Received April 6, 2020, accepted April 28, 2020, date of publication May 4, 2020, date of current version May 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992202

Approximate Query Answering Based on Topological Neighborhood and Semantic Similarity in OpenStreetMap

ANNA FORMICA, MAURO MAZZEI, ELAHEH POURABBAS, AND MAURIZIO RAFANELLI
National Research Council, Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", I-00185 Rome, Italy
Corresponding author: Anna Formica (anna.formica@iasi.cnr.it)

ABSTRACT In this paper we focus on a pictorial query language, referred to as *Geographical Pictorial Query Language* (GeoPQL), and we revise its formal semantics by considering the polygon-polyline, polyline-polyline, and polygon-polygon topological relationships. This work proposes the *Approximate Answering Engine* (AAE) within a Distributed System, referred to as GeoPQLJSON (GeoPQLJ). The AAE provides approximate answers to query with empty results by following two directions: the *Operator Conceptual Neighborhood* (OCN) graph, and the *OpenStreetMap* (OSM) attribute hierarchy, giving maximum flexibility to the user choices. According to the former, the geo-operators of the queries can be replaced with the ones labeling the adjacent nodes of the OCN graph. By following the latter, the system evaluates the OSM attribute semantic similarity according to the information content approach, and proposes possible attribute replacements to the user. Note that the presence of OSM attributes allows the quick and direct access to large amount of geographical data, without requiring in our case the use of the topological elements. The functionalities of the Distributed GeoPQLJ System are illustrated by several query examples.

INDEX TERMS Pictorial query language, operator conceptual neighborhood, openstreetmap, semantic similarity, approximate answering.

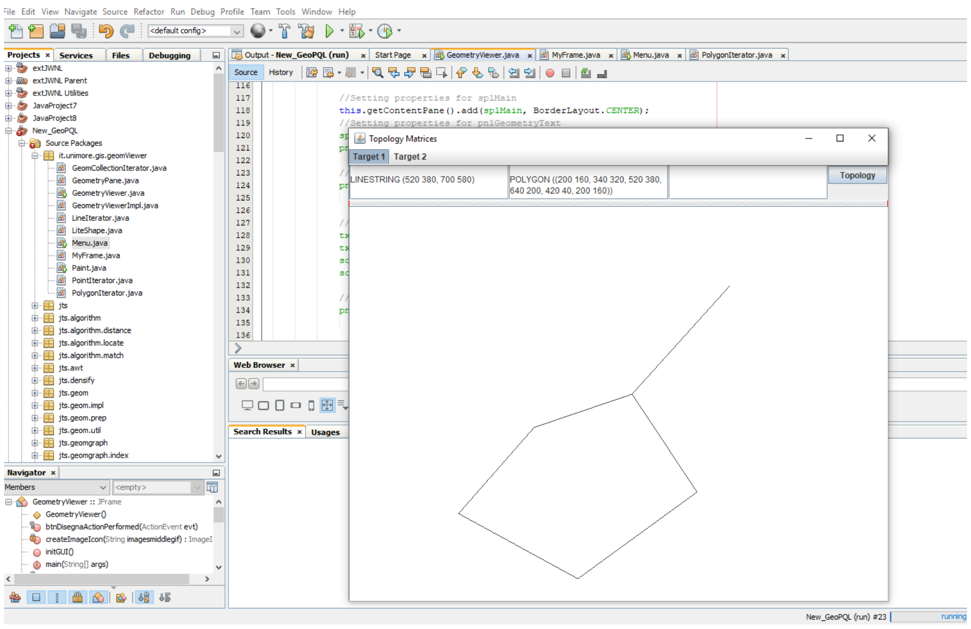
I. INTRODUCTION

Distributed Geographic Information Systems (GIS) have significantly enriched the expressiveness of geographic data models thanks to the contribution of users. OpenStreetMap (OSM) [3] is an example of distributed environment which allows a new way of collecting geographic information through the crowd rather than organizations, and brought free access to a plethora of geographic information.

In general, in a world-wide distributed GIS, the presence of native data models for querying geographic data is a key requirement [10]. Geographic queries indeed can be better expressed by using graphical metaphors in query languages which are powerful to express the user's mental model of the query [34]. In GIS, geographic query languages should satisfy two basic requirements: they must be powerful and easy to use at the same time. Powerful, because they have to retrieve information about complex database schemas, keeping track of several relations existing among data. Easy to use, because the access to the stored information should not be limited to experts, but should be conceived for non-specialized end-users [11], [33]. These two basic requirements find a common solution in the development of advanced visual geographic query languages [20].

In this paper, we concentrate on pictorial query languages, i.e., visual languages where queries are formulated by free-hand drawing [25]. In particular, we focus on the *Geographical Pictorial Query Language* (GeoPQL) [23], [24], [28], which is a pictorial query language that provides drawing facilities to formulate queries and correctly interprets their syntax and semantics. In the context of GIS, one of the main challenges regards the possibility of providing approximate answers to queries with empty results by relying on both topological neighborhood and semantic similarity approaches. While in the context of distributed GIS, one more challenge concerns the query response time due to the access to large amount of data. In this work we address both these problems and propose an integrated approach for providing approximate answers to queries with empty results based on OSM. It is a digital archive of geospatial data whose

The associate editor coordinating the review of this manuscript and approving it for publication was Waleed Alsabhan.



```
//Setting properties for splMain
this.getContentPane().add(splMain, BorderLayout.CENTER);
//Setting properties for pnlGeometryText
```

LINESTRING (520 380, 700 580)	POLYGON ((200 160, 340 320, 520 380, 640 200, 420 40, 200 160))

FIGURE 1. **Pictorial formulation of *highway TCH amenity* query.**

physical characteristics are represented on the surface that are referred to as "features", e.g., roads, buildings, parks, etc.. These basic features represent the database structure, and each feature is associated with a set of attributes representing geographical characteristics.

OSM is a potential alternative to commercial data and aims at generating and maintaining a free editable map database of the world in a collaborative manner without restrictive rules of the traditional copyright and license commitments [12]. In this perspective, we exploit the large amount of data associated with attributes in OSM and provide approximate answers to queries with empty results.

In this paper we have revised the formal semantics of the GeoPQL operators, and we focus on the polygon-polygon, polyline-polyline and polygon-polyline topological relationships. Then, in order to elaborate geographical queries in the distributed environment, we have associated with the revised GeoPQL operators the GeoPQLJSON (GeoPQLJ for short) functions which conform to GeoJSON format specification [1], in line with [26]. In particular, in this work we extend the GeoPQLJ functions related to polygon-polyline relationship, introduced in the mentioned paper, to the polygon-polygon, and polyline-polyline ones. These functions are based on the simple and text based GeoJSON format specification. Since GeoJSON is a JavaScript Object Notation (JSON) encoding [18], [41], the parsing and data interchanging for web services are flexible. For this reason it is one of the most popular encodings for transferring data to client-side map visualization.

The main contribution of this paper is the definition of the *Approximate Answering Engine* (AAE) within a system, referred to as *GeoPQLJ Distributed System*, which provides approximate answers, by following two different directions: the topological neighborhood based on the *Operator Conceptual Neighborhood* (OCN) graph, and the OSM *attribute hierarchy*. In the case of empty answers, according to the former direction, the geo-operators of the queries can be replaced with the geo-operators labeling the adjacent nodes of the OCN graph, on the basis of the user preferences; according to the latter direction, by keeping the same geo-operators, the attributes of the query can be replaced by accessing the list of OSM attributes, giving maximum flexibility to the user choices. In particular, the WordNet similarity engine can be used in order to evaluate OSM attribute similarity [7]. Among the similarity metrics that can be elaborated by WordNet, we selected the semantic similarity measure proposed by Lin in [36], which has been extensively experimented in the literature and shows higher correlation with human judgment.

In order to clarify the problem addressed in this paper, suppose the user is interested in the number of highways touching some amenities where it is possible to have a coffee break during a trip in Amsterdam. Such a query can be pictorially defined in our system as shown in Figure 1, where the features *highway* and *amenity* are associated with polyline and polygon respectively, and *highway* is the target of the query (see in Figure 1 the top left side where *Target 1* is associated with *highway*). This query, as shown in detail in the experiment presented in Section V of this work, is translated according to the GeoJSON format specification and, by selecting the feature *amenity* and the related attribute *cafe*, is submitted to OSM. As described in the experiment, in this case the system provides an empty answer (see Figure 8), i.e., there are no highways touching an amenity with a coffee shop in Amsterdam, in the selected bounding box. In order to have approximate non-empty answers to this query, the user is provided with different possible alternatives, on the basis of his/her preferences. One of them is for instance the replacement of the touch geo-operator with pass-through, because of their adjacency in the OCN graph (see Figure 2). By replacing the touch geo-operator with pass-through, the answer to the query is non-empty and in particular the systems returns 3 highways, among the 52425, which pass through an amenity with a coffee shop in Amsterdam (see Figure 9). As described in Section V, we will see that another possibility, depending on the user preferences, is the modification of the query attributes by replacing them with the most similar ones in OSM, if there are any, according to the formal semantics of Lin [36].

Overall, in this paper we present an integrated approach for approximate query answering in distributed GIS, whose benefits can be summarized as follows: the user can express his/her queries pictorially by using drawing facilities; in the case of empty results, the proposed system provides possible approximate answers by following two different approaches, i.e., the topological neighborhood of geo-operators or the semantic similarity of OSM attributes; the presence of attributes in OSM allows the quick and direct access to large amount of geographical data, without requiring in our proposal the use of the topological elements such as point, polygon or polyline.

The paper is structured as follows. In the next section, the related work is given. In Section III, the revised GeoPQL operators are presented, and the OCN graph for polygon-polyline, polyline-polyline, and polygon-polygon relationships are given. In Section IV, the GeoPQLJ Distributed System is described. In Section V, the main functionalities of the GeoPQLJ Distributed System are illustrated. Finally, in Section VI the conclusion is given and in the Appendix the GeoPQLJ functions are illustrated.

II. RELATED WORK

In order to access and share data in distributed GIS [16], [35], the current Web-based systems adopt traditional browser/server architectures [35]. In these environments, the browser usually sends a request to the server, which processes it and returns the result to the browser. In this activity, spatial query languages become guidelines for Web-based GIS.

In [9] a framework, called XQuery for OpenStreetMap (XOSM), for integrating and querying OSM and Linked Geo Open Data (LGOD) resources is presented. It is equipped with a Web Tool and a XQuery based library that allows the definition of queries combining OSM layers created from LGOD. This library is based on the spatial operators introduced in [21] and [22]. The preliminary version of XOSM is given in [8], for the retrieval of layers with boolean spatial/keywords operators, in line with [17]. With regard to the implementation, the authors use the BaseX XQuery processor, and the PostGIS system over PostgreSQL. In [31], the logical and topological inconsistencies in the *Volunteered Geographic Information* (VGI), with a focus on OSM, is addressed. In the mentioned work, it has been demonstrated that the parameters such as direction, distance, and topological relationships between objects can directly affect human comprehension and analysis results. Therefore, by considering these relationships, the spatial similarity in multi-representation is used to build a framework to determine the probable inconsistencies in OSM, and fulfill the spatial quality assurance in VGI. In [40], the problem of computing and representing topological relations solely from geometries is considered and, to solve such a problem, ontologies and multi-layered topological relations, as well as a dataset of topologically linked places derived from DBpedia and OSM have been used. Since DBpedia places are only represented as point coordinates, in the mentioned work, OSM is used to match as many places from DBpedia with their corresponding polygon or polyline geometries in OSM.

With respect to the mentioned papers, in our proposal, we also use OSM but we address attributes in order to directly access large amount of data and, furthermore, in the case of queries with empty answers, we give the possibility to the users to select semantically similar attributes. In addition, we adopt a pictorial approach in order to formulate geographic queries.

Open Geospatial Consortium (OGC) GeoSPARQL standard [2], [15], is the genesis of a significant amount of work on the combination of the Resource Description Framework (RDF) and the Web Ontology Language (OWL) with geospatial data for representing and querying data on the Semantic Web. In order to facilitate querying in GeoSPARQL, for instance in [29] and [42], a graphical geospatial query tool, called GeoQuery, is presented. It includes a map-based user interface to search functions and geospatial operators enabling queries against GeoSPARQL. In our work we adopt an inherently different approach with respect to the mentioned papers for the following reasons. First of all, in GeoPQLJ we address the user's mental model and provide him/her with a drawing environment for easily expressing pictorial queries. Whereas in the mentioned papers the query formulation by means of the graphical interface is performed according to a procedural paradigm. In fact, in our approach, the query is formulated in a non-procedural way and therefore the order in the query specification is not relevant. Furthermore, as already mentioned, in our proposal the system provides approximate answers by exploiting both topological neighborhood and semantic similarity. Note that the topic of approximate query answering has been extensively investigated in the literature in different contexts, for instance, in relational databases [19], graph modeled-data [37], aggregate databases [39], and GIS [13]. Our proposal falls under the area of approximate query answering in GIS.

In [38], the Overpass API (or OSM3S) is proposed which is an extension of the API to select parts of an OSM layer. It has a proper query language which can be specified by an Extensible Markup Language (XML) template. However, OSM3S facilities (i.e., query composition and filtering) cannot be combined with spatial topological operators (e.g., touch and cross), which indeed are at the basis of our proposed approach.

With regard to the Geography Markup Language (GML), in [30] the authors propose GQL, which is a query language conceived to support spatial queries over GML documents. In particular, they extend the underlying data model, the algebra, and the semantics of XQuery. Furthermore, in [32], a similar approach has been proposed, by also addressing the performance problem of manipulating large GML documents. With respect to the aforementioned papers, our approach can be used in a general distributed system thanks to GeoJSON (and its functions) which, by making use of spatial operators, allows the access to geo-spatial repositories in an efficient way.

Regarding the semantic similarity in [14] the authors present a Semantic Network based on OSM and propose an approach to compute semantic similarity of geographic classes in this Network. However, this work differs from our proposal because it does not rely on the information content approach, but on co-citation algorithms that compute the semantic similarity of concepts in a graph of inter-linked objects based on the intuition that similar objects are referenced together. Furthermore, in [27] a method for measuring the semantic similarity of geographic classes organized as PartOf hierarchies has been investigated. In particular, the proposed method takes into account both the concept similarity within the PartOf hierarchy (through the information content approach) and the tuple similarity (through the sets of typed attributes). However, in the mentioned paper ISA hierarchies are not addressed.

Note that, to the best of our knowledge, in the literature, there are no proposals integrating at the same time the different characteristics of the GeoPQLJ Distributed System presented in this paper, that are: a non-procedural declarative pictorial query language with related drawing facilities; approximate answers to query with empty results obtained by replacing topological geo-operators or OSM attributes on the basis of the OCN graph and the widely experimented semantic similarity of Lin [36], respectively; quick access to large amount of geographical data in OSM. Therefore, comparisons with other proposals have not been presented.

III. GeoPQL OPERATORS

In this paper, we focus on GeoPQL which is based on the notion of *Symbolic Graphical Objects* ($\mathcal{SGO}$) [24], [28]. It has been defined to graphically represent the spatial configurations of geographic entities (i.e., *point*, *polyline*, and *polygon*), and the spatial relationships between $\mathcal{SGO}$.[1]

Definition ($\mathcal{SGO}$): Given a GIS, a *Symbolic Geographical Object* ($\mathcal{SGO}$) is a pair $\langle$*geometric_type*, $\Lambda\rangle$ where:

- *geometric_type* can be a *point*, a *polyline* or a *polygon*;
- Λ is an ordered set of pairs of coordinates, which defines the spatial extent and position of the $\mathcal{SGO}$ with respect to the coordinate reference system of the working area.

The GeoPQL algebra consists of a set of binary geo-operators, which are *logical* (Geo-union, Geo-any, Geo-alias), *metrical* (Geo-difference, and Geo-distance), and *topological* (Geo-disjunction, Geo-touch, Geo-inclusion,[2] Geo-intersect, Geo-cross, Geo-pass-through, Geo-overlap, Geo-equal). Our focus, as mentioned in the Introduction, is on the polygon-polyline, polyline-polyline, and polygon-polygon topological relationships. Therefore, in each of the above mentioned cases, we consider the related set of the topological operators. In particular, in the case of:

- polygon-polyline topological relationships, they are:
  - Geo-disjunction (DSJ)
  - Geo-inclusion (INC)
  - Geo-touch (TCH)
  - Geo-intersect (INT)
  - Geo-pass-through (PTH)
- polyline-polyline topological relationships, they are:
  - Geo-disjunction (DSJ)
  - Geo-touching (TCH)
  - Geo-inclusion (INC)
  - Geo-crossing (CRS)
  - Geo-equality (EQL)
- polygon-polygon topological relationships, they are:
  - Geo-disjunction (DSJ)
  - Geo-touching (TCH)
  - Geo-overlapping (OVL)
  - Geo-inclusion (INC)
  - Geo-equality (EQL).

In Table 1 the above geo-operators and some related configurations are shown.

Below we assume that, for a given $\mathcal{SGO}$, the subscripts i, b, and e denote respectively, the *interior*, *boundary*, and *exterior* points of the $\mathcal{SGO}$. Furthermore, for any $\mathcal{SGO}$, if $\mathcal{P}$ is a polygon and $\mathcal{L}$ is a polyline, the following holds:

$$\mathcal{P} = \mathcal{P}_i \cup \mathcal{P}_b \text{ and } \mathcal{L} = \mathcal{L}_i \cup \mathcal{L}_b$$

Note that in the definitions below, the order of the operands is not relevant. Furthermore, in this paper we focus on pictorial configurations representing only one geo-operator, i.e., combined geo-operators are not addressed.

In the following, the formal semantics of the polygon-polyline geo-operators is given.

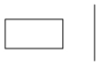
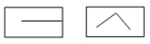
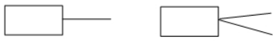
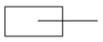
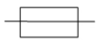
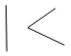
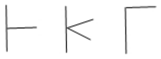
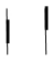
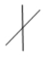
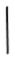
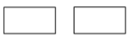
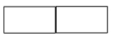
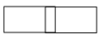
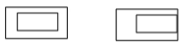
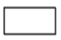
Definition (Polygon-Polyline Geo-Operators): Given a polygon $\mathcal{P}$, and a polyline $\mathcal{L}$, which are two $\mathcal{SGO}$, the binary geo-operations DSJ, INC, TCH, INT, and PTH, are formally defined as follows, where $k, j \in \{i,b,e\}$:

- DSJ (geo-disjunction):
  $\mathcal{P}$ DSJ $\mathcal{L}$ iff $\mathcal{P}_k \cap \mathcal{L}_j = \emptyset\; j, k \neq e$

[1] A polyline is non self-intersecting (self-crossing), without loops, spirals, and bifurcations. A polygon is simple, i.e., it does not intersect itself.

[2] Note that in our approach the operators *cover* and *covered-by*, extensively used in the literature, can be represented by using the Geo-inclusion operator.

TABLE 1. Geo-operators and some related configurations.

polygon-polyline	
DSJ	
INC	
TCH	
INT	
PTH	
polyline-polyline	
DSJ	
TCH	
INC	
CRS	
EQL	
polygon-polygon	
DSJ	
TCH	
OVL	
INC	
EQL	

- INC (geo-inclusion):
  $\mathcal{P}$ INC $\mathcal{L}$ iff $\mathcal{P}_k \cap \mathcal{L}_j = \mathcal{L}_j$, $k = i, j \neq e$
- TCH (geo-touch):
  $\mathcal{P}$ TCH $\mathcal{L}$ iff $\exists x \in \mathcal{L}_j \cap \mathcal{P}_b, j \neq e$, and a neighborhood $I(x)$: $(I(x) \cap \mathcal{P}_e \neq \emptyset$ AND $I(x) \cap \mathcal{P}_i = \emptyset)$ OR $(I(x) \cap \mathcal{P}_e = \emptyset$ AND $I(x) \cap \mathcal{P}_i \neq \emptyset)$.
- INT (geo-intersect):
  $\mathcal{P}$ INT $\mathcal{L}$ iff $\mathcal{P}_k \cap \mathcal{L}_i \neq \emptyset, k = i, e$.
  and $(\mathcal{L}_b \cap \mathcal{P}_e \neq \emptyset$ AND $\mathcal{L}_b \cap \mathcal{P}_i \neq \emptyset)$ OR $(\mathcal{L}_b \cap \mathcal{P}_i \neq \emptyset)$
- PTH (geo-pass-through):
  $\mathcal{P}$ PTH $\mathcal{L}$ iff $\mathcal{P}_k \cap \mathcal{L}_i \neq \emptyset, k = i, e$ AND $\mathcal{L}_b \cap \mathcal{P}_e \neq \emptyset$

Below, the notion of *semi-neighborhood* is recalled, which will be used to formally define the *geo-operators* between polylines [25].

Definition (Semi-Neighborhood): Given a polyline $\mathcal{L} \in \mathcal{SGO}$, let $\mathcal{R}_a^{\mathcal{L}}$ and $\mathcal{R}_b^{\mathcal{L}}$ be the semi-planes of $\mathcal{R}^2$ defined by $\mathcal{L}$ and its extension. Let us consider a point $x \in \mathcal{L}$ and a neighborhood of x, $I(x)$. Then, $I(x)_a^{\mathcal{L}}$ and $I(x)_b^{\mathcal{L}}$ represent the two semi-neighborhoods of $I(x)$ belonging to $\mathcal{R}_a^{\mathcal{L}}$ and $\mathcal{R}_b^{\mathcal{L}}$, respectively, without $\mathcal{L}$.

Finally, the formal semantics of the polyline-polyline and polygon-polygon geo-operators are presented.

Definition (Polyline-Polyline Geo-Operators): Given two polylines $\mathcal{L}$, and $\mathcal{M}$, which are two $\mathcal{SGO}$, the binary geo-operations DSJ, TCH, INC, CRS, and EQL, are formally defined as follows, where $k, j \in \{i,b,e\}$:

- DSJ (geo-disjunction):
  $\mathcal{L}$ DSJ $\mathcal{M}$ iff $\mathcal{L}_k \cap \mathcal{M}_j = \emptyset \; k, j \neq e$
- TCH (geo-touch):
  Assume $S = \mathcal{L}_j \cap \mathcal{M}_k \neq \emptyset$ where $j, k \neq e$. Then:
  $\mathcal{L}$ TCH $\mathcal{M}$ iff $\exists\, x \in S$ and a small enough neighborhood $I(x)$: $I(x)_a^{\mathcal{M}} \cap \mathcal{L}_j = \emptyset$ AND $I(x)_b^{\mathcal{M}} \cap \mathcal{L}_j \neq \emptyset$ AND $\mathcal{L}_e \cap \mathcal{M}_j \neq \emptyset$ AND $\mathcal{L}_j \cap \mathcal{M}_e \neq \emptyset$,
  where $I(x)_a^{\mathcal{M}}, I(x)_b^{\mathcal{M}}$ are semi-neighborhoods of $I(x)$.
- INC (geo-inclusion):
  $\mathcal{L}$ INC $\mathcal{M}$ iff $\mathcal{L}_e \cap \mathcal{M}_j = \emptyset, j \neq e$.
- CRS (geo-cross):
  Assume $S = \mathcal{L}_j \cap \mathcal{M}_k \neq \emptyset$ where $j, k \neq e, b$. Then:
  $\mathcal{L}$ CRS $\mathcal{M}$ iff $\exists\, x \in S$ and a small enough neighborhood $I(x)$: $I(x)_a^{\mathcal{M}} \cap \mathcal{L}_j \neq \emptyset$ AND $I(x)_b^{\mathcal{M}} \cap \mathcal{L}_j \neq \emptyset$,
  where $I(x)_a^{\mathcal{M}}$ and $I(x)_b^{\mathcal{M}}$ are semi-neighborhoods of $I(x)$.
- EQL (geo-equal):
  $\mathcal{L}$ EQL $\mathcal{M}$ iff $\mathcal{L} \cap \mathcal{M} = \mathcal{L} = \mathcal{M}$.

Definition (Polygon-Polygon Geo-Operators): Given two polygons $\mathcal{P}$, and $\mathcal{Q}$, which are two $\mathcal{SGO}$, the binary geo-operations DSJ, TCH, OVL, INC, and EQL, are formally defined as follows, where $k, j \in \{i,b,e\}$:

- DSJ (geo-disjunction):
  $\mathcal{P}$ DSJ $\mathcal{Q}$ iff $\mathcal{P}_k \cap \mathcal{Q}_j = \emptyset \; j, k \neq e$
- TCH (geo-touch):
  $\mathcal{P}$ TCH $\mathcal{Q}$ iff $\mathcal{P}_b \cap \mathcal{Q}_b \neq \emptyset$ AND $\mathcal{P}_i \cap \mathcal{Q}_i = \emptyset$.
- OVL (geo-overlap):
  $\mathcal{P}$ OVL $\mathcal{Q}$ iff $\mathcal{P}_k \cap \mathcal{Q}_j \neq \emptyset, \forall j, k$.
- INC (geo-inclusion):
  $\mathcal{P}$ INC $\mathcal{Q}$ iff $\mathcal{P} \cap \mathcal{Q} = \mathcal{Q}$
- EQL (geo-equal):
  $\mathcal{P}$ EQL $\mathcal{Q}$ iff $\mathcal{P} \cap \mathcal{Q} = \mathcal{P} = \mathcal{Q}$

The above geo-operators are invoked in the GeoPQLJ functions, which are introduced in the Appendix.

A. OCN GRAPH

In this section, below we recall the definition of *Operator Conceptual Neighborhood* (OCN) graph for topological relationships introduced in our previous work [25].

Definition (OCN Graph): The *Operator Conceptual Neighborhood* (OCN) graph is a graph where each node is labeled by one geo-operator corresponding to a possible pictorial configuration, and an arc directly connects two nodes if and only if it is possible to transit from one configuration

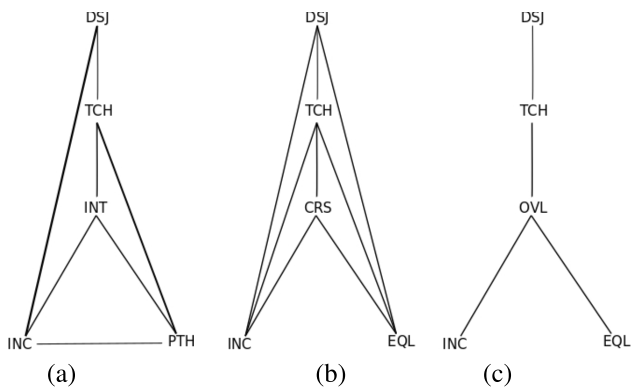


FIGURE 2. OCN graph of polygon-polyline (a), polyline-polyline (b), and polygon-polygon (c) topological relationships.

to another by applying either a *translation* or a *rotation* operations.

By *neighborhood* we mean the continuous translation or rotation of objects within a given topological relationship. For instance, if two polygons touch and one is moved towards the other, they must first overlap before one is included in the other. Essentially, in the graph two nodes are adjacent if and only if the operators they denote can be transformed into each other by continuously modifying the related $\mathcal{SGO}$, applying either a *translation* or a *rotation*.

According to different topological relationships between $\mathcal{SGO}$, in Figure 2 the OCN graphs of polygon-polyline, polyline-polyline, and polygon-polygon topological relationships are shown, respectively.

For instance, let us consider the OCN graph shown Figure 2(c) related to the polygon-polygon topological relationship. The transitions from DSJ to TCH, from TCH to OVL, and from OVL to INC nodes by applying the translations operations are shown in Figure 3. Accordingly, in the transition from DSJ to TCH, we obtain the configuration shown in Figure 3(b), where the polygons have one boundary line in common. Analogously, the configurations shown in Figure 3(c) and Figure 3(d) are the results of the transitions from TCH to OVL, and then from OVL to INC, respectively. Note that the transition from OVL to EQL occurs if and only if the polygons coincide. Furthermore, according to the semantics of the geo-operator INC introduced in Section III, both configurations shown in Figure 3(d) are equivalent.

Analogously, in Figure 2(a) related to the polygon-polyline topological relationship, from the INT node it is possible to transit to the adjacent TCH, INC and PTH nodes. Similarly, in Figure 2(b), related to the polyline-polyline topological relationship, the transitions from CRS to TCH nodes can be obtained by applying a translation operation. Whereas from TCH to INC it is required a rotation and eventually a translation operation. Furthermore, the transition from CRS to EQL occurs by applying a rotation if and only if the lengths of the polylines coincide. Similar considerations hold for the remaining cases.

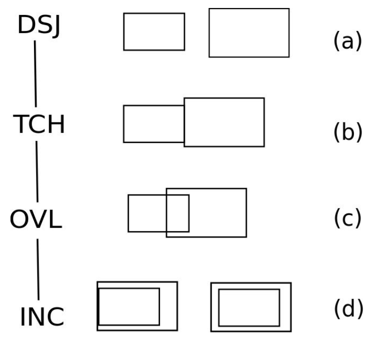


FIGURE 3. Transition from DSJ to other nodes in the polygon-polygon OCN graph.

IV. THE GeoPQLJ DISTRIBUTED SYSTEM

The GeoPQLJ Distributed System is based on the GeoPQLJ functions which conform to the GeoJSON format specification. It is a format for encoding a variety of geographic data structures using JSON [18]. A GeoJSON object may represent a region of space (*Geometry*), a spatial entity (*Feature*), or a list of Features (*FeatureCollection*). It supports the following geometry types: *Point*, *LineString*, *Polygon*, *MultiPoint*, *MultiLineString*, *MultiPolygon*, and *GeometryCollection*.

The geometry types of GeoJSON are defined in the OpenGIS Simple Features implementation Specification for SQL (SFSQL) [2], [6]. They are: 0-dimensional Point, and MultiPoint; 1-dimensional curve LineString, and MultiLineString; 2-dimensional surface Polygon, and MultiPolygon; and the heterogeneous GeometryCollection. GeoJSON representations of instances of these geometry types are analogous to the Well-Known Text (WKT) and Well-Known Binary (WKB) representations, originally defined by OGC [2]. In particular, the former is a text markup language for representing geometry objects according to a vector format and reference systems of spatial objects. The latter is used to transfer and store the same information in specific geographic databases.

The GeoPQLJ functions have been inspired by taking into account the syntax of *Turf.js* [5]. In the following, the spatial types *Point*, *LineString*, and *Polygon*, which allow us to define the spatial operators' functions, are given. These operators are: *disjoint*, *inclusion*, *touch*, *intersect*, *passthrough*, *cross*, *overlap*, and *equal*, which are admissible for representing the topological relationships between $\mathcal{SGO}$. Table 2 summarizes the above mentioned functions with the corresponding definitions and invoked GeoPQL operators.

In Figure 4 the diagram of the GeoPQLJ Distributed Systems is shown. As shown in this figure, within the local GeoPQL system a *Feature* can be associated with a set of *attributes*. Note that, in this paper we assume that in a query a feature is associated with at most one attribute. Suppose the user expresses a pictorial query q. It is transformed into a query by identifying the corresponding operator *GeoPQL_op*, as follows:

$$Feature_1\ GeoPQL\_op\ Feature_2$$

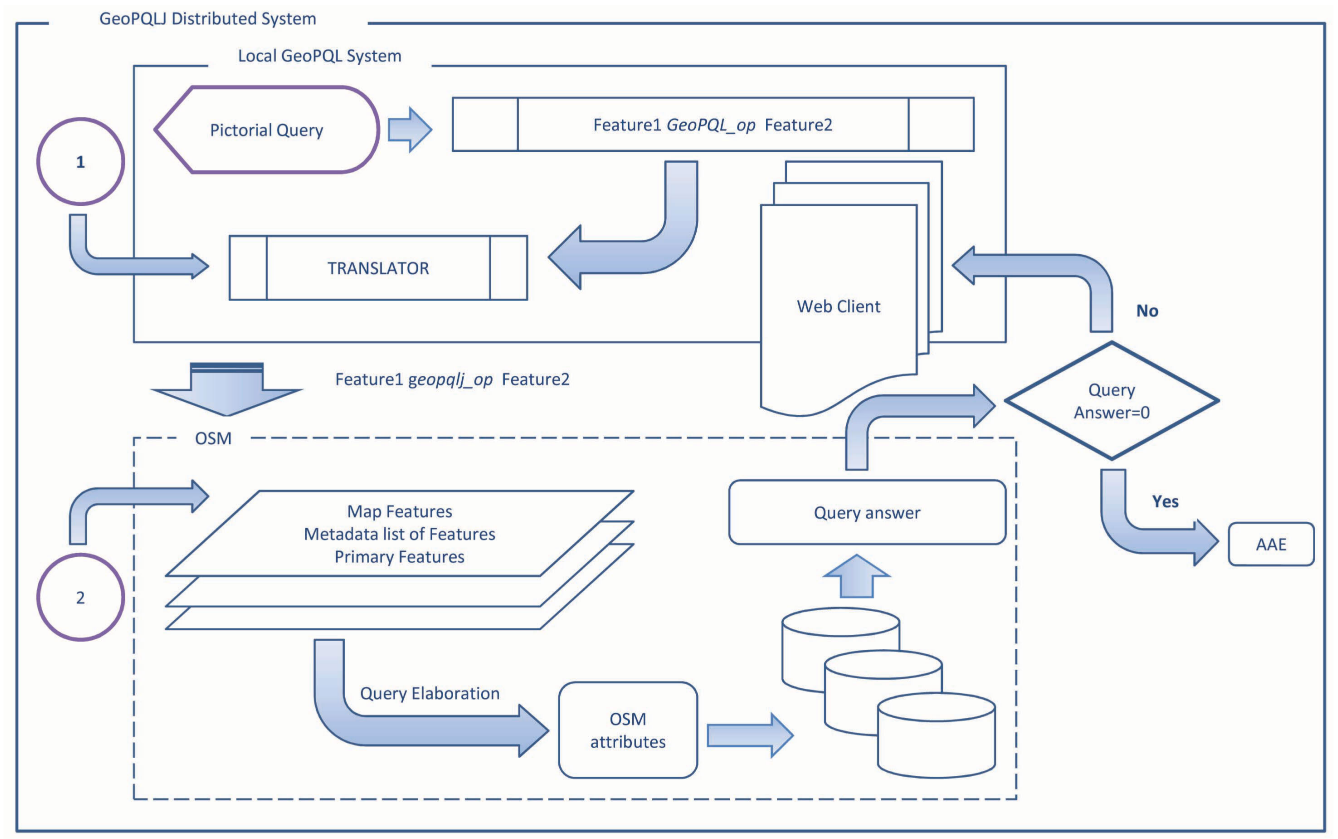


FIGURE 4. GeoPQLJ Distributed System diagram.

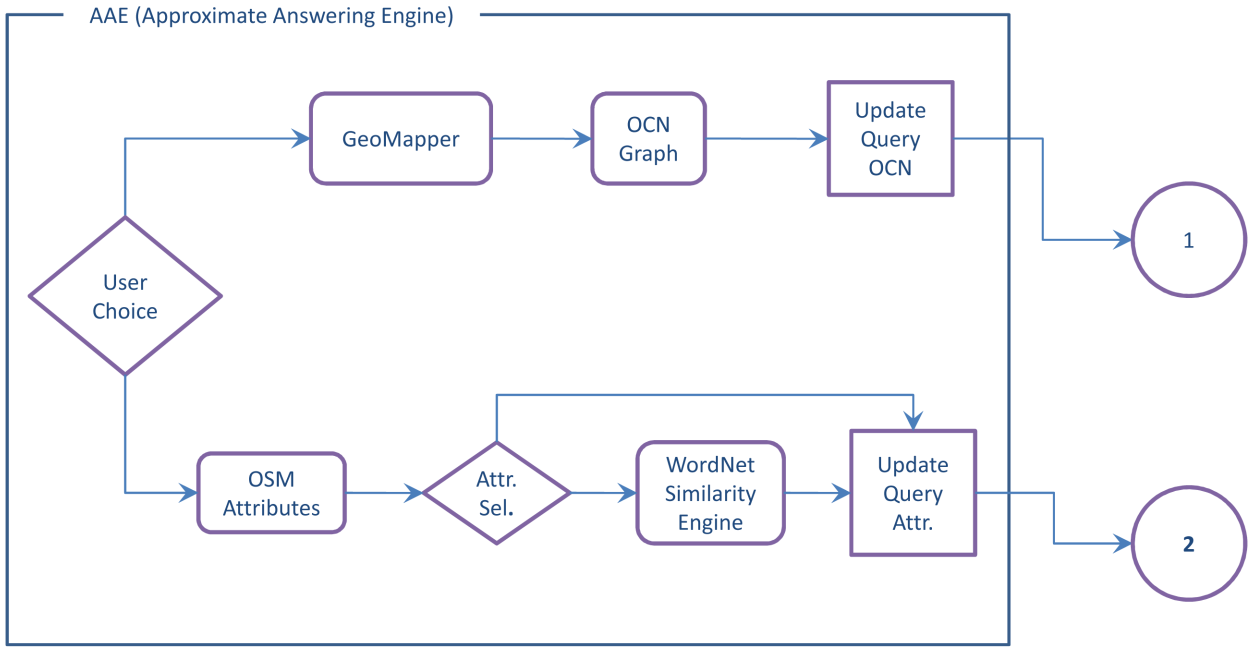


FIGURE 5. Approximate Answering Engine (AAE) diagram.

where $Feature_i, i = 1, 2$, are associated with *geometric_types*, and $Feature_1$ is the target of the query. Successively, the query is translated into the format defined according to the GeoPQLJ functions described above, and the following

TABLE 2. GeoPQLJ functions.

$Name$	$Def.$	$Op.$
$geopqlj\_dsj$	Ret. true when $\mathcal{SGO}_1$ is disjoint from $\mathcal{SGO}_2$	DSJ
$geopqlj\_inc$	Ret. true when $\mathcal{SGO}_1$ is in $\mathcal{SGO}_2$	INC
$geopqlj\_tch$	Ret. true when $\mathcal{SGO}_1$ touches $\mathcal{SGO}_2$	TCH
$geopqlj\_int$	Ret. true when $\mathcal{SGO}_1$ intersects $\mathcal{SGO}_2$	INT
$geopqlj\_pth$	Ret. true when $\mathcal{SGO}_1$ passes through $\mathcal{SGO}_2$	PTH
$geopqlj\_crs$	Ret. true when $\mathcal{SGO}_1$ crosses $\mathcal{SGO}_2$	CRS
$geopqlj\_ovl$	Ret. true when $\mathcal{SGO}_1$ overlaps $\mathcal{SGO}_2$	OVL
$geopqlj\_eql$	Ret. true when $\mathcal{SGO}_1$ is equal to $\mathcal{SGO}_2$	EQL

GeoPQLJ query is generated:

$$Feature_1 \ geopqlj\_op \ Feature_2$$

For instance, suppose $Feature_1$ and $Feature_2$ are associated respectively with the *geometric_types* polygon and polyline, the above *geopqlj_op* is elaborated as follows:

$geopqlj\_op($
$\quad geom1\_from\_wkt('Polygon([[x_1, y_1], [x_2, y_2], \ldots,$
$\qquad [x_n, y_n], [x_1, y_1]])'),$
$\quad geom2\_from\_wkt('LineString([[x_1, y_1], [x_2, y_2], \ldots,$
$\qquad [x_n, y_n]])')$
$\quad) \rightarrow true$

where *geom1* and *geom2* are the *geometric_types* associated with $Feature_1$ and $Feature_2$, respectively. The *wkt* method converts the geometries to WKT geometry formats [5]. In the *wkt* method, the WKTReader extracts the geometry objects from either *Readers* (i.e., the abstract class for reading character streams) or *Strings*. It is a parser that allows the reading of the geometry objects from text blocks embedded in other data formats (e.g., XML). The syntax of the *geopqlj_op* are given in the Appendix.

Then, the above query is sent to OSM which connects to the data sources in order to select the required URLs (see the dashed box shown in Figure 4 and, eventually, to associate attributes with features:

$$Feature_1(attr_1) \ geopqlj\_op \ Feature_2(attr_2)$$

In other words, OSM uses the GeoJSON data format in order to access the territorial data directly from distributed data sources, as well as to analyze and process them. OSM uses the well-known *MapFeatures*, *Metadata list of Features*, and *Primary Features*. They capture the general information about the metadata that contain the list of features corresponding to the query. This list is associated with the Geodetic Parameter Registry (EPSG), that allows the accurate overlapping of the required features.

As already mentioned in the Introduction, OSM is a digital archive of geospatial data represented on the surface, as for instance roads, buildings, parks, etc., that are objects referred to as features. The OSM features allow the access to the content of data attributes, and the related spatial coordinates. They represent the database structure, and each of them is associated with a set of *attributes* denoting geographical characteristics. Finally, the query answer is sent to the Web Client, and if it is non-empty, it is shown as a map. Otherwise, in the case the answer is empty, the AAE is activated, which is shown in Figure 5, and described in the next subsection.

A. APPROXIMATE ANSWERING ENGINE

In this section, the AAE proposed in this paper is illustrated. In the presence of empty answers, it provides the user with one or more results that better approximate the given queries. Analogously to GeoPQL, in OSM a query is defined by two features ($Feature_i$, $i = 1, 2$), each having the same name of the class it refers to. The first feature is the *target* of the query, and defines the elements expected in the answer. For this reason, we say that in a query *symmetry* does not hold because by exchanging the features, and therefore the target of the query, in general we do not have the same result. Furthermore, also in OSM, a feature is associated with an eventually empty set of *attributes*, which in this paper is restricted to a singleton. The set of attributes belonging to a given feature in OSM is organized according to a ISA hierarchy, which is often a *forest*. For instance, consider the previous query q:

$$Feature_1(attr_1) \ geopqlj\_op \ Feature_2(attr_2)$$

and suppose that the answer to this query is empty (see on right side of Figure 4). In order to give an approximate answer, the AAE system provides the user with two different solutions. They are based on the OCN graph and the OSM attribute hierarchy, respectively, as descried below.

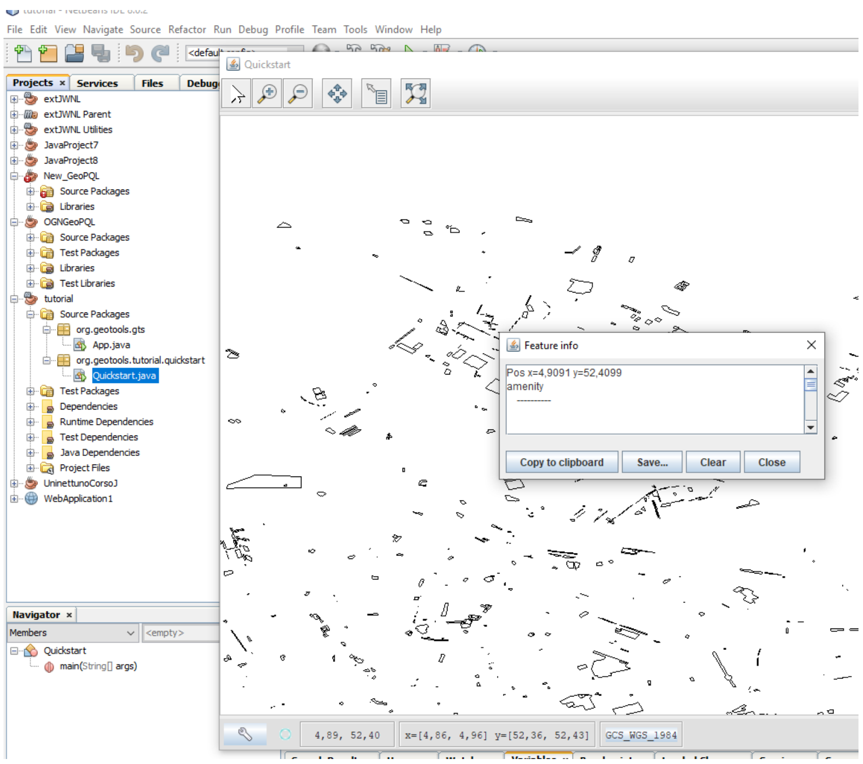
- **OCN graph**. According to this approach, the OCN graph corresponding to the topological relationships involved in the query is considered. To this end, the geo-operator of the above query is mapped to the original GeoPQL operator of q, by means of the *GeoMapper*, as follows:

  $$Feature_1(attr_1) \ GeoPQL\_op \ Feature_2(attr_2)$$

  Then, the geo-operators labeling the adjacent nodes of the *GeoPQL_op* in the OCN graph are considered. Therefore, the user decides which is the operator that best approximates the original query, on the basis of his/her interests. Suppose the user chooses, among the nodes adjacent to *GeoPQL_op*, the node labeled with $GeoPQL\_op_k$. Then, the query is updated as follows:

  $$Feature_1(attr_1) \ GeoPQL\_op_k \ Feature_2(attr_2)$$

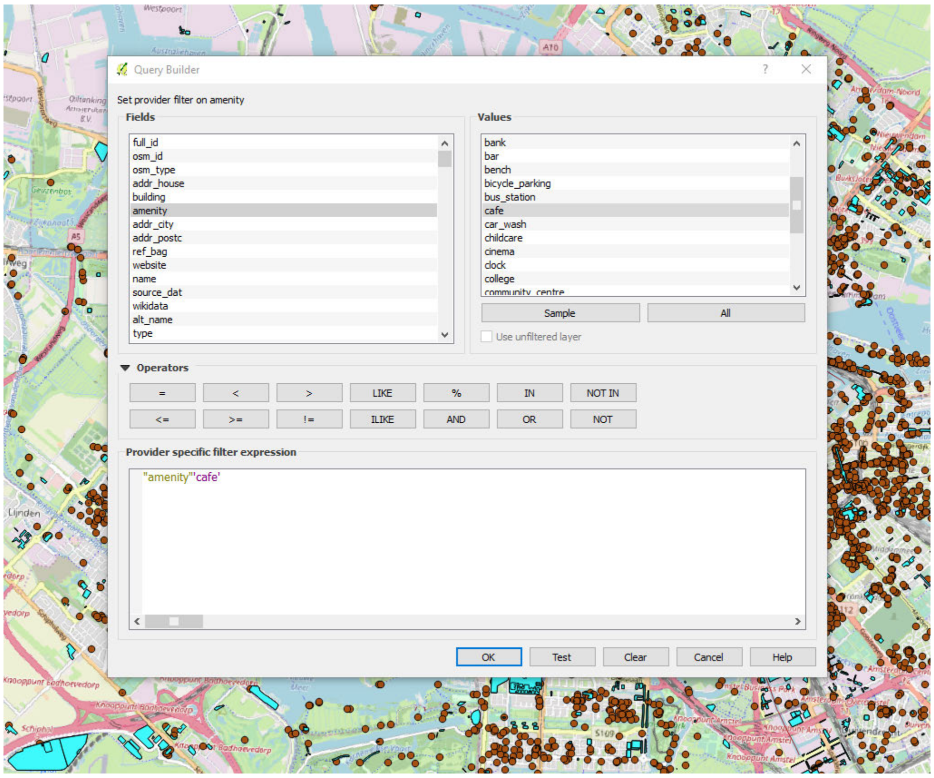
  which is again submitted to the local GeoPQL system, as shown in Figure 4 (see Step ①), and elaborated according to the steps illustrated above in the GeoPQLJ Distributed System. The result is then proposed to the user as a approximate answer to the query q.
- **OSM attribute ISA hierarchy**. In order to have an approximate answer to the query q, in place of the OCN graph, it is possible to access the sets of OSM attributes associated with the features involved in the query. Attributes in OSM are organized according to ISA hierarchies. The goal of the approach is to replace



```
Pos x=4,9091 y=52,4099
amenity
  ---------
```

```
4,89, 52,40    x=[4,86, 4,96] y=[52,36, 52,43]
```

FIGURE 6. Selection of *amenity* feature in OSM.

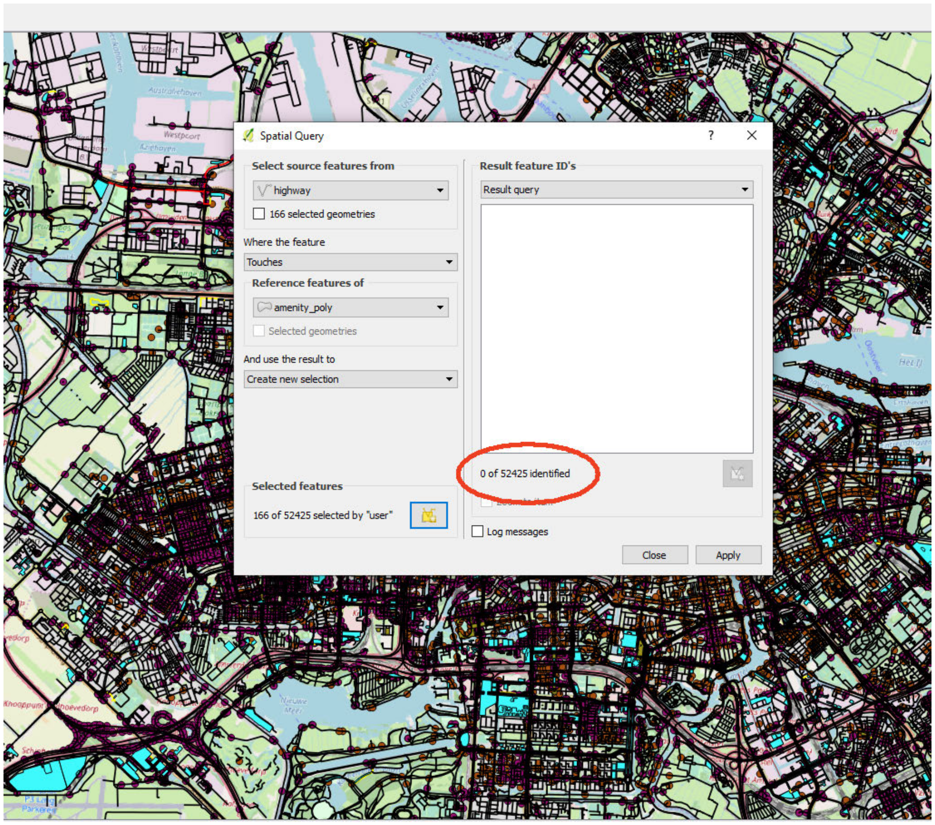


Set provider filter on amenity

Provider specific filter expression

```
"amenity"'cafe'
```

FIGURE 7. Filtering *amenity* feature by selecting *cafe* attribute in OSM.



Spatial Query

Select source features from: highway

166 selected geometries

Where the feature: Touches

Reference features of: amenity_poly

Selected geometries

And use the result to: Create new selection

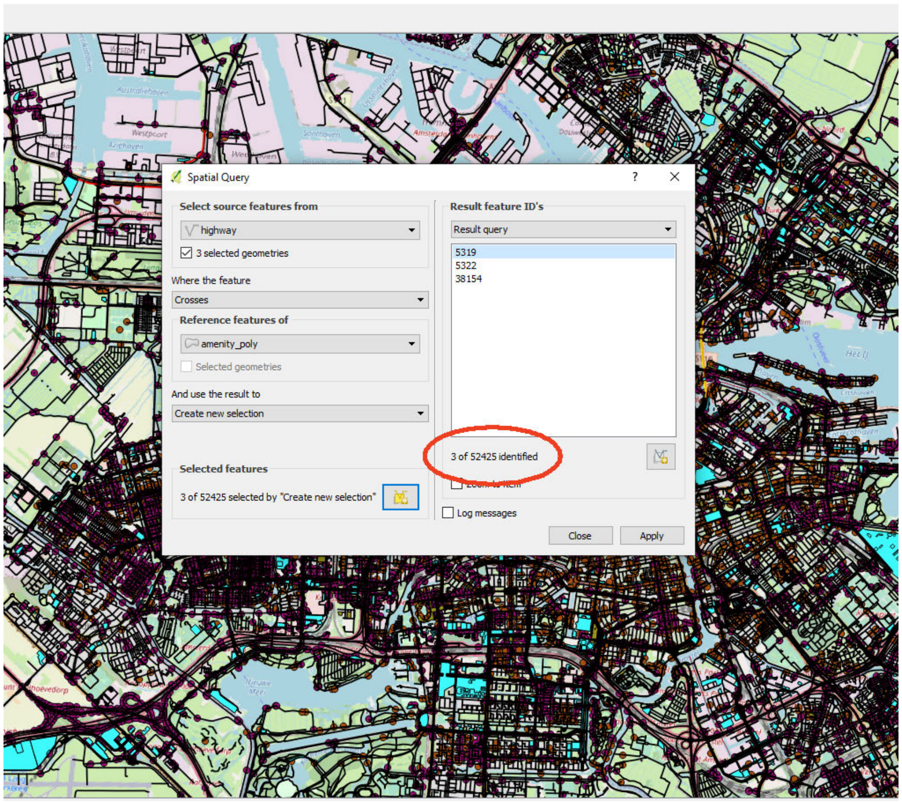
Selected features: 166 of 52425 selected by "user"

Result feature ID's: Result query

0 of 52425 identified

Log messages

FIGURE 8. Empty answer to *highway geopqlj_tch amenity(cafe)* query.



Spatial Query

Select source features from: highway

☑ 3 selected geometries

Where the feature: Crosses

Reference features of: amenity_poly

Selected geometries

And use the result to: Create new selection

Selected features: 3 of 52425 selected by "Create new selection"

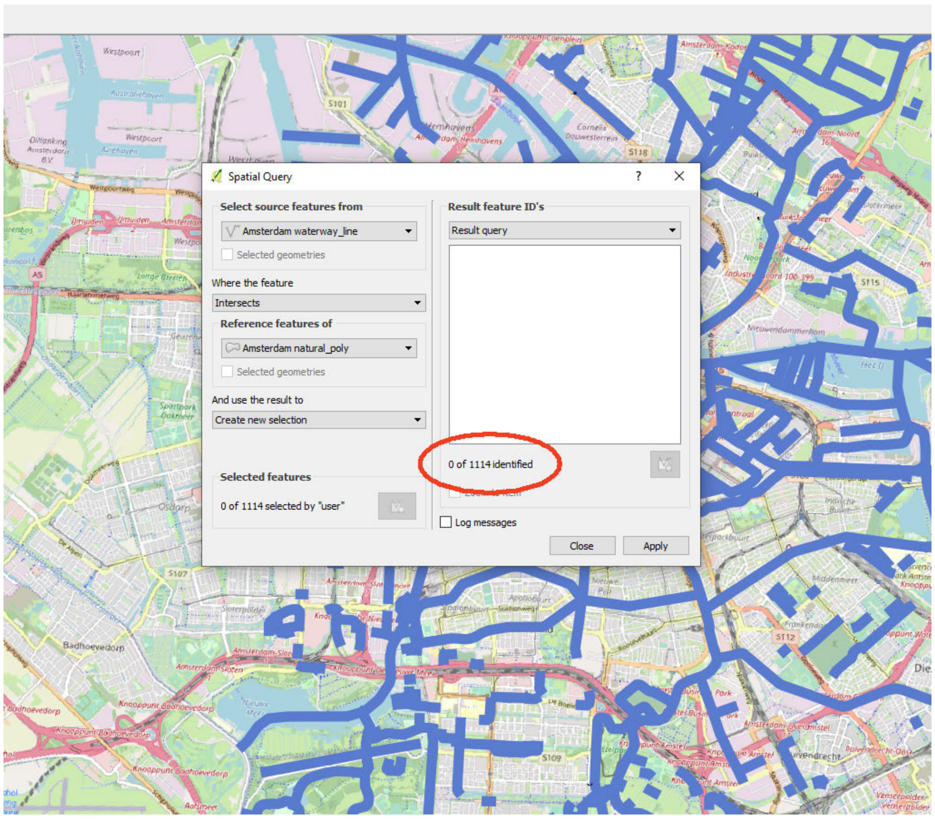
Result feature ID's: Result query

```
5319
5322
38154
```

3 of 52425 identified

Log messages

FIGURE 9. Answer to *highway PTH amenity(cafe)* query.



Spatial Query

Select source features from

Amsterdam waterway_line

Selected geometries

Where the feature

Intersects

Reference features of

Amsterdam natural_poly

Selected geometries

And use the result to

Create new selection

Selected features

0 of 1114 selected by "user"

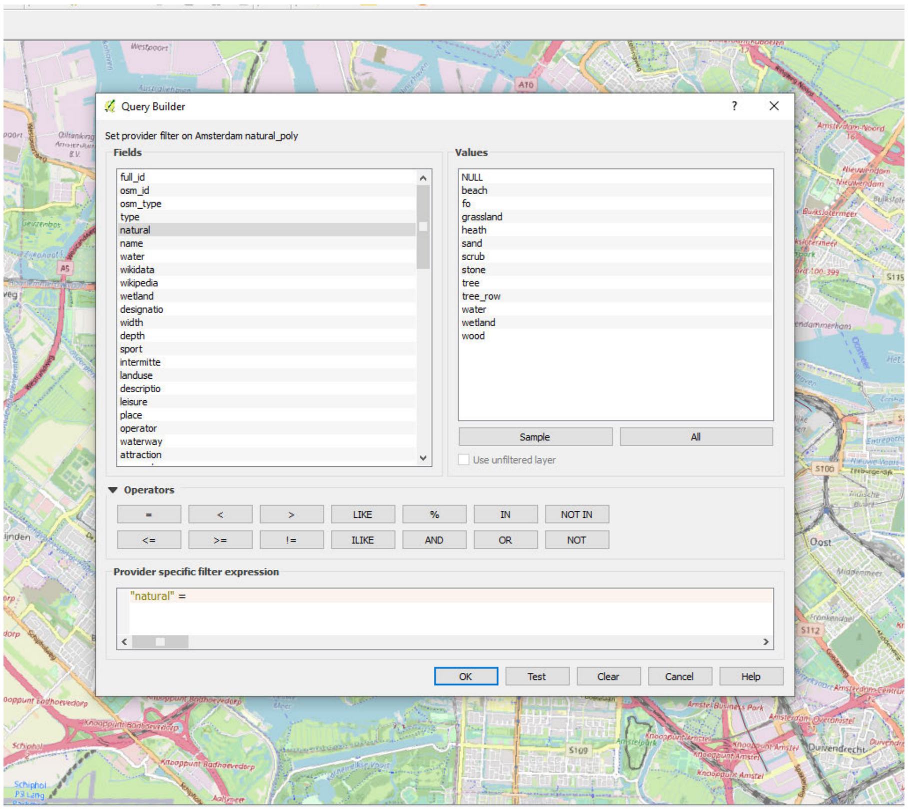
Result feature ID's

Result query

0 of 1114 identified

Log messages

FIGURE 10. Empty answer to *waterway INT natural(grassland)* query.



Query Builder

Set provider filter on Amsterdam natural_poly

Fields

full_id
osm_id
osm_type
type
natural
name
water
wikidata
wikipedia
wetland
designatio
width
depth
sport
intermitte
landuse
descriptio
leisure
place
operator
waterway
attraction

Values

NULL
beach
fo
grassland
heath
sand
scrub
stone
tree
tree_row
water
wetland
wood

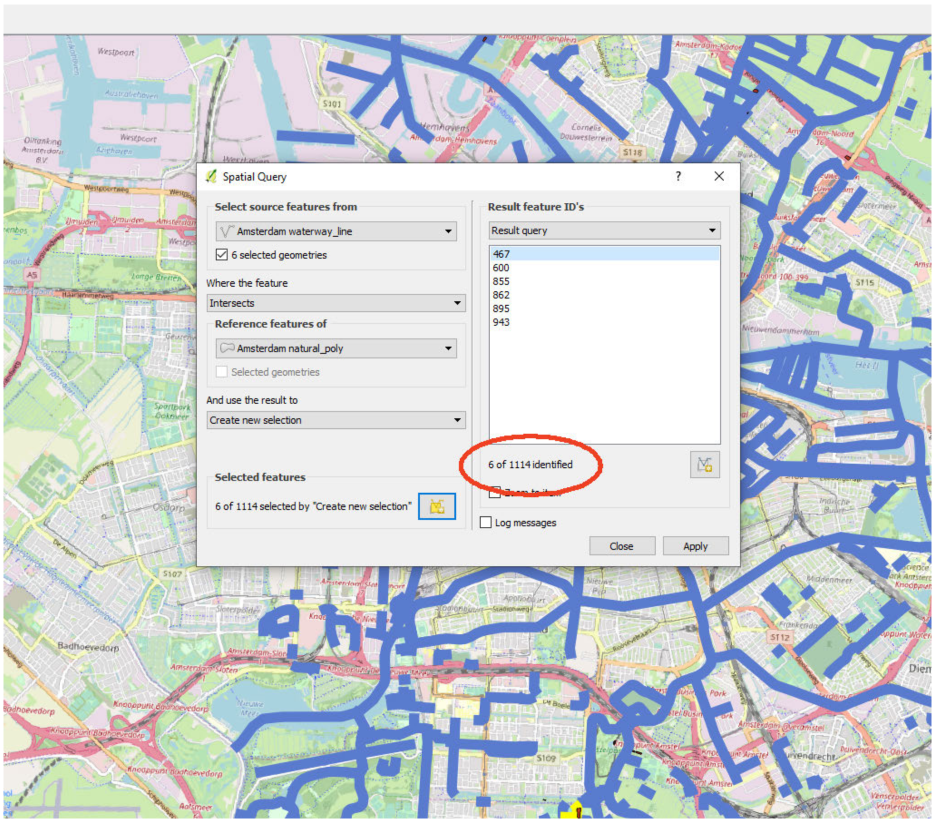
Use unfiltered layer

Operators

Provider specific filter expression

"natural" =

FIGURE 11. List of attributes of *natural* feature in OSM.



Spatial Query

Select source features from: Amsterdam waterway_line
☑ 6 selected geometries
Where the feature: Intersects
Reference features of: Amsterdam natural_poly
☐ Selected geometries
And use the result to: Create new selection

Result feature ID's: Result query

467
600
855
862
895
943

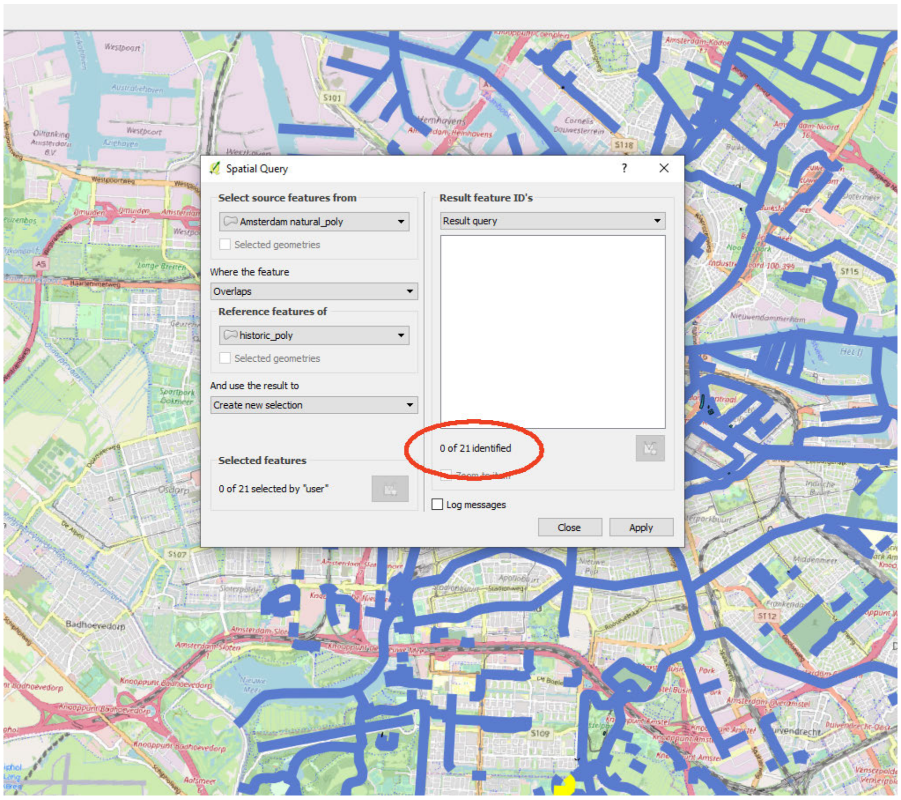
6 of 1114 identified

Selected features
6 of 1114 selected by "Create new selection"

☐ Log messages

Close Apply

FIGURE 12. Answer to *waterway geopqlj_int natural(wood)* query.



Spatial Query

Select source features from: Amsterdam natural_poly
☐ Selected geometries
Where the feature: Overlaps
Reference features of: historic_poly
☐ Selected geometries
And use the result to: Create new selection

Result feature ID's: Result query

0 of 21 identified

Selected features
0 of 21 selected by "user"

☐ Log messages

Close Apply

FIGURE 13. Empty answer to *natural(grassland) OVL historic* query.

one or both the attributes of the query q with the ones preferred by the user, if there are any, or similar attributes in the OSM ISA hierarchy. In particular, in our approach attribute similarity is performed according to the semantic similarity approach defined by Lin [36], also referred to as *information content* approach. It is based on the association of probabilities with the attributes (concepts) of the ISA hierarchy. The *probability* of an attribute a is defined as:

$$p(a) = \frac{f(a)}{M} \tag{1}$$

where *f(a)* is the *frequency* of the attribute a estimated using noun frequencies from large text corpora, as for instance the *Brown Corpus of American English*, and M is the total number of observed instances of nouns in the corpus. According to the standard argumentation of information theory, the information content of a is defined as $-\log p(a)$. We assume that the ISA hierarchy is a tree, therefore the least upper bound (*lub*) of any pair of attributes is always defined and provides the maximum information content shared by the pair of attributes in the hierarchy. Formally, given two attributes a_i and a_j of the ISA hierarchy, their similarity, referred to as $sim(a_i,a_j)$, is defined as the maximum information content shared by the attributes divided by the sum of their information contents:

$$sim(a_i, a_j) = \frac{2 \log p(lub(a_i, a_j))}{\log p(a_i) + \log p(a_j)} \tag{2}$$

In order to evaluate Lin's similarity, in our proposal we relay on the WordNet similarity engine [7]. This engine allows the evaluation of the similarity between nouns, verbs, adjective, etc. by following different similarity metrics defined in the literature. In our proposal, the information content approach of Lin is adopted because it has been extensively experimented and shows a higher correlation with human judgment with respect to most of the similarity methods defined in the literature [36].

Consider one of the attributes of the query q, for instance $attr_1$, and suppose the user, once accessed the set of OSM attributes related to the $Feature_1$, wants to replace $attr_1$ with the attribute $attr_k$ that is either the most similar to $attr_1$ according to Lin, or it is one among his/her favorite ones, independently of the similarity values. Then, the query:

$$Feature_1(attr_k)\ geopqlj\_op\ Feature_2(attr_2)$$

is submitted to the local GeoPQLJ Distributed System (see Step ②), and elaborated. The result to this query is then proposed to the user as an approximate answer to the query q.

V. THE EXPERIMENT

In this section some experiments are shown by considering the geodata related to the city of Amsterdam, stored in the OSM data repositories. Note that OSM data includes all the elements of maps such as nodes, ways, and relations, which are gathered in the *Planet.osm* file of 1192.4 GB size when uncompressed [4]. In order to evaluate the proposed approach, an area of Amsterdam delimited by a given bounding box was selected, and the related XML information has been extracted from OSM. In this area, the number of nodes is 15249954, and the size of data is 557 MB.

TABLE 3. Similarity values of *grassland* with some attributes of *natural*.

Attributes of the natural feature	*Lin's similarity scores*
tree	0.15
sand	0.08
beach	0.13
water	0.12
wood	0.35
wetland	0.12

Let us start by considering the query presented in the Introduction, i.e., suppose the user is interested in the number of highways touching some amenities where it is possible to have a coffee break during a trip in Amsterdam. This query, that is pictorially defined in the Local GeoPQL System as shown in Figure 1 in the Introduction, is formulated in GeoPQL as:

$$highway\ TCH\ amenity$$

where the *geometric_types* associated with the features *highway* and *amenity* are polyline and polygon respectively and, as anticipated in the Introduction, *highway* is the target of the query. It is translated into the following query by associating with *TCH* the *geopqlj_tch* operator as follows:

$$highway\ geopqlj\_tch\ amenity$$

which is elaborated as:

$geopqlj\_tch($
$\quad geom1\_from\_wkt('LineString([[x_1, y_1], [x_2, y_2], \ldots,$
$\qquad [x_n, y_n]])')$
$\quad geom2\_from\_wkt('Polygon([[x_1, y_1], [x_2, y_2], \ldots,$
$\qquad [x_n, y_n], [x_1, y_1]])'),$
$\quad) \rightarrow true$

In Figure 6, the selection of the feature *amenity* is illustrated. In addition, since the user is interested in an amenity where it is possible to have a coffee, he/she can also access the list of attributes of amenity, as shown in Figure 7, in order to select the preferred one which, in this case, is *cafe*, i.e.:

$$highway\ geopqlj\_tch\ amenity(cafe)$$

This query is submitted to OSM which, in the case of the selected bounding box for Amsterdam, provides an empty answer as shown in Figure 8.

In particular the above query involves 52425 highways, 2013 amenities, and 17 coffee shops. Note that, among them, 166 of 52425 is the number of highways touching the amenities (see bottom left side in Figure 8). However, with the

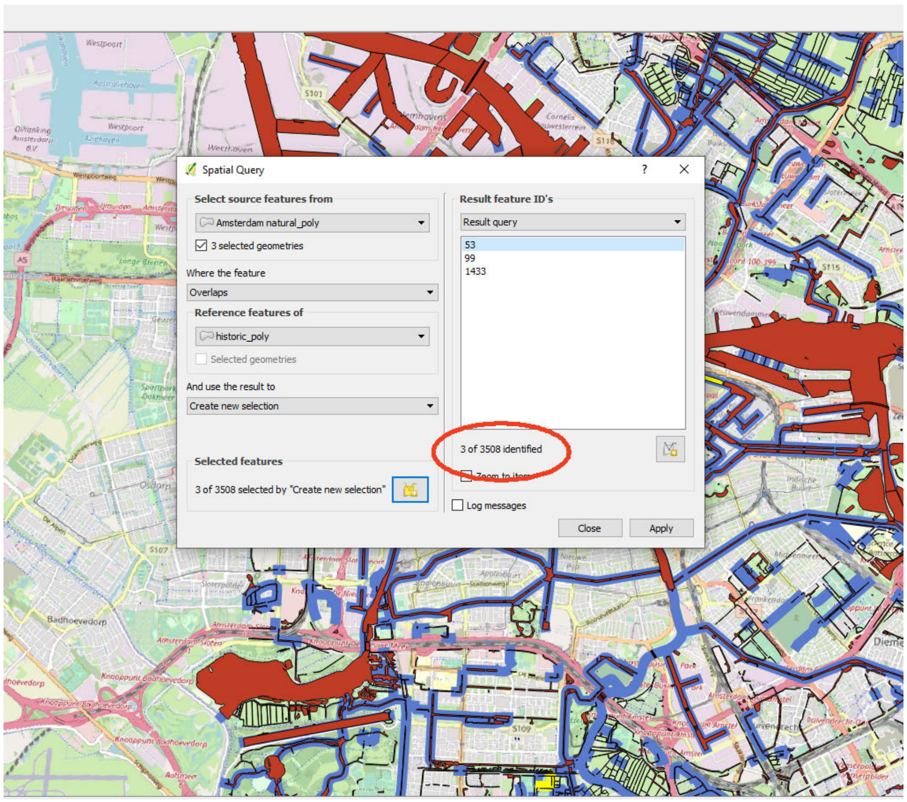


FIGURE 14. Answer to *natural(water) geopqlj_ovl historic* query.

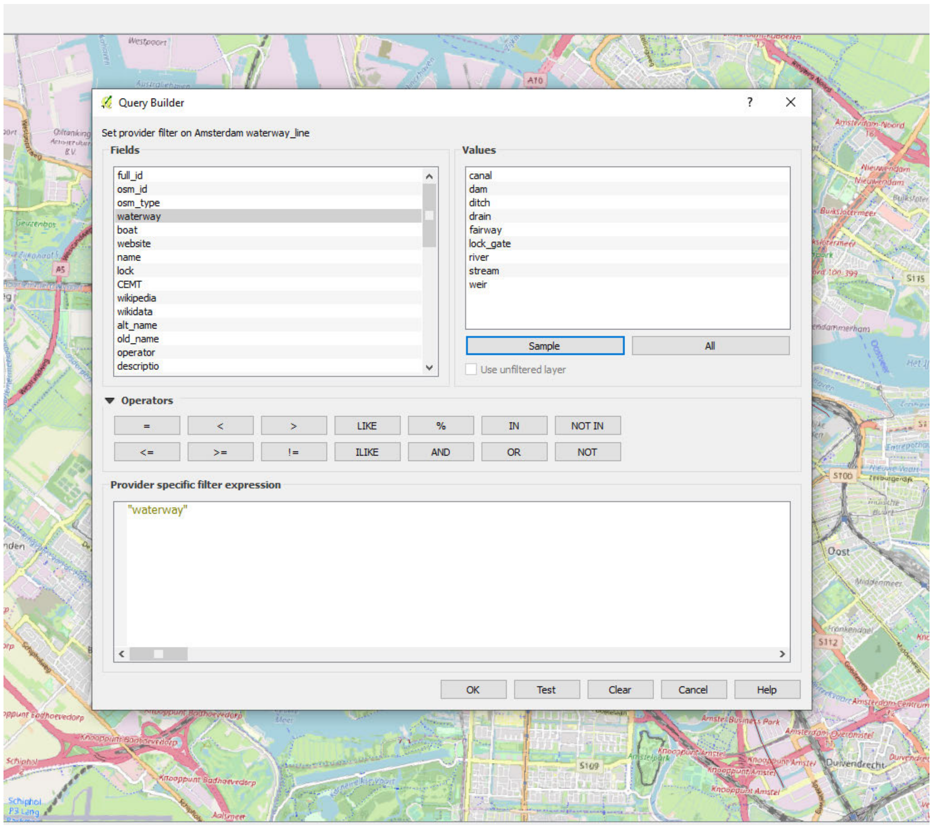
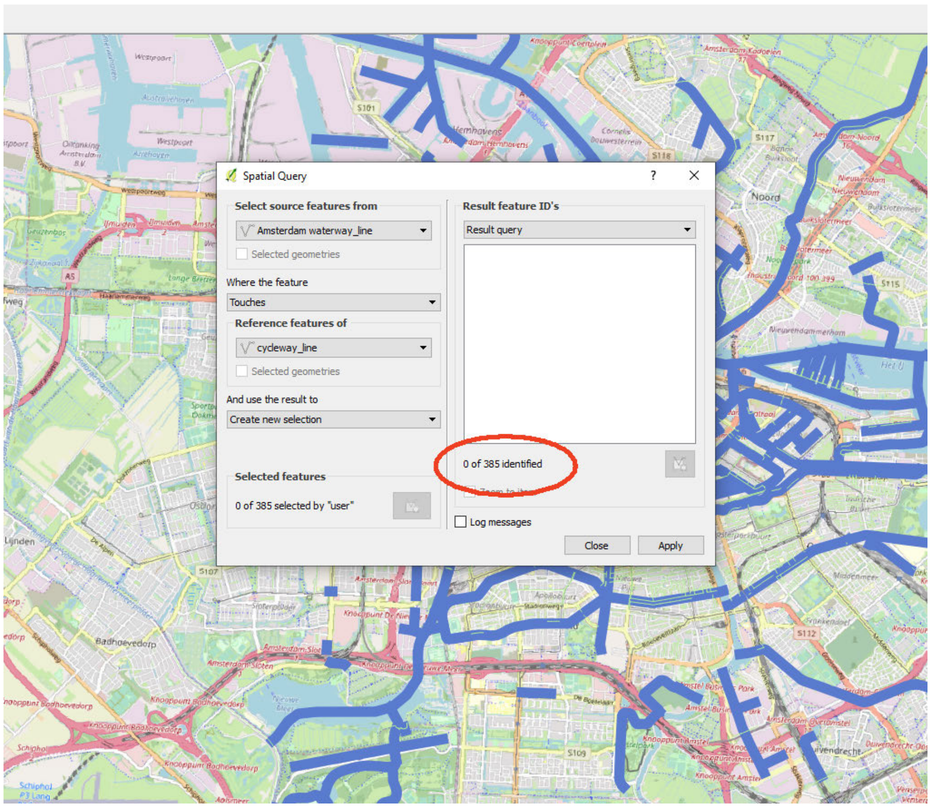
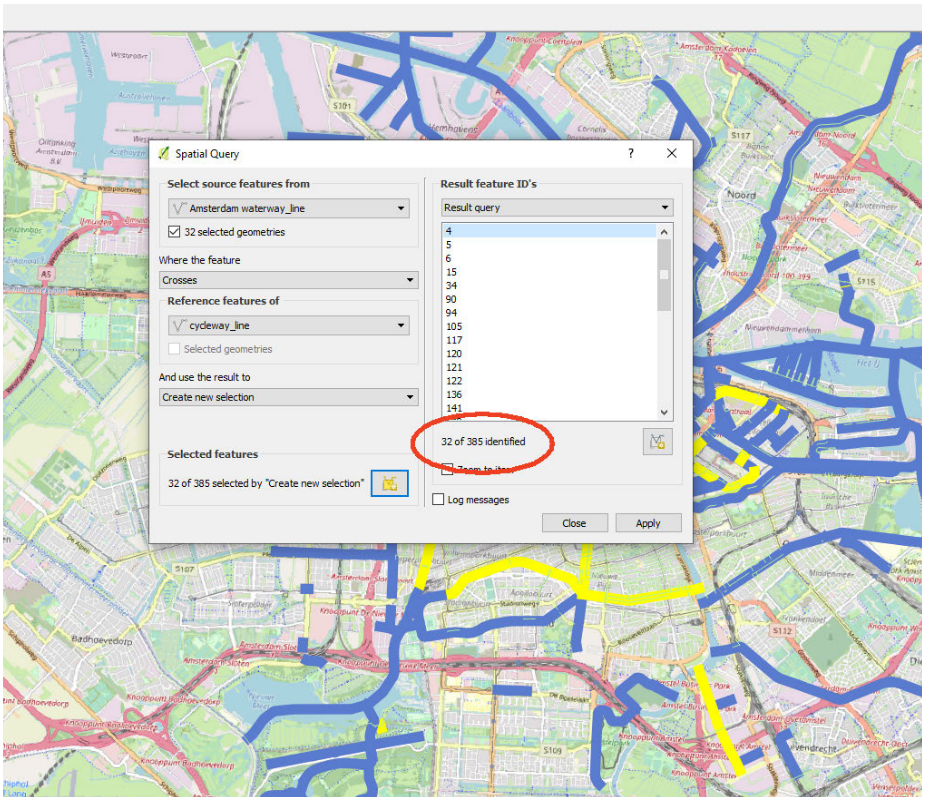


FIGURE 15. List of attributes of *waterway* in OSM.



Select source features from
Amsterdam waterway_line
Selected geometries
Where the feature
Touches
Reference features of
cycleway_line
Selected geometries
And use the result to
Create new selection
Selected features
0 of 385 selected by "user"
Result feature ID's
Result query
0 of 385 identified
Log messages

FIGURE 16. Empty answer to *waterway(canal) TCH cycleway* query.



Select source features from
Amsterdam waterway_line
☑ 32 selected geometries
Where the feature
Crosses
Reference features of
cycleway_line
Selected geometries
And use the result to
Create new selection
Selected features
32 of 385 selected by "Create new selection"
Result feature ID's
Result query
4
5
6
15
34
90
94
105
117
120
121
122
136
141
32 of 385 identified
Log messages

FIGURE 17. Answer to *waterway(canal) CRS cycleway* query.

TABLE 4. Benchmarks about query response time in seconds (*s*).

Feaure$_1$ *Geo-op. Feaure*$_2$	*#Feature*$_1$	*#Feature*$_2$	*Answer*	*Time(s)*	*Figure*
highway TCH amenity(cafe)	52425	2013	0	<1	8
highway PTH amenity(cafe)	52425	2013	3	22	9
waterway INT natural(grassland)	1114	4279	0	<1	10
waterway INT natural(wood)	1114	4279	6	7	12
natural(grassland) OVL historic	4279	78	0	<1	13
natural(water) OVL historic	3508	78	3	<1	14
waterway(canal) TCH cycleway	1114	1382	0	<1	16
waterway(canal) CRS cycleway	1114	1382	32	<1	17

addition of the attribute *cafe*, the answer to this query is empty, i.e., there are no highways touching an amenity with a coffee shop in Amsterdam, as shown in Figure 8.

Therefore the user can look for possible approximated answers to this query by following our proposed approach. For instance, suppose he/she wants to follow the first direction, i.e., replacing the geo-operator *TCH*. Therefore, as shown in Figure 5, according to the GeoMapper, the *geopqlj_tch* geo-operator is converted to the corresponding GeoPQL *TCH*, and the OCN graph of the polygon-polyline topological relationships is accessed. Suppose the user selects the *PTH* geo-operator because, among the adjacent nodes to *TCH*, he/she is interested in the highways which pass trough a coffee. Then, the query is updated as follows:

$$highway\ PTH\ amenity(cafe)$$

and submitted to the system (see Step ①). The obtained answer is 3, i.e., there are 3 among the 52425 highways which pass through an amenity with a coffee shop in Amsterdam, as shown in Figure 9.

In order to illustrate the second direction, consider the following query:

$$waterway\ INT\ natural(grassland)$$

where *waterway* is the target of the query and is of type polyline, whereas *natural* is of type polygon, taking into account that in Amsterdam there are 1114 waterways, 4279 naturals, and 21 grasslands. Also in this case, the answer to this query is empty, i.e., there are no waterways intersecting a natural which is a grassland in Amsterdam, as shown in Figure 10.

Suppose the user wants to modify the attribute *grassland* of *natural*. Then, he/she accesses the list of OSM attributes of the *natural* feature which are organized according to the ISA hierarchy shown in Figure 11, that are *tree*, *sand*, *beach*, *water*, etc..

Furthermore, assume the user wants to select the most similar attribute to *grassland*. Therefore, each of the attributes shown in Figure 11 is compared with *grassland*, whose sense according to WordNet is: "land where grass or grasslike vegetation grows and is the dominant form of plant life". The similarity values between *grassland* and each attribute associated with *natural* are computed according to the semantic approach of Lin. A subset of attributes of *natural* representing the most similar ones with *grassland* are shown in the Table 3.

In this table, the most similar attribute to *grassland* is *wood* (their similarity is 0.35), whose sense is "the trees and other plants in a large densely wooded area". Therefore, by considering that there are 1114 waterways, 52425 naturals (as mentioned above) and 33 woods, the above query is modified as follows:

$$waterway\ geopqlj\_intnatural(wood)$$

and submitted to the system (see Step ②), where *geopqlj_int* corresponds to the original geo-operator *INT* of the query. As shown in Figure 12 the answer to this query is equal to 6, i.e., there are 6 waterways among 1114 identified, which intersect a wood natural area (see bottom left side of Figure 12).

Consider now a query involving the polygon-polygon topological relationship and suppose the user is interested in all the grasslands overlapping the historical buildings in Amsterdam. This query can be formulated as follows:

$$natural(grassland)\ OVLhistoric$$

where both the *geometric_types* of *natural* and *historic* are polygons, considering that there are 4279 naturals of type polygon as shown before, 21 grasslands, and 78 historic buildings of type polygon. The answer to this query is empty, see Figure 13, i.e., there are no grasslands, among the 21 in Amsterdam, overlapping one of its 78 historical buildings.

Suppose now the user prefers to replace the attribute *grassland* rather than modifying the *OVL* operator by accessing the OCN graph. In particular, in place of considering the most similar attribute of *grassland* in the ISA hierarchy (which, as shown above, is *wood*), the user is interested in the specific attribute *water*. Therefore, taking into account that in Amsterdam there are 3508 waters, the query becomes:

$$natural(water)\ geopqlj\_ovl\ historic$$

where *geopqlj_ovl* corresponds to the *OVL* geo-operator.

As shown in the bottom left side of Figure 14, the answer to this query is 3, i.e., there are 3 waters among the 3508 naturals, which overlap historic buildings in Amsterdam.

Finally, in order to provide an example about a query involving the polyline-polyline topological relationship, suppose the user is interested in obtaining the canals, which are waterways, touching a cycleway (the attributes associated with *waterway* are shown in Figure 15). Considering that there are 1114 waterways, 385 canals, and 1382 cycleways, the query is the following:

$$waterway(canal)\ TCH\ cycleway$$

where the *geometric_types* of both the features *waterway* and *cycleway* are polylines. Also in this case the answer to this query is empty, as shown in Figure 16. Suppose the user is interested in accessing the OCN graph and replacing the TCH operator with the CRS operator, therefore asks the canals which cross, rather than touch, the cycleways in Amsterdam. The answer to this query is 32, i.e., there are 32 among the 385 canals which cross a cycleway in Amsterdam, as shown in Figure 17.

In Table 4, the benchmarks related to the response time in seconds (s) of the eight queries described above are presented. Note that, we ran the queries on a workstation 2CPU Intel Xeon, 2.666 GHZ, 16 slots, DDR4 RAM 128 GB, GPU Memory16 GB GDDR5X. As we observe, the execution time of the query shown in Figure 9 is significantly higher ($22s$) than the others because the computation of the PTH geo-operator requires more complex operations, according to the formal semantics defined in Section III. In fact, besides the verification of the conditions related to the intersection among the internal and external points of the involved features, also the condition about both the boundary points of the polyline, that must be external to the polygon, has to be checked.

VI. CONCLUSION

In this paper we proposed the *Approximate Answering Engine* within the GeoPQLJ Distributed System, which provides answers to empty queries according to either the *Operator Conceptual Neighborhood* graph or the *OpenStreetMap* (OSM) attribute hierarchy, giving maximum flexibility to the user choices. In the GeoPQLJ Distributed System, queries are formulated by the GeoPQL pictorial query language in the Local GeoPQL System, whose semantics has been revised for the polygon-polyline, polyline-polyline, and polygon-polygon topological relationships, and the corresponding GeoPQLJ functions have been defined. The system has been illustrated by several query examples.

As a future work, we are planning to give the possibility to the user to modify in the query both the geo-operators and the OSM attributes at the same time. Furthermore, in this context, we are extending the proposed approach to topological relationships involving directed polylines and we are investigating the formal semantics of the related geo-operators.

APPENDIX

Below the types and the GeoPQLJ functions are given in the case of the polygon-polyline, polygon-polygon, and polyline-polyline topological relationships.

Types:

```
const point = {
  "type": "Feature",
  "properties": ,
"geometry": {
 "type": "Point",
 "coordinates": [[x, y]]
   } }
const line = {
"type": "Feature",
"properties": ,
"geometry": {
"type": "LineString",
"coordinates":[[x1, y1], [x2, y2], ...,[xn, yn]]
   }  }
const polygon = {
"type": "Feature",
"properties": ,
"geometry": {
"type": "Polygon",
"coordinates": [[x1, y1], [x2, y2], ..., [xn, yn], [x1, y1]]
   }  }
```

In the following functions, SGO_1 and SGO_2 are generalizations of the geometric types defined above:

1) declare function geopqlj_dsj:disjoint(SGO_1, SGO_2) {GeoPQLJ:booleanQuery(SGO_1,SGO_2,"GeoPQLJ")}
where the function *disjoint* is defined as follows:

```
function disjoint(geom1, geom2) {
switch (geom1.type) {
case "LineString":
switch (geom2.type) {
case "LineString":
return !isLineOnLine(geom1, geom2);
case "Polygon":
return !isLineInPoly(geom1, geom2);
}
break;
case "Polygon":
switch (geom2.type) {
case "LineString":
return !isLineInPoly(geom1, geom2);
case "Polygon":
return !isPolyInPoly(geom1, geom2);
 }  }  }
```

2) declare function geopqlj_inc:inclusion(SGO_1, SGO_2) {GeoPQLJ:booleanQuery(SGO_1, SGO_2, "GeoPQLJ") }
where the function *inclusion* is defined as follows:

```
function inclusion(geom1, geom2) {
switch (geom1.type) {
case "LineString":
switch (geom2.type) {
case "LineString":
return isLineOnLine(geom1, geom2);
case "Polygon":
return isLineInPoly(geom1, geom2);
}
break;
case "Polygon":
switch (geom2.type) {
case "LineString":
return isLineInPoly(geom2, geom1);
case "Polygon":
return isPolyInPoly(geom2, geom1);
} } }
```

3) declare function geopqlj_tch:touch(SGO_1, SGO_2) {GeoPQLJ:booleanQuery(SGO_1, SGO_2,"GeoPQLJ") }
where the function *touch* is defined as follows:

```
function touch(geom1, geom2) {
switch (geom1.type) {
case "LineString":
switch (geom2.type) {
case "Polygon":
return isPointOnLineEnd(geom1, geom2);
}
break;
 case "Polygon":
switch (geom2.type) {
case "LineString":
return isPointOnLineEnd(geom2, geom1);
} } }
```

4) declare function geopqlj_int:intersect(SGO_1,SGO_2) {GeoPQLJ:booleanQuery(SGO_1, SGO_2,"GeoPQLJ")
where the function *intersect* is defined as follows:

```
function intersect(geom1, geom2) {
switch (geom1.type) {
case "LineString":
switch (geom2.type) {
case "Linestring":
return !booleanDisjoint(geom1, geom2);
case "Polygon":
return !booleanDisjoint(geom1, geom2);
}
break;
case "Polygon":
switch (geom2.type) {
case "LineString":
return !booleanDisjoint(geom2, geom1);
case "Polygon":
return !booleanDisjoint(geom2, geom1);
} } }
```

5) declare function geopqlj_pth:passthrough(SGO_1, SGO_2) {GeoPQLJ:booleanQuery (SGO_1, SGO_2,"GeoPQLJ")
where the function *passthrough* is defined as follows:

```
function passthrough(geom1, geom2) {
switch (geom1.type) {
case "LineString":
switch (geom2.type) {
case "LineString"':
return doLineStringsCross(geom1,geom2);
case "Polygon"':
return doLineStringAndPolygonCross(geom1, geom2);
}
break;
case "Polygon":
switch (geom2.type) {
case "LineString":
return doLineStringAndPolygonCross(geom2, geom1);
} } }
```

6) declare function geopqlj_eql:equal(SGO_1, SGO_2) {GeoPQLJ:booleanQuery (SGO_1, SGO_2,"GeoPQLJ")
where the function *equal* is defined as follows:

```
function passthrough(geom1, geom2) {
switch (geom1.type) {
case "LineString":
switch (geom2.type) {
case "LineString"':
return IsEqual(geom1,geom2);
break;
case "Polygon":
switch (geom2.type) {
case "Polygon":
return IsEqual(geom2, geom1);
} } }
```

REFERENCES

[1] (2019). *GeoJSON*. [Online]. Available: https://www.geojson.org
[2] (2019). *OGC*. [Online]. Available: https://www.opengeospatial.org/standards
[3] (2019). *OSM*. [Online]. Available: https://www.openstreetmap.org
[4] (2019). *Planet.osm*. [Online]. Available: https://wiki.openstreetmap.org/wiki/Planet.osm
[5] (2019). *Turf.js*. [Online]. Available: https://www.turfjs.org
[6] (2019). *SFSQL*. [Online]. Available: https://www.vividsolutions.com/jts/JTSHome.htm
[7] (2020). *WordNet*. [Online]. Available: http://wordnet.princeton.edu
[8] J. M. Almendros-Jiménez and A. Becerra-Terón, "Querying open street map with XQuery," in *Proc. 1st Int. Conf. Geographical Inf. Syst. Theory, Appl. Manage.*, Apr. 2015, pp. 28–30.
[9] J. M. Almendros-Jiménez, A. Becerra-Terón, and M. Torres, "Integrating and querying OpenStreetMap and linked geo open data," *Comput. J.*, vol. 62, no. 3, pp. 321–345, Mar. 2019.
[10] P. Amirian, A. Basiri, and A. Winstanley, "Evaluation of data management systems for geospatial big data," in *Computational Science and its Applications—ICCSA* (Lecture Notes in Computer Science), vol. 8583, B. Murgante, S. Misra, A. M. A. C. Rocha, C. Torre, J. G. Rocha, M. I. Falcão, D. Taniar, B. O. Apduhan, and O. Gervasi, Eds. New York, NY, USA: Springer-Verlag, 2014, pp. 678–690.
[11] E. M. Aoidh, M. Bertolotto, and D. C. Wilson, "Understanding geospatial interests by visualizing map interaction behavior," *Inf. Visualizat.*, vol. 7, nos. 3–4, pp. 275–286, Sep. 2008.
[12] J. J. Arsanjani, A. Zipf, P. Mooneyand, and M. Helbich, "An introduction to OpenStreetMap in geographic information science: Experiences, research, and applications, in *OpenStreetMap in GIScience* (Lecture Notes in Geoinformation and Cartography), J. J. Arsanjani, A. Zipf, P. Mooney, and M. Helbich, Eds. Cham, Switzerland: Springer, 2015, pp. 1–15.
[13] L. G. Azevedo, G. Zimbrão, and J. M. de Souza, "Approximate query processing in spatial databases using raster signatures," in *Advances in Geoinformatics*, C. A. Davis and A. M. V. Monteiro, Eds. Berlin, Germany: Springer, 2007, pp. 69–86.

[14] A. Ballatore, M. Bertolotto, and D. C. Wilson, "Geographic knowledge extraction and semantic similarity in OpenStreetMap," *Knowl. Inf. Syst.*, vol. 37, no. 1, pp. 61–81, Oct. 2013.

[15] R. Battle and D. Kolas, "Enabling the geospatial semantic Web with parliament and GeoSPARQL," *Semantic Web*, vol. 3, no. 4, pp. 355–370, 2012.

[16] H. Bo and L. Hui, "Design of a query language for accessing spatial analysis in the Web environment," *GeoInformatica*, vol. 3, no. 2, pp. 165–183, 1999.

[17] O. Boucelma and F. G. Colonna, "GQuery: A query language for GML," in *Proc. 24th Urban Data Manage. Symp.*, 2004, pp. 118–126.

[18] T. Bray, *The Javascript Object Notation (JSON) Data Interchange Format*, document RFC 7159, RFC Editor, Mar. 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7159.txt

[19] M. Calautti, L. Libkin, and A. Pieris, "An operational approach to consistent query answering," in *Proc. 37th ACM SIGMOD-SIGACT-SIGAI Symp. Princ. Database Syst.*, May 2018, pp. 239–251.

[20] D. Calcinelli and M. Mainguenaud, "Cigales, a visual query language for a geographical information system: The user interface," *J. Vis. Lang. Comput.*, vol. 5, no. 2, pp. 113–132, Jun. 1994.

[21] E. Clementini, F. P. Di Felice, and O. P. van Oosterom, "A small set of formal topological relationships suitable for end-user interaction," in *Advances in Spatial Databases* (Lecture Notes in Computer Science), vol. 692. New York, NY, USA: Springer-Verlag, 1993, pp. 277–295.

[22] M. J. Egenhofer, "Reasoning about binary topological relations," in *Proc. 2nd Int. Symp. Large Spatial Databases (SSD)* in Lecture Notes in Computer Science, vol. 525. New York, NY, USA: Springer-Verlag, 1991, pp. 143–160.

[23] F. Ferri and M. Rafanelli, "GeoPQL: A geographical pictorial query language that resolves ambiguities in query interpretation," in *Journal on Data Semantics III* (Lecture Notes in Computer Science), vol. 3534. New York, NY, USA: Springer-Verlag, 2005, pp. 50–80.

[24] F. Ferri, E. Pourabbas, and M. Rafanelli, "The syntactic and semantic correctness of pictorial configurations to query geographic databases by PQL," in *Proc. ACM Symp. Appl. Comput. (SAC)*, Madrid, Spain, 2002, pp. 432–437.

[25] A. Formica, M. Mazzei, E. Pourabbas, and M. Rafanelli, "Approximate answering of queries involving polyline–polyline topological relationships," *Inf. Visualizat.*, vol. 17, no. 2, pp. 128–145, Apr. 2018.

[26] A. Formica, M. Mazzei, E. Pourabbas, and M. Rafanelli, "Querying distributed GIS with GeoPQLJ based on GeoJSON," in *Proc. 5th Int. Conf. Geographical Inf. Syst. Theory, Appl. Manage.*, Heraklion, Greece, May 2019, pp. 175–182.

[27] A. Formica and E. Pourabbas, "Content based similarity of geographic classes organized as partition hierarchies," *Knowl. Inf. Syst.*, vol. 20, no. 2, pp. 221–241, Aug. 2009.

[28] A. Formica, E. Pourabbas, and M. Rafanelli, "Constraint relaxation of the polygon-polyline topological relation for geographic pictorial query languages," *Comput. Sci. Inf. Syst.*, vol. 10, no. 3, pp. 1053–1075, 2013.

[29] R. Grove, J. Wilson, D. Kolas, and N. Wiegand, "GeoSPARQL query tool—A geospatial semantic Web visual query tool," *WEBIST*, vol. 2, pp. 33–40, Apr. 2014.

[30] J. Guan, F. Zhu, J. Zhou, and L. Niu, "GQL: Extending XQuery to query GML documents," *J. Geo-Spatial Inf. Sci.*, vol. 9, no. 2, pp. 118–126, 2006.

[31] P. Hashemi and R. A. Abbaspour, "Assessment of logical consistency in OpenStreetMap based on the spatial similarity concept," in *OpenStreetMap in GIScience: Experiences, Research, and Applications* (Lecture Notes in Geoinformation and Cartography), J. J. Arsanjani, A. Zipf, P. Mooney, and M. Helbich, Eds. Cham, Switzerland: Springer, 2015, pp. 19–36.

[32] C.-H. Huang, T.-R. Chuang, D.-P. Deng, and H.-M. Lee, "Building GML-native Web-based geographic information systems," *Comput. Geosci.*, vol. 35, no. 9, pp. 1802–1816, Sep. 2009.

[33] T. Kapler and W. Wright, "GeoTime information visualization," *Inf. Visualizat.*, vol. 4, no. 2, pp. 136–146, Jun. 2005.

[34] W. Kuhn, "Metaphors create theories for users," in *Spatial Information Theory A Theoretical Basis for GIS* (Lecture Notes in Computer Science), vol. 716, A. U. Frank and I. Campari, Eds. New York, NY, USA: Springer-Verlag, 1993, pp. 366–376.

[35] Z. Xie, L. Wu, Z. Chen, and X. Wu, "Distributed geographic structure query language-DGSQL," *J. Softw. Eng.*, vol. 9, no. 2, pp. 328–336, Feb. 2015.

[36] D. Lin, "An information-theoretic definition of similarity," in *Proc. 15th Int. Conf. Mach. Learn.*, J. W. Shavlik, Ed. Madison, WI, USA: Morgan Kaufmann, 1998, pp. 296–304.

[37] F. Mandreoli, R. Martoglia, G. Villani, and W. Penzo, "Flexible query answering on graph-modeled data," in *Proc. 12th Int. Conf. Extending Database Technol. Adv. Database Technol. (EDBT)*, 2009, pp. 216–227.

[38] R. M. Olbricht, "Data retrieval for small spatial regions in OpenStreetMap," in *OpenStreetMap in GIScience* (Lecture Notes in Geoinformation and Cartography), J. J. Arsanjani, A. Zipf, P. Mooney, and M. Helbich, Eds. Cham, Switzerland: Springer, 2015, pp. 101–122.

[39] E. Pourabbas and A. Shoshani, "Efficient estimation of joint queries from multiple OLAP databases," *ACM Trans. Database Syst.*, vol. 32, no. 1, p. 2, Mar. 2007.

[40] B. Regalia, K. Janowicz, and G. McKenzie, "Computing and querying strict, approximate, and metrically refined topological relations in linked geographic data," *Trans. GIS*, vol. 23, no. 3, pp. 601–619, Jun. 2019.

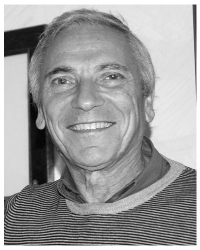
[41] S. S. Sriparasa, *JavaScript and JSON Essentials*. Birmingham, U.K.: Packt, 2013.

[42] N. Wiegand, R. Grove, J. Wilson, and D. Kolas, "Querying geospatial data over the Web: A GeoSPARQL interface," in *Proc. GeoRich*, 2014, pp. 1–6.

ANNA FORMICA received the degree (Hons.) in mathematics from the University of Rome "La Sapienza," in 1989. She is currently a Researcher with the "Istituto di Analisi dei Sistemi ed Informatica" (IASI) "Antonio Ruberti," Italian National Research Council (Consiglio Nazionale delle Ricerche-CNR), Rome, where she works with the Software and Knowledge-based Systems (SaKS) Group. She serves as a Referee of several international journals and conferences and she took part in various research projects of the European Framework Programs and Bilateral Projects with international institutions. Her current research interests are semantic web, similarity reasoning, formal specification and validation of domain ontologies, geographical information systems, e-learning, formal concept analysis, fuzzy sets, and rough set theory.

MAURO MAZZEI received the master's degree in civil and environmental engineering, and the degree in geomatics from the University of Rome "Guglielmo Marconi". He is currently a Researcher with the "Istituto di Analisi dei Sistemi ed Informatica" (IASI) "Antonio Ruberti," Italian National Research Council (Consiglio Nazionale delle Ricerche-CNR), Rome, where he works with the Software and Knowledge-based Systems (SaKS) Group. He is part of the pool of experts on the European INSPIRE Program (Infrastructure for Spatial Information in Europe). He published several research articles in international journals and conferences, and he took part in various research projects of the European Framework Programs and Bilateral Projects with international institutions. His current research interests are: analysis and modeling of domains that relate geographic information science in the open source field, definitions of spatial data infrastructure and co-design architectures; analysis, study and testing of service-oriented architecture (soa) systems.

ELAHEH POURABBAS received the M.S. degree in electrical engineering from the University of Rome "La Sapienza," in 1992, and the Ph.D. degree from the University of Bologna, in 1997. She is currently a Research Scientist with the Istituto di Analisi dei Sistemi ed Informatica (IASI) "Antonio Ruberti," Italian National Research Council. In 2005, she received the Fulbright Fellowship in support of research carried out at the University of California-Lawrence Berkeley National Laboratory, Berkeley, USA. She served as a Referee of several international journals and conferences. She has taken part in various research projects of the European Framework Programs and bilateral projects with international institutions. Her research interests include query processing, data analysis and management, GIS, data warehousing and OLAP, semantic web, and similarity reasoning.



MAURIZIO RAFANELLI received the degree in mathematics from the University of Rome "La Sapienza," in 1976. He is currently an Associate Researcher with the "Istituto di Analisi dei Sistemi ed Informatica" (IASI) "Antonio Ruberti," Italian National Research Council (Consiglio Nazionale delle Ricerche-CNR), Rome, where he works with the Information Systems and Knowledge Representation Group. In 1988 and 1998, he organized as the Program Chair and the General Chairman, the IV International Working Conference on Statistical and Scientific Database Management and the X International Conference on Scientific and Statistical Database Management, respectively. He serves as a Referee of several international journals and conferences. His current research interests are geographical information systems and advanced query languages.