS. Utilizing the existing connection interface between QGIS and PostGIS, these SQL queries are then executed within the QGIS platform, facilitating the subsequent visualization of results within the QGIS environment.

**Table 6.** List of prefixes used for SPARQL queries.

Acronym	URL
geo	< <a href="http://www.opengis.net/ont/geosparql/">http://www.opengis.net/ont/geosparql/</a> > (accessed on 10 June 2023)
geof	<a href="http://www.opengis.net/def/function/geosparql">http://www.opengis.net/def/function/geosparql</a> (accessed on 10 June 2023)
units	<a href="http://www.opengis.net/def/uom/OGC/1.0/">http://www.opengis.net/def/uom/OGC/1.0/</a> (accessed on 10 June 2023)
crs	<a href="http://www.opengis.net/def/crs/OGC/1.3/CRS84/">http://www.opengis.net/def/crs/OGC/1.3/CRS84/</a> (accessed on 10 June 2023)
geo-sf	<a href="http://www.opengis.net/def/dataType/OGC-SF/1.0/">http://www.opengis.net/def/dataType/OGC-SF/1.0/</a> (accessed on 10 June 2023)
dcb	<a href="http://www.semanticweb.org/OntologyRasterDataCube/">http://www.semanticweb.org/OntologyRasterDataCube/</a> (accessed on 10 June 2023)
geop	<a href="http://www.opengis.net/def/property/OGC-GeoSPARQL/1.0/">http://www.opengis.net/def/property/OGC-GeoSPARQL/1.0/</a> (accessed on 10 June 2023)

**Q5** What is the spatial distribution of the maximum soil temperature within the urban area named Sofia between two times, 15 March 2009 and 25 March 2009?

```

select ?location (max(?val) as ?max_soil_temp) where {
  ?stl dcb:hasDateTime ?time_ref;
  dcb:hasSpatialRef ?sp. ?sp dcb:asFieldWKT ?location.
  ?stl dcb:hasField ?f. ?f dcb:hasFieldValue ?val;
  dcb:hasVariableName ?var.
  ?urb dcb:area_name ?urb_name;
  geo:hasGeometry ?urbgeom.
  ?urbgeom geo:asWKT ?urb_wkt.
  FILTER (?time_ref >= "2009-03-14T00:00:00+01:00"^^xsd:dateTimeStamp && ?time_ref <=
    "2009-03-25T00:00:00+02:00"^^xsd:dateTimeStamp && ?var = 'LST_Day_1km' &&
    geof:sfWithin(?location, ?urb_wkt) && ?urb_name = 'Sofia') } group by ?location

```

**Q6** What is the spatial distribution of the highest difference in soil temperature in each raster cell between day and night within a feature f during a time interval between 15 March 2009 and 25 March 2009?

```

select ?location (MAX((?v1 - ?v2)) as ?max_diff_soil_temp) where {
  ?stl dcb:hasDateTime ?time_ref;
  dcb:hasSpatialRef ?sp. ?sp dcb:asFieldWKT ?location.
  ?stl dcb:hasField ?fpv;
  dcb:hasField ?fpv2. ?fpv dcb:hasFieldValue ?v1;
  dcb:hasVariableName 'LST_Day_1km'.
  ?fpv2 dcb:hasFieldValue ?v2; dcb:hasVariableName~'LST_Night_1km'.

  ?urb dcb:area_name ?urb_name;
  geo:hasGeometry ?urbgeom.
  ?urbgeom geo:asWKT ?urb_wkt.
  FILTER (?time_ref >= "2009-03-14T00:00:00+01:00"^^xsd:dateTimeStamp && ?time_ref <=
    "2009-03-25T00:00:00+02:00"^^xsd:dateTimeStamp && geof:sfWithin(?location,
    ?urb_wkt) && ?urb_name = 'Sofia') } group by ?location

```

Q5 applies temporal aggregation along with a spatial filter to extract the maximum value of soil temperature for each raster cell in the selected urban area. Figure 9 (left) shows the visualization of the query result from Q5, with colors from blue to red indicating the maximum soil temperature from low to high. While Q6 applies the same operations, it calculates, in addition, the difference between day and night soil temperatures, which requires crossing two data cubes. In Figure 9 (right), the red-colored area shows a strong disparity, while the blue-colored area shows a relative concordance between day and night soil temperatures.

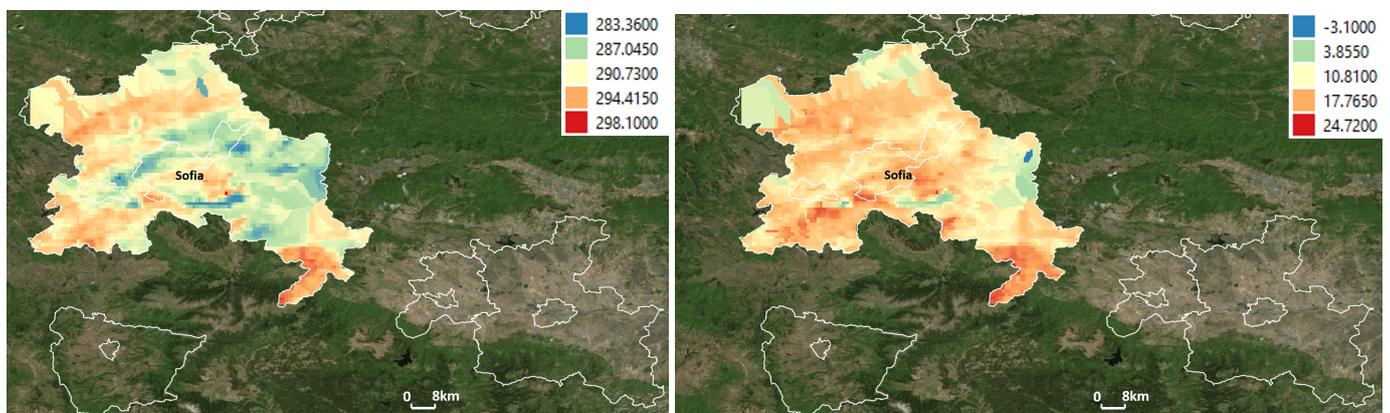


Figure 9. Map visualization of query results of Q5 (left) and Q6 (right).

**Q7** What is the maximum soil temperature inside the entity holding the maximum number of airports between two times, 15 March 2009 and 14 May 2009?

```

select ?urbnm ?airport_number (max(?val) as ?max_soil_temp) where {

```

```

?st1 dcb:hasDateTime ?dt;
dcb:hasSpatialRef ?sp. ?sp dcb:asFieldWKT ?gwkt. ?st1 dcb:hasField ?f.
?f dcb:hasFieldValue ?val;
dcb:hasVariableName ?var.
?urb2 dcb:area_name ?urbnm;
geo:hasGeometry ?urbgeom2. ?urbgeom2 geo:asWKT ?urb_wkt2.
{
select ?urb_name (count(?airname) as ?airport_number) where {
?urb dcb:area_name ?urb_name;
geo:hasGeometry ?urbgeom. ?urbgeom geo:asWKT ?urb_wkt.
?airport dcb:airport_name ?airname;
geo:hasGeometry ?geom. ?geom geo:asWKT ?air_wkt.
Filter(geof:sfContains(?urb_wkt, ?air_wkt) )
} group by ?urb_name order by desc(?airport_number) limit 1 }

FILTER (?dt >= "2009-03-15T00:00:00+01:00"^^xsd:dateTimeStamp && ?dt
<="2009-05-04T00:00:00+02:00"^^xsd:dateTimeStamp && ?var = 'LST_Day_1km' &&
geof:sfWithin(?gwkt, ?urb_wkt2) && ?urbnm = ?urb_name) } group by ?urbnm
?airport_number order by ?airport_number

```

Q7 uses a subquery to make a selection of the urban area containing the maximum number of airports and then returns that area for applying spatial and temporal aggregation to select the maximum soil temperature value. The resulting map shows the selected area along with the maximum value, which is 286.072 (see Figure 10).

**Q8 How does the maximum soil temperature evolve during the period between 15 March 2009 and 25 March 2009?**

```

select ?soil_temp_location ?time_stamp ?val_soil_temp where {

?st1 dcb:hasDateTime ?time_stamp;
dcb:hasSpatialRef ?sp. ?sp dcb:asFieldWKT ?soil_temp_location.

?st1 dcb:hasField ?f. ?f dcb:hasFieldValue ?val_soil_temp;
dcb:hasVariableName ?var.
{ select ?dt (max(?val) as ?max_soil_temp) where {
?st1 dcb:hasDateTime ?dt;
dcb:hasField ?f. ?f dcb:hasFieldValue ?val;
dcb:hasVariableName ?var.

FILTER (?dt >= "2009-03-15T00:00:00+01:00"^^xsd:dateTimeStamp && ?dt <=
"2009-03-25T00:00:00+02:00"^^xsd:dateTimeStamp && ?var = 'LST_Day_1km') }
group by ?dt }
Filter(?val_soil_temp = ?max_soil_temp && ?time_stamp = ?dt && ?var =
'LST_Day_1km')}} order by ?time_stamp

```

**Q9 When and where did the highest soil temperature value occur during the period between 15 March 2009 and 17 March 2009 and what are the entities and their types that this value falls within?**

```

select ?feature ?entityType ?soil_temp_location (?t as ?Time) (?val as ?VALUE) where
{

?feature rdf:type ?entityType;
geo:hasGeometry ?fgeom. ?fgeom geo:asWKT ?urban_area.
{
select ?soil_temp_location ?t ?val where {

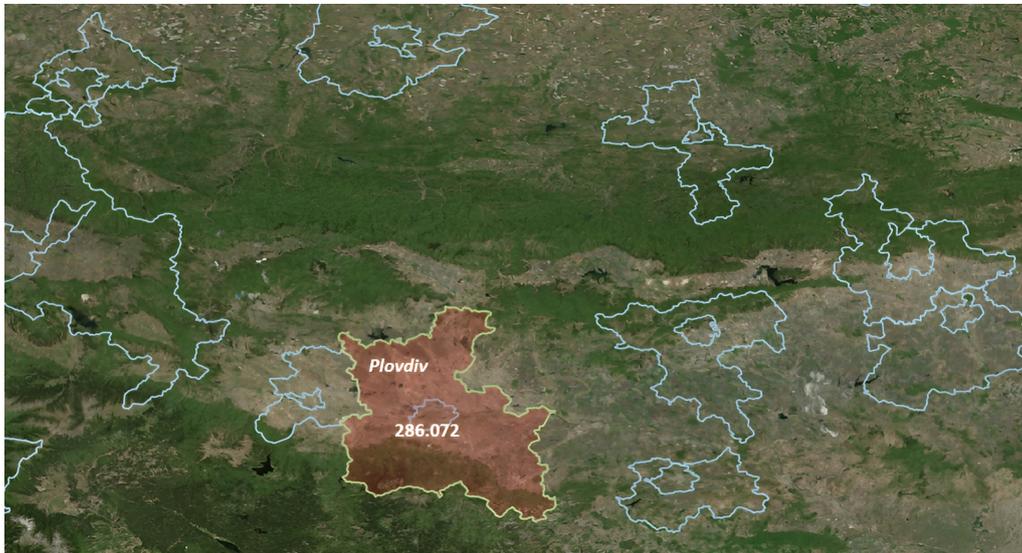
?st1 dcb:hasDateTime ?t;
dcb:hasSpatialRef ?sp. ?sp dcb:asFieldWKT ?soil_temp_location.
?st1 dcb:hasField ?f. ?f dcb:hasFieldValue ?val;
dcb:hasVariableName ?var.
{
select ?dt (max(?val) as ?max_val) where {
?st1 dcb:hasDateTime ?dt;
dcb:hasField ?f. ?f dcb:hasFieldValue ?val;
dcb:hasVariableName ?var.

```

```

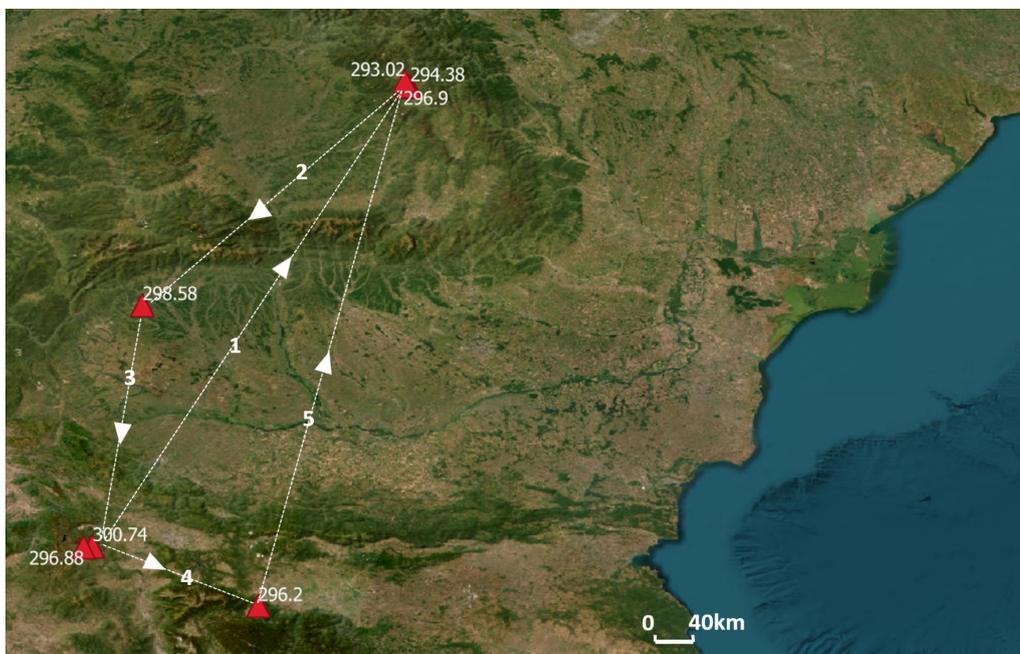
FILTER (?dt > "2009-03-14T00:00:00+01:00"^^xsd:dateTimeStamp && ?dt <
"2009-03-17T00:00:00+02:00"^^xsd:dateTimeStamp && ?var = 'LST_Night_1km') }
group by ?dt }
Filter(?val = ?max_val && ?t = ?dt && ?var = 'LST_Night_1km')
}} Filter(geof:sfWithin(?soil_temp_location, ?urban_area) && ?entityType not in
(geo:SpatialObject, geo:Feature )) }}

```



**Figure 10.** Map visualization of the query result for Q7.

Q8 uses a subquery to calculate the maximum value for each time. This sequence of values is then returned and used in the outer query to find the locations corresponding to each maximum value. Figure 11 shows the path that was drawn to track the maximum at each time.



**Figure 11.** Visualization of the query result for Q8. The numbers ranging from 1 to 5 delineate the chronological order of paths followed, beginning from the initial point and concluding at the final destination while tracking the maximum temperature. The directional arrows positioned along these pathways serve to illustrate the specific direction of movement.

Q9 involves two nested subqueries. The first selects the maximum value of the soil temperature for each time and returns this sequence to the outer subquery to find all locations corresponding to each maximum value in this sequence. We then use a spatial filter with a topological relationship to try to find the geometric features in which these locations are contained. Finally, using the reasoning capabilities of the system, the query can infer the entity type of each geometric feature that meets the spatial criteria. Figure 12 shows the result of this query.

SPARQL results		SQL translation				
feature_ID	entityType	soil_temp_location	Time			VALUE
ura-96	Urban Area	"POINT(23.04128199783415 42.83141391676648)"	"2009-03-16T00:00:00+01:00"xs:dateTimeStamp			"287.04"xs:decimal
ura-80	Urban Area	"POINT(24.767361214353123 42.10527024638906)"	"2009-03-16T00:00:00+01:00"xs:dateTimeStamp			"287.04"xs:decimal
ura-94	Urban Area	"POINT(27.802879836507177 43.75992549530744)"	"2009-03-16T00:00:00+01:00"xs:dateTimeStamp			"287.04"xs:decimal
ura-18	Urban Area	"POINT(23.73171368444174 47.6406277338248)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-13	Urban Area	"POINT(27.30291206344651 45.759796587549516)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-98	Urban Area	"POINT(24.326913414275868 42.31954214910928)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-80	Urban Area	"POINT(24.61260928459625 42.12907823556242)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-78	Urban Area	"POINT(27.064832171512858 43.28376571144028)"	"2009-03-16T00:00:00+01:00"xs:dateTimeStamp			"287.04"xs:decimal
ura-36	Urban Area	"POINT(23.50553778710477 42.79570193297644)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-96	Urban Area	"POINT(23.50553778710477 42.79570193297644)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-80	Urban Area	"POINT(25.017345100883457 41.97432630580559)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-36	Urban Area	"POINT(23.517441781701454 42.79570193297644)"	"2009-03-16T00:00:00+01:00"xs:dateTimeStamp			"287.04"xs:decimal
ura-96	Urban Area	"POINT(23.517441781701454 42.79570193297644)"	"2009-03-16T00:00:00+01:00"xs:dateTimeStamp			"287.04"xs:decimal
ura-80	Urban Area	"POINT(25.02924909548014 41.89099834362884)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-92	Urban Area	"POINT(26.112512603778253 43.74802150071076)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-99	Urban Area	"POINT(23.17222593839766 41.85528635983887)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-78	Urban Area	"POINT(27.052928176916176 43.236149733053566)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-35	Urban Area	"POINT(24.743553225159758 42.140982230159096)"	"2009-03-16T00:00:00+01:00"xs:dateTimeStamp			"287.04"xs:decimal
ura-80	Urban Area	"POINT(24.743553225159758 42.140982230159096)"	"2009-03-16T00:00:00+01:00"xs:dateTimeStamp			"287.04"xs:decimal
ura-80	Urban Area	"POINT(24.636417273789615 42.12907823556242)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-80	Urban Area	"POINT(24.826881187336536 41.98623030040227)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-18	Urban Area	"POINT(23.565057760088184 47.84299564196834)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-14	Urban Area	"POINT(23.255553900574437 45.02174892255542)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-62	Urban Area	"POINT(23.255553900574437 45.02174892255542)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-99	Urban Area	"POINT(23.279361889767802 41.902902338225516)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-92	Urban Area	"POINT(26.088704614584888 43.724213511517405)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-97	Urban Area	"POINT(25.660160809104315 41.93861432201555)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal
ura-13	Urban Area	"POINT(27.243392090463097 45.771700582146195)"	"2009-03-15T00:00:00+01:00"xs:dateTimeStamp			"286.42"xs:decimal

Figure 12. List of query results of Q9.

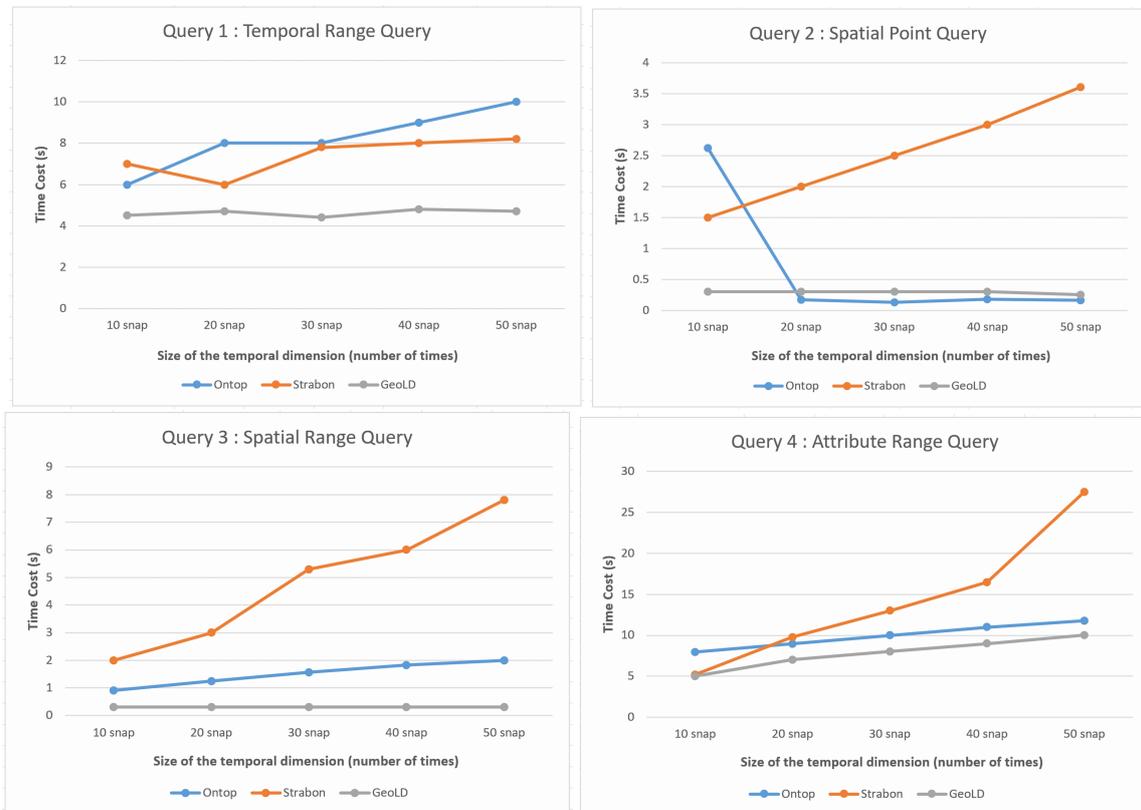
### 5.3. Performance Evaluation

This section evaluates the computational performance of our implemented framework. This experimentation consists of two parts:

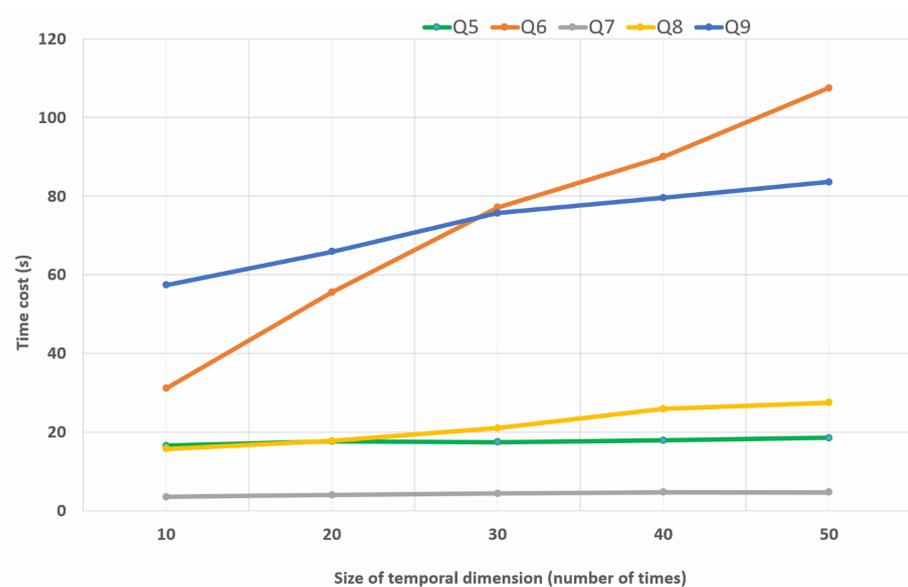
- In the first experiment, we compare our framework implemented on the basis of the Ontop system against GeoLD and Strabon. For this comparison, we use the four basic spatiotemporal queries formulated in Appendix A. We examine the runtime performance of each query and observe how the graphical curves evolve as the size of the temporal dimension (i.e., the number of time-stamped layers) increases for each system. These experiments have been conducted and are illustrated in Figure 13.
- The second experiment consists of examining, using the Ontop system, the evolution of the execution time of advanced geospatial queries reformulated in Section 5.2. Similarly, we examine the runtime cost as the size of the temporal dimension increases. However, in this evaluation, a comparison is made between and among these queries to see how the Ontop system behaves with each of them. This is illustrated in Figure 14.

For each query in the experiments, three separate consecutive calls are made to collect their average response times. For the first part of the experiment, we can clearly see from Figure 13 that for the three queries, Q2, Q3, and Q4, the framework implemented under the VKG system performs well, and scales quite well with the growing size of the temporal dimension. The performance of Ontop is very close to that of the GeoLD system, even though the GeoLD system uses an array-based database system (Rasdaman), which is considered the most efficient for storing and querying raster data cubes. Both Ontop and GeoLD significantly outperform the Strabon system for these queries. However, with respect to Q1, Ontop does not perform as well as GeoLD and Strabon. This may be due to several reasons. Indeed, the wider the temporal range of the query is, the less efficient the spatial index of the database can be. The reason is that all raster cells of the study area are requested, which leads to a high amount of requested spatiotemporal cells. Thus, the on-the-fly translation process of this output data into RDF format that is performed by the Ontop system becomes costly and has a significant impact on the performance and scalability of the system. In this case, a time range query with a spatial filter could be highly desirable to obtain a better response. In addition, a partitioning strategy for temporal data

in PostgreSQL could help improve the scalability of the system. Finally, improving the computational performance of the algorithm that translates geospatial data on the fly into RDF format could help reduce the cost of the query answer.



**Figure 13.** Graphs highlighting the evolution of computation time of four queries of the Ontop system depending on the size of temporal window in comparison with Geold and Strabon.



**Figure 14.** Graphs highlighting the evolution of computation time of five advanced queries on the Ontop system depending on the size of temporal window.

In the second part of the experiment, we can observe three main patterns: the first one shows that Ontop performs very well for three queries, Q5, Q7, and Q8. These

queries involve a time range with a spatial filter (Q5), a spatial and temporal aggregation (Q7), and a spatial-only aggregation (Q8). The second trend is related to the behavior of Ontop regarding the Q9 query, which involves spatial aggregation together with a spatial filter. In addition, it involves a reasoning operation to infer the entity types from selected instances. The execution time in this case becomes more expensive but the system still shows acceptable scalability. The last trend is related to query Q6. It is obvious that when the size of input data increases, the system shows poor scalability and the curve tends to have exponential growth. This is due to the fact that this query involves a binary operation between two data cubes. This type of query can be performed naturally and efficiently when the data cubes are encoded as multidimensional arrays. However, with a VKG system like Ontop and its relational databases, this type of query requires the development of optimal operators or functions that can efficiently handle such binary operations.

It is worth mentioning that, to perform such an experiment, we used the Ontop version 4.2.1, which is integrated with the Protégé software. This package enables the user to load the ontology, define the mapping, and reformulate SPARQL queries. The generic data model schema has been implemented in the PostgreSQL 13 software. Tests were performed on a Dell LAT 7520 with 3 GHz Max, 11th generation, Intel Core i7, and 32 GB of RAM. The operating system used was Windows 10.

## 6. Conclusions and Discussion

This paper proposed a framework to enable the semantic integration and advanced querying of raster data cubes based on VKG. In this framework, we defined a semantic representation model for raster data cubes as a module that extends the GeoSPARQL ontology. With such a model, we could combine the semantics of raster data cubes with feature-based models that involve geometries as well as spatial/topological relationships. This allowed us to formulate spatiotemporal queries using SPARQL in a natural way by using ontological concepts at an appropriate level of abstraction. In order to evaluate the expressivity of any semantic representation model, we confronted it with a set of scenarios that, in our case, were a set of geospatial queries. We further presented a taxonomy within which each query can find its semantic positioning. We then took a sample of scenarios and classified them according to this taxonomy. Furthermore, we explained the system architecture that enabled us to implement our proposed framework. We applied our approach to forest fire data and formulated nine basic and advanced spatiotemporal queries to retrieve information from only raster and from both raster and vector layers, respectively. To evaluate the computational performance of our implemented framework, we designed an experiment with two parts: In the first one, we compared our framework implemented using the Ontop system against GeoLD and Strabon. In the second one, we examined the runtime cost as the size of the temporal dimension increased, and made a comparison between and among these queries to see how the system behaved with each query.

In conclusion, the proposed framework expands the ability to express enhanced geospatial queries and enables the intelligent integration of a semantic data cube. Furthermore, the implementation of this framework shows reasonable performance, especially with respect to the complexity of the formulated queries. However, for some queries, the system shows weak performance and scalability. This mainly concerns queries involving binary operations on data cubes (e.g., arithmetic, topological, etc.). In addition, other operations are not yet properly supported. This includes rolling or focal operations, as well as operations that manipulate the shape and dimensions of the data cube. For future work, this part still needs to be developed and supported efficiently at the SPARQL level in order to enable the full manipulation and integration of the data cube. Another challenge for future work is to develop an approach that not only handles static geometric features but also evolving entities and to combine them with raster data cubes, which can help develop a powerful and intelligent analytical query system for dynamic GIS.

**Author Contributions:** Conceptualization and methodology, Younes Hamdani, Guohui Xiao, Linfang Ding, and Diego Calvanese; implementation, Younes Hamdani and Guohui Xiao; investigation, Younes Hamdani; writing—original draft preparation, Younes Hamdani; writing—review and editing, Younes Hamdani, Linfang Ding, Guohui Xiao, and Diego Calvanese; supervision, Diego Calvanese. All authors have read and agreed to the published version of the manuscript.

**Funding:** The funding for this research was provided through grants from the Kempe Foundation specifically designated for the advancement of artificial intelligence and software development.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data used for the experimentation are available in [46,47]. Furthermore, the reader has the opportunity to access the ontology in OWL format, accompanied by the file containing the mapping between the ontology and the table schemas. This resource can be found at the following link: <https://github.com/ynsHamdani/Raster-DataCube-VKG> (Raster Data Cube) (accessed on 6 August 2023).

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

## Abbreviations

The following abbreviations are used in this manuscript:

VKG	Virtual knowledge graphs
OGC	Open Geospatial Consortium
OBDA	Ontology-based data access
RDF	Resource Description Framework
EO	Earth Observation
WCPS	Web Coverage Processing Service
ETL	Extract, transform, load
TRQ	Time range query
FRQ	Field range query
SPQ	Spatial point query
SRQ	Spatial range query

## Appendix A

**Q1** What are the raster cells (i.e., locations) and their values related to the variable `LST_Day_1km` between two times, 15 March 2009 and 25 March 2009?

```
select ?raster_cell ?LST_Day_value ?time_stamp where {
  ?f dcb:hasSTL ?stl. ?stl dcb:hasDateTime ?time_stamp;
  dcb:hasSpatialRef ?sp. ?sp dcb:asFieldWKT ?raster_cell.
  ?f dcb:hasFieldValue ?LST_Day_value; dcb:hasVariableName ?var.

  FILTER (?time_stamp > "2009-03-15T00:00:00+02:00"^^xsd:dateTimeStamp &&
    ?time_stamp < "2009-03-25T00:00:00+02:00"^^xsd:dateTimeStamp && ?var = 'LST_Day_1km'
  ) }
```

**Q2** What are the raster cells and their values related to the variable `LST_Night_1km` where these values are superior to value 304 between two times, 15 March 2009 and 25 March 2009?

```
select ?raster_cell ?LST_Night_value ?time_stamp where {
  ?stl dcb:hasDateTime ?time_stamp;
  dcb:hasSpatialRef ?sp. ?sp dcb:asFieldWKT ?raster_cell;
  ?stl dcb:hasField ?f.
  ?f dcb:hasFieldValue ?LST_Night_value;
  dcb:hasVariableName ?var.

  FILTER (?dt > "2009-03-15T00:00:00+01:00"^^xsd:dateTimeStamp && ?time_stamp <
    "2009-03-25T00:00:00+02:00"^^xsd:dateTimeStamp && ?var = 'LST_Night_1km' &&
    ?LST_Night_value > 304 )
```

```
}

```

**Q3 What are the raster cells and their values related to the variable Lai\_500m at the points between two times, 15 March 2009 and 25 March 2009?**

```
select ?raster_cell ?Lai_value ?time_stamp where {
  ?st1 dcb:hasDateTime ?time_stamp;
  dcb:hasSpatialRef ?sp. ?sp dcb:asFieldWKT ?raster_cell;
  ?st1 dcb:hasField ?f.
  ?f dcb:hasFieldValue ?Lai_value;
  dcb:hasVariableName ?var.

  FILTER (?time_stamp >= "2009-03-14T00:00:00+01:00"^^xsd:dateTimeStamp && ?time_stamp
    < "2009-03-18T00:00:00+02:00"^^xsd:dateTimeStamp && ?var = 'LST_Day_1km' &&
    geof:ehEquals(?raster_cell, "POINT(27.362432036429922
    41.914806332822195)"^^geo:wktLiteral))
}
```

**Q4 What are the raster cells and their values related to the variable Lai\_500m inside the polygon P between two times, 15 March 2009 and 25 March 2009?**

```
select ?raster_cell ?Lai_value ?time_stamp where {
  ?st1 dcb:hasDateTime "2009-03-14T00:00:00+01:00"^^xsd:dateTimeStamp;
  dcb:hasSpatialRef ?sp.?sp dcb:asFieldWKT ?raster_cell;
  ?st1 dcb:hasField ?f. ?f dcb:hasFieldValue ?Lai_value;
  dcb:hasVariableName ?Lai_500m'.

  FILTER ( geof:sfContains("POLYGON((25.857121 44.215505,25.857121 44.612258,26.463456
    44.612258,26.463456 44.215505,25.857121
    44.215505))"^^geo:wktLiteral,?raster_cell ))}
```

## References

- Lu, M.; Appel, M.; Pebesma, E. Multidimensional arrays for analysing geoscientific data. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 313. [\[CrossRef\]](#)
- Baumann, P.; Misev, D.; Merticariu, V.; Huu, B.P. Array databases: Concepts, standards, implementations. *J. Big Data* **2021**, *8*, 28. [\[CrossRef\]](#)
- Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; de Melo, G.; Gutierrez, C.; Kirrane, S.; Gayo, J.E.L.; Navigli, R.; Neumaier, S.; et al. Knowledge Graphs. *ACM Comput. Surv.* **2021**, *54*, 1–37. [\[CrossRef\]](#)
- Andrejev, A.; Risch, T. Scientific SPARQL: Semantic Web queries over scientific data. In Proceedings of the 28th IEEE International Conference on Data Engineering Workshops, Arlington, VA, USA, 1–5 April 2012; pp. 5–10.
- Masmoudi, M.; Lamine, S.B.A.B.; Zghal, H.B.; Archimede, B.; Karray, M.H. Knowledge hypergraph-based approach for data integration and querying: Application to Earth Observation. *Future Gener. Comput. Syst.* **2021**, *115*, 720–740. [\[CrossRef\]](#)
- Almobydeen, S.B.; Viqueira, J.R.; Lama, M. GeoSPARQL query support for scientific raster array data. *Comput. Geosci.* **2022**, *159*, 105023. [\[CrossRef\]](#)
- Kyzirakos, K.; Karpathiotakis, M.; Bereta, K.; Garbis, G.; Nikolaou, C.; Smeros, P.; Giannakopoulou, S.; Dogani, K.; Koubarakis, M. The spatiotemporal RDF store strabon. In Proceedings of the International Symposium on Spatial and Temporal Databases, Munich, Germany, 21–23 August 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 496–500.
- Kyzirakos, K.; Savva, D.; Vlachopoulos, I.; Vasileiou, A.; Karalis, N.; Koubarakis, M.; Manegold, S. GeoTriples: Transforming geospatial data into RDF graphs using R2RML and RML mappings. *J. Web Semant.* **2018**, *52*, 16–32. [\[CrossRef\]](#)
- Tran, B.H.; Aussenac-Gilles, N.; Comparot, C.; Trojahn, C. Semantic Integration of Raster Data for Earth Observation on Territorial Units. *ISPRS Int. J. Geo-Inf.* **2022**, *11*, 149. [\[CrossRef\]](#)
- Xiao, G.; Lanti, D.; Kontchakov, R.; Komla-Ebri, S.; Güzel-Kalaycı, E.; Ding, L.; Corman, J.; Cogrel, B.; Calvanese, D.; Botoeva, E. The Virtual Knowledge Graph System Ontop. In Proceedings of the 19th International Semantic Web Conference (ISWC), Part II, Athens, Greece, 2–6 November 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 259–277.
- Brown, P.G. Overview of SciDB: Large scale array storage, processing and analysis. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, Indianapolis, IN, USA, 6–10 June 2010; pp. 963–968.
- Baumann, P.; Dehmel, A.; Furtado, P.; Ritsch, R.; Widmann, N. The multidimensional database system RasDaMan. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, WA, USA, 1–4 June 1998; pp. 575–577.
- Baumann, P.; Mazzetti, P.; Ungar, J.; Barbera, R.; Barboni, D.; Beccati, A.; Bigagli, L.; Boldrini, E.; Bruno, R.; Calanducci, A.; et al. Big data analytics for earth sciences: The EarthServer approach. *Int. J. Digit. Earth* **2016**, *9*, 3–29. [\[CrossRef\]](#)

14. Mišev, D.; Baumann, P.; Bellos, D.; Wiehle, S. BigDataCube: A scalable, federated service platform for Copernicus. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 4103–4112.
15. Baumann, P. Datacube standards and their contribution to analysis-ready data. In Proceedings of the 2018 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Valencia, Spain, 22–27 July 2018, pp. 2051–2053.
16. Gao, F.; Yue, P.; Cao, Z.; Zhao, S.; Shangguan, B.; Jiang, L.; Hu, L.; Fang, Z.; Liang, Z. A multi-source spatio-temporal data cube for large-scale geospatial analysis. *Int. J. Geogr. Inf. Sci.* **2022**, *36*, 1853–1884. [\[CrossRef\]](#)
17. Baumann, P. The OGC Web Coverage Processing Service (WCPS) standard. *Geoinformatica* **2010**, *14*, 447–479. [\[CrossRef\]](#)
18. Aiordăchioaie, A.; Baumann, P. Petascope: An open-source implementation of the OGC WCS geo service standards suite. In Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany, 30 June–23 July 2020; Springer: Berlin/Heidelberg, Germany, 2010; pp. 160–168.
19. Brizhinev, D.; Toyer, S.; Taylor, K.; Zhang, Z. Publishing and using Earth observation data with the RDF data cube and the discrete global grid system. *W3C Work. Group Note OGC Discuss. Pap. W3C* **2017**, 20170928, 16–125.
20. Atkinson, R. *QB4st: RDF Data Cube Extensions for Spatio-Temporal Components*; W3C Working Group: Wakefield, MA, USA, 2017.
21. Zinke, C.; Ngomo, A.C.N. Discovering and Linking Spatio-Temporal Big Linked Data. In Proceedings of the IGARSS 2018–2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 22–27 July 2018; pp. 411–414.
22. Andrejev, A.; Toor, S.; Hellander, A.; Holmgren, S.; Risch, T. Scientific analysis by queries in extended SPARQL over a scalable e-Science data store. In Proceedings of the 9th IEEE International Geoscience on e-Science, Beijing, China, 22–25 October 2013; pp. 98–106.
23. Homburg, T.; Staab, S.; Janke, D. Geosparql+: Syntax, semantics and system for integrated querying of graph, raster and vector data. In Proceedings of the 19th International Semantic Web Conference (ISWC), Part I, Athens, Greece, 2–6 November 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 258–275.
24. Arocena, J.; Lozano, J.; Quartulli, M.; Olaizola, I.; Bermudez, J. Linked Open Data for raster and vector geospatial information processing. In Proceedings of the 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Milan, Italy, 26–31 July 2015; pp. 5023–5026.
25. Homburg, T.; Prudhomme, C.; Würriehausen, F.; Karmacharya, A.; Boochs, F.; Roxin, A.; Cruz, C. Interpreting heterogeneous geospatial data using Semantic Web technologies. In Proceedings of the 16th International Conference on Computational Science and Its Applications (ICCSA), Part III, Beijing, China, 4–7 July 2016; Springer: Berlin/Heidelberg, Germany, 2016, pp. 240–255.
26. Alirezaie, M.; Kiselev, A.; Längkvist, M.; Klügl, F.; Loutfi, A. An ontology-based reasoning framework for querying satellite images for disaster monitoring. *Sensors* **2017**, *17*, 2545. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Arenas, H.; Aussenac-Gilles, N.; Comparot, C.; Trojahn, C. Semantic integration of geospatial data from earth observations. In Proceedings of the European Knowledge Acquisition Workshop, Bologna, Italy, 19–23 November 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 97–100.
28. Kyzirakos, K.; Karpathiotakis, M.; Koubarakis, M. Strabon: A semantic geospatial DBMS. In Proceedings of the The Semantic Web–ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, 11–15 November 2012; Proceedings, Part I 11; Springer: Berlin/Heidelberg, Germany, 2012; pp. 295–311.
29. Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Rosati, R. Linking data to ontologies. *J. Data Semant.* **2008**, *10*, 133–173. [\[CrossRef\]](#)
30. Kontchakov, R.; Rodriguez-Muro, M.; Zakharyashev, M. Ontology-Based Data Access with Databases: A Short Course. In Proceedings of the Reasoning Web: Semantic Technologies for Intelligent Data Access—9th International Summer School Tutorial Lectures (RW), Mannheim, Germany, 30 July–2 August 2013; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8067, pp. 194–229. [\[CrossRef\]](#)
31. Xiao, G.; Calvanese, D.; Kontchakov, R.; Lembo, D.; Poggi, A.; Rosati, R.; Zakharyashev, M. Ontology-Based Data Access: A Survey. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), IJCAI Org., Stockholm, Sweden, 13–19 July 2018; pp. 5511–5519. [\[CrossRef\]](#)
32. Rodriguez-Muro, M.; Rezk, M. Efficient SPARQL-to-SQL with R2RML mappings. *J. Web Semant.* **2015**, *33*, 141–169. [\[CrossRef\]](#)
33. Calvanese, D.; Cogrel, B.; Komla-Ebri, S.; Kontchakov, R.; Lanti, D.; Rezk, M.; Rodriguez-Muro, M.; Xiao, G. Ontop: Answering SPARQL queries over relational databases. *Semant. Web* **2017**, *8*, 471–487. [\[CrossRef\]](#)
34. Battle, R.; Kolas, D. Geosparql: Enabling a geospatial Semantic Web. *Semant. Web J.* **2011**, *3*, 355–370. [\[CrossRef\]](#)
35. Bereta, K.; Koubarakis, M. Ontop of geospatial databases. In Proceedings of the The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, 17–21 October 2016; Proceedings, Part I 15; Springer: Berlin/Heidelberg, Germany, 2016; pp. 37–52.
36. Bereta, K.; Xiao, G.; Koubarakis, M. Ontop-spatial: Ontop of geospatial databases. *J. Web Semant.* **2019**, *58*, 100514. [\[CrossRef\]](#)
37. Ding, L.; Xiao, G.; Calvanese, D.; Meng, L. Consistency assessment for open geodata integration: An ontology-based approach. *Geoinformatica* **2021**, *25*, 733–758. [\[CrossRef\]](#)
38. Ding, L.; Xiao, G.; Calvanese, D.; Meng, L. A framework uniting ontology-based geodata integration and geovisual analytics. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 474. [\[CrossRef\]](#)
39. Masmoudi, M. Knowledge Hypergraph Based-Approach for Multi-Source Data Integration and Querying: Application for Earth Observation Domain. Ph.D. Thesis, INPT, Toulouse, France 2020.
40. Gebbert, S.; Pebesma, E. A temporal GIS for field based environmental modeling. *Environ. Model. Softw.* **2014**, *53*, 1–12. [\[CrossRef\]](#)

41. Galton, A. Fields and objects in space, time, and space-time. *Spat. Cogn. Comput.* **2004**, *4*, 39–68. [[CrossRef](#)]
42. Goodchild, M.F.; Yuan, M.; Cova, T.J. Towards a general theory of geographic representation in GIS. *Int. J. Geogr. Inf. Sci.* **2007**, *21*, 239–260. [[CrossRef](#)]
43. Hamdani, Y. Spatio-Temporal Hybrid Model for Temporal GIS: Application to Marine Geomorphology. Ph.D. Thesis, Université de Bretagne Occidentale, Brest, France 2020.
44. Hamdani, Y.; Thibaud, R.; Claramunt, C. A hybrid data model for dynamic GIS: Application to marine geomorphological dynamics. *Int. J. Geogr. Inf. Sci.* **2021**, *35*, 1475–1499. [[CrossRef](#)]
45. Xiao, G.; Ding, L.; Cogrel, B.; Calvanese, D. Virtual Knowledge Graphs: An overview of systems and use cases. *Data Intell.* **2019**, *1*, 201–223. [[CrossRef](#)]
46. European Commission, Eurostat/GISCO. Nomenclature of Territorial Units for Statistics. 2021. Available online: <https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts> (accessed on 17 April 2023).
47. Prapas, I.; Kondylatos, S.; Papoutsis, I. A Datacube for the Analysis of Wildfires. 2021. Available online: [https://zenodo.org/record/4943354#.Y\\_-QwoTMKMo](https://zenodo.org/record/4943354#.Y_-QwoTMKMo) (accessed on 17 April 2023).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.