OnGIS: Ontology Driven Geospatial Search and Integration

Marek Šmíd and Zdeněk Kouba

Faculty of Electrical Engineering, Czech Technical University in Prague
smidmare@fel.cvut.cz,kouba@fel.cvut.cz

Abstract. Helping non-expert users to query over complex spatial data requires abstracting from GIS services and operations. This paper introduces an abstraction layer based on OWL ontologies. It provides an intuitive searching GUI for exploring data semantically integrated from different sources. The language OWL 2 QL has been chosen as it performs open-world semantic web reasoning suitable for data integration, while having good computational properties. To support spatial constraints, a custom OWL 2 QL reasoner has been designed as part of this work. It allows relating different objects by means of spatial joins. A prototype system backed by the custom reasoner has been tested on the urban planning domain.

Keywords: Geospatial semantics, OWL 2 QL, Data integration

1 Introduction

The problem OnGIS tries to solve is a semantic search over spatial data, which is motivated by the need of presenting GIS services to non-expert users (with neither technical nor GIS background) in a simple way, still allowing him/her to perform complex queries. The use case at the end of this section shows that it can help a user to obtain relevant information from a complex GIS server, which would be otherwise difficult without understanding the GIS domain.

Another advantage of using semantic technologies for the search is that it can be crafted to work with multiple data sources, which allows for filtering by relations between objects from different sources.

Semantics of the queries is supported by an OWL 2 QL [1] reasoner (see Section 3), which supports e.g. sub-class, sub-property, domain and range axioms. For example, when there is an axiom saying that class *Places of Worship* in an ontology has a sub-class *Churches* in another ontology, it helps to find instances of *Churches* from the latter ontology, when a user asks for instances of the *Places of Worship* class.

A more complex example proving that OWL 2 QL goes beyond RDFS[1] expressivity consists of the following two axioms: *Churches* is a sub-class of the range of the *marriageTakesPlaceIn* object property, and *marriageTakesPlaceIn*

[1] http://www.w3.org/TR/rdf-schema/, cit. 24.7.2012

is a sub-property of the inverse of the *hostsSocialEvent* object property. Then, asking for all places that host social events (i.e. querying the domain of *hostsSocialEvent*) retrieves also the instances of *Churches*.

The main advantage of OWL 2 QL over OWL 2 DL is its tractability, which allows for storing instances in a relational database and posing ontological queries[2] by means of query reformulation to SQL[3]. This makes it possible to store large amounts of data, which is typical in GIS systems, and reason over them. Still, OWL 2 QL is quite powerful (has class and role hierarchies, limited negation, etc.) and it keeps the open world assumption. It is also well settled as a W3C standard.

A motivation, where OnGIS can be used, is to visualize data and maps of the department of urban planning of Prague (the capital of the Czech Republic), being a part of URM (Útvar rozvoje hlavního města Prahy[4]) — City Development Authority of Prague.

URM collects many data sets obtained from various government institutions, taking care of e.g. pollution, noise, flood risks, and land prices. URM groups these data sets into map layers and services, which are available to the public for various analyses (e.g. setting prices for real estate companies, finding a suitable site to build a house for a family). But for a user, who is not a GIS expert, it is not easy to search in a catalogue of GIS services, to find relevant map layers and to work with them.

Thus, the metadata of URM GIS services were extracted (mostly by querying ArcGIS[5] SOAP web services), and stored into an ontology. This ontology was annotated with the OnGIS annotations (see Section 4), making it possible for OnGIS to search it and display the URM GIS services in a map.

2 Related Work

2.1 Linking Ontologies with Databases and Semantic Integration

There are quite many systems for linking ontologies to databases, here are some examples implementing OWL 2 QL or another $DL{-}Lite$ language [2] (see Section 3) that can link to databases or other efficient storages, e.g. QuOnto[6] [3], ROWLKit [4], Mastro [5], OWLIM[7] (OWLIM-SE has spatial support according to [6], but very limited), Stardog[8]. None of the mentioned systems supports (to our knowledge) complex spatial queries.

[2] Queries consisting of elements with defined meanings given by an ontology, and relations between the elements.

[3] This technique of query answering is called Ontology-Based Data Access (OBDA), which uses an ontology as a mediator to access non-semantic data.

[4] `http://www.urm.cz/`, cit. 24.7.2012

[5] A series of GIS software by ESRI, see `http://www.esri.com/software/arcgis`, cit. 24.7.2012

[6] `http://www.dis.uniroma1.it/~quonto/`, cit. 7.10.2011

[7] `http://www.ontotext.com/owlim`, cit. 24.7.2012

[8] `http://stardog.com/`, cit. 24.7.2012

For mapping ontology entities to a database, we have chosen a set of our own simple OWL annotations. It was impossible to reuse existing mapping approaches, e.g. in D2RQ [7], because they do not fit the OnGIS needs, since they do not have some required features (e.g. filtering table results) and they do not allow for needed optimizations (e.g. query containment).

There has been many works on semantic integration ([8] gives an overview). An example is CARIN [9]. The CARIN family of languages combines Horn rules and description logics. It deals with designing a sound and complete inference procedure for answering queries. An application of CARIN, Information Manifold, as presented in [10], serves a similar purpose as OnGIS — information gathering providing uniform access to multiple structured information sources. It uses materialized database views that guarantee accessing only relevant sources. However, Information Manifold does not deal with spatial data sources. C-OWL [11] is an interesting approach for linking semantic data using contextualized ontologies based on OWL.

General database integration tools are not suitable, though some of them support spatial data integration, since they lack a semantic layer.

2.2 GIS Querying Systems

The authors of [12] propose a system for mapping ontology axioms to SQL queries on a database with the focus on geospatial data. Though using ontologies, it does not rely on OWL or any other reasoning. Also it does not focus on multiple data sources.

The system DO-ROAM [13] is quite similar to our OnGIS. It is a web service that focuses on finding places according to activities that a person could perform there. It uses its own OBDA system, which maps ontology concepts and properties to database queries in quite a simple way. The implemented OBDA system is probably not a general OWL reasoner. Again, it does not support multiple data sources, and is not general enough to support reasoning over different domains.

The ontology-based information system in [14] focuses on spatio-thematic query answering for city maps. Its reasoner implements a very expressive logic that has some features OWL 2 DL has not (and vice versa). The system implements its own custom storage, which directly includes the inference algorithms and the query evaluation engine. Spatial data in an ABox[9] are represented e.g. as RCC relations only, or by using a special spatial ABox. But it does not integrate multiple data sources.

The system in [15] links an RDF[10] ontology to databases and WFS[11]. It uses custom rules and algorithms for query rewriting, but it does not provide the standard OWL semantics. However, it supports query answering from multiple data sources, specifically WFS servers for spatial data and databases (via the D2R interface) for attributes.

[9] A set of all ontology axioms about individuals — assertions.

[10] `http://www.w3.org/RDF/`, cit. 24.7.2012

[11] Web Feature Service, an Open Geospatial Consortium (OGC) standard, see `http://www.opengeospatial.org/standards/wfs/`, cit. 24.7.2012

The spatial decision support system in [16] integrates various data sources (OGC standards WMS, WFS, WCS, WPS) and links them with ontologies. It also uses catalogue services via ontologies and automatic web service discovery. However, it focuses more on geospatial analysis and ontology alignment than spatial search.

The authors in [6] use Parliament triple store, supporting geospatial indexes, for storing spatial data and for making complex spatial queries via GeoSPARQL (see Section 3.2) over them. However they use a precomputed data set and do not directly support data integration.

3 OWL 2 QL

OWL 2 QL [1] is a profile of the Web Ontology Language (OWL). The key feature is its tractability (along with other OWL 2 profiles) traded for expressiveness, which is lower compared e.g. to OWL 2 DL. The tractability brings the advantage that description logic queries can be reformulated into SQL and thus RDBMSs (relational database management systems) can be used as OWL 2 QL storage.

OWL 2 QL is based on $DL{-}Lite^{\mathcal{H}}_{core}$, a member of the $DL{-}Lite$ language family defined (in its extended version) in [2], being in turn a member of the description logics family [17]. $DL{-}Lite^{\mathcal{H}}_{core}$ constructs for defining concepts and roles in description logics syntax are:

$$B ::= A \,|\, \exists R\,, \qquad C ::= B \,|\, \neg B\,, \qquad R ::= P \,|\, P^-\,,$$

where A denotes a concept name, B a basic concept, and C a general concept. Symbol P denotes a role name, and R a complex role.

The semantics is defined by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a nonempty interpretation domain and $\cdot^{\mathcal{I}}$ is an interpretation function, that assigns to each individual an element of $\Delta^{\mathcal{I}}$, to each concept name a subset of $\Delta^{\mathcal{I}}$, and to each role name a binary relation over $\Delta^{\mathcal{I}}$.

Semantics of the used constructs are defined in Table 1.

Table 1. Constructs used in $DL{-}Lite$ and their semantics

Syntax	Semantics	Comment
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	concept name
P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	role name
P^-	$(P^-)^{\mathcal{I}} = \{(b,a) \mid (a,b) \in P^{\mathcal{I}}\}$	inverse of a role
$\exists R$	$(\exists R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b : (a,b) \in R^{\mathcal{I}}\}$	existential quantification
$\neg B$	$(\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$	negation of a basic concept

A TBox[12] can be defined by inclusion axioms of the form: $B \sqsubseteq C$, and $R_1 \sqsubseteq R_2$, interpreted by $\mathcal{I}$ as $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, resp. $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$.

[12] A set of all ontology terminological axioms — subsumptions of concepts, domains, etc.

An ABox consists of the following assertion axioms: $A(a)$, and $P(a, b)$, where a, b are individuals interpreted by $\mathcal{I}$ as $a^{\mathcal{I}}, b^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The axioms are interpreted by $\mathcal{I}$ as $a^{\mathcal{I}} \in A^{\mathcal{I}}$, resp. $\left(a^{\mathcal{I}}, b^{\mathcal{I}}\right) \in P^{\mathcal{I}}$.

OWL 2 QL extends $DL{-}Lite$ with various features not affecting its tractability, e.g. data roles.

3.1 Owlgres

Owlgres[13] [18] is an open source Java implementation of a $DL{-}Lite^{\mathcal{H}}_{core}$ reasoner developed by Clark & Parsia[14], backed by a DBMS for persisting data (it is tailored to work with PostgreSQL[15] databases).

The original database schema used in Owlgres has two sets of database tables, one for TBox, with one table listing all classes, one for all object properties, etc., and one set for ABox. This original schema for ABox stores all class assertions in one table, all object property assertions in another table, and similarly for data properties and annotations.

3.2 OwlgresMM

In [19], two other schemas are designed, implemented and compared to the original one in Owlgres. The result is that for most cases, storing class assertions (resp. object and data property assertions) in separate tables per named class (resp. named object and data property), which is one of the new schemas, is the most efficient option. OnGIS uses OwlgresMM, which is our extension of Owglres based on this schema.

Another feature of OwlgresMM is that it supports simple annotations for defining how the classes and properties are mapped into a database. But the key quality is it can work with multiple databases. After input query reformulation, it distributes the query among multiple databases that are linked from the imported TBoxes, and it generates the SQL queries to the ones suitable to answer at least a part of the query (i.e. containing data, which can help spatially restricting the query results).

It partially supports GeoSPARQL [20], an OGC[16] standard for geospatial queries and simple geometry manipulation, extending SPARQL[17], a query language for RDF. GeoSPARQL was an inspiration for spatial filters (e.g. within, within distance, and bounding box) and geometry accessors (e.g. geometry, centroid, and area) implementation, with full compliance pending. There are more features implemented, like support for aggregations, various filters, etc.

OwlgresMM is tailored to work with PostgreSQL with PostGIS extension[18].

[13] `http://pellet.owldl.com/owlgres`, cit. 7.10.2011
[14] `http://clarkparsia.com/`, cit. 7.10.2011
[15] `http://www.postgresql.org/`, cit. 24.7.2012
[16] Open Geospatial Consortium, `http://www.opengeospatial.org/`, cit. 24.7.2012
[17] `http://www.w3.org/TR/rdf-sparql-query/`, cit. 24.7.2012
[18] PostGIS (`http://postgis.refractions.net/`, cit. 24.7.2012) adds support for spatial data to PostgreSQL. It allows efficient storage, indexing, and retrieval of geographical data in database tables.

4 Design

OnGIS tries to follow typical web search scenarios. A user is not used to give a search engine a structured query, but instead enters a few keywords, and examines the results the query engine gives back. Then, the user picks the relevant search results, which are added to the list of items to be displayed. This sequence can be repeated several times, until the user fills the list with all the items he/she wants to be displayed.

The front-end part of OnGIS (see Fig. 1) consists of a modular web application. The key aspect of the web application is that it does not depend on any domain specific data structure nor on any technique of obtaining data. The way, how to obtain data, is provided by plugins. Each plugin has to conform to an API, which allows for user query answering, for providing geometries, and for displaying layers. Plugins can support searching over ontology entities, databases, etc., and displaying layers e.g. from PostGIS databases, WMS[19] and ArcGIS servers.

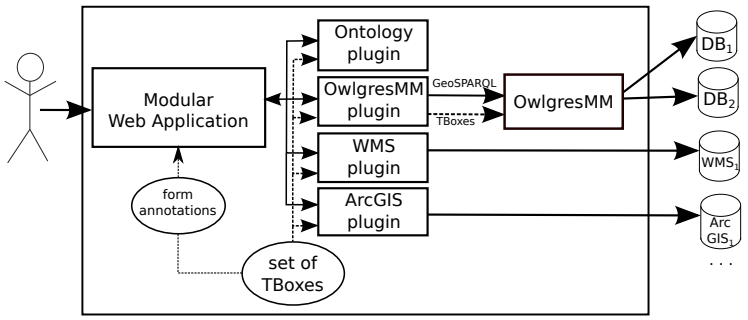

Fig. 1. Overall architecture.

Independence of the web application on domain structure is ensured by using the following specific set of OWL annotations, which serve as an interface between the domain specific ontologies and the web application.

searchable specifies a data property, which should be searched when a user enters a query.

geometry specifies an entity representing spatial geometries — it is useful for spatial queries.

filterable specifies, if a data property is suitable for filtering — it is useful for filtering spatial features (e.g. by attribute queries) to be displayed on a map.

partof specifies relation between entities by an object property, defining part-of relation — it is useful for linking an object to its integral components.

[19] An OGC standard of a simple mechanism for obtaining raster maps, see `http://www.opengeospatial.org/standards/wms/`, cit. 24.7.2012.

priority specifies a numeric priority of entities, which affects the order in which the entities should appear in the search result list.

displayable specifies an entity, which can be displayed on a map as a layer. It may have annotation sub-properties, that are handled by different OnGIS plugins, e.g. for Postgis, WMS, ArcGIS, and ArcGIS RESTful map servers.

An example of the annotations used on the URM domain is illustrated in Fig. 2. Only a part is displayed, what is missing is e.g. the *searchable* annotation marking the URM domain data properties "hasKeywords", "hasDefinition", "hasDescription", etc.

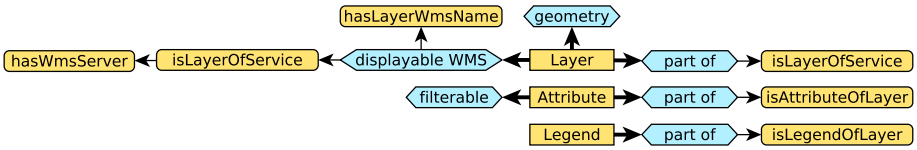

Fig. 2. Illustration of part of OnGIS annotations (the blue diamonds) on URM domain terms (all yellow rectangles). The thick arrows represent annotating, the thin arrows point to annotation values (which are round rectangles). E.g. the "part of" annotation links to an object property, which can be used to relate instances of the annotated class to its parts.

A key advantage of OnGIS compared to other similar systems is that it can spatially relate search results. The relations between objects can be entered in two ways:

simple The simple way is to add a spatial restriction to a search result. Currently, only two restrictions are possible: *inside* restriction and *distance* restriction. The inside restriction filters all other search results, so that they have to be contained inside the search result defining the restriction. The distance restriction is similar — all other search results have to be within the specified distance. These restrictions can be used on more than one search results, but not all plugins need to support multiple restrictions.

by links Another way of specifying relations is by defining links, which is meant for advanced users. A link can be established between two search results, which filters both search results in such a way that they have to be within the specified distance (pair-wise, one feature corresponding to one search result entity, the second feature corresponding to the other search result entity). More link types are to be designed.

A query consists of ontology entities found by the "searchable" annotated properties. In case of the entities annotated as "geometries", in the simple mode, a user can attach the *inside* and the *distance* restrictions to the entities. The semantics of the restrictions is that for each entity attached with one, it restricts

all other "geometry" entities. It is performed in a recursive fashion, e.g. for entities A, B, C, where B, C have some restrictions, then A is restricted by B, which is restricted by C, and also A is restricted by C, which is restricted by B; the cycles in the tree are cut. The restrictions do not apply for "single object" geometries, which are as such reported by OnGIS plugins (not groups of objects, but specific objects, that are directly sought for by the user, and it does not make sense to filter them).

In case of the entities annotated as "filterable", the user can enter filter expressions, which are applied to automatically added (if not already present) entities, of which the "filterable" entities are "part of". The semantics of the links mode is simply restricting only the entities involved in the links. When the query is evaluated, the "geometry" entities annotated with one of the "displayable" annotations are shown on a map.

5 Proof of Concept

An OnGIS prototype has been built as a Java EE[20] application. The prototype provides a web interface for searching and displaying maps using OpenLayers[21].

There are four plugins implemented and used:

Owlapi plugin is used for searching over ontologies via the OWLAPI interface. It performs search only, no layer retrieval nor spatial queries are supported.

OwlgresMM plugin is a connector to OwlgresMM. It receives query requests from the web interface, constructs a SPARQL query and queries appropriate databases using OwlgresMM. It also handles *displayablePostgis* annotation, for which it fetches geometries from appropriate PostGIS databases and generates a layer.

WMS plugin is used for displaying layers in the map from WMS servers. It is driven by *displayableWms* annotation.

ArcGIS plugin is used for displaying layers in the map from ArcGIS servers. It is driven by *displayableArcgis* and *displayableArcgisRest* annotations.

The prototype has been tested with OwlgresMM connecting to two databases, one with OpenStreetMap[22] data, and the other with GeoNames[23] data.

OpenStreetMap is publicly available geographical data of the World. We implemented a special utility, which imports the data to our own spatially enabled database in a suitable format — each category of features in a separate table, and generates an ontology to database tables mapping. As the ontology for OpenStreetMap data, LinkedGeoData[24] ontology was used, which is a part of Linked Data. Its expressivity is quite limited, and it is no problem to fit it into

[20] `http://docs.oracle.com/javaee/`, cit. 24.7.2012

[21] An open-source JavaScript library for displaying and interacting with maps from many raster and vector sources, see `http://openlayers.org/`, cit. 24.7.2012.

[22] `http://www.openstreetmap.org/`, cit. 24.7.2012

[23] `http://www.geonames.org/`, cit. 24.7.2012

[24] `http://linkedgeodata.org/`, cit. 24.7.2012

the OWL 2 QL expressivity. GeoNames is a geographical database of points. However, it has not been used in the example shown below.

The semantics of crowdsourced data, like OpenStreetMap and GeoNames, is not always very precise, compared to the professional data, as from URM, with more accurate semantics. Therefore, it would be useful to employ data available from other local authorities, e.g. cadastral offices.

As an example of many experiments performed (involving various natural objects, ways, city places, etc.), let us search for places of worship with two restrictions: they have to be close to a park (with maximum distance of 100 m), and they have to be inside a specific part of Prague (borough named "Praha 2"). Such queries, integrating the URM services and other data, may help citizens in making their housing decisions. The OnGIS system connects all the data mentioned earlier: URM GIS services, OpenStreetMap and GeoNames data.

The query has to be posed to the system in an understandable way. One way would be to write it as a piece of text with simple rigid structure, with its terms having known meanings. We chose another one, constructing the query iteratively by adding objects in the query (e.g. a place, a group of places, an attribute) one by one. A user enters a keyword for each object in the query first, and from the results found he/she picks the appropriate one, which is added to a list. As an example, we entered the keyword "worship", which found the class "Place of Worship" from OpenStreetMap ontology, that was added to the final query object list, as in Fig. 3. We followed entering the rest of the terms (see the caption of the figure).

Fig. 3 also shows the appropriate filter fields in the list. For example, the *name* attribute is an instance of the class Attributes in the domain specific URM GIS services ontology. This class is annotated with the general OnGIS *filterable* annotation, hence it does make sense to provide the user with text input field for filtering. We entered "Praha 2", the requested borough here. The other objects are either URM GIS layer instances (boroughs) or OpenStreetMap classes (parks and places of worship), which are all annotated as being both displayable (thus they are shown in the map in Fig. 4) and as being geometries (suitable for performing spatial queries). Thus these items contain fields for filtering via maximum distance to them and for filtering other items only to those being inside of them. Therefore, we ticked the inside restriction of boroughs, and entered "100" (meters) to the max. distance restriction of parks.

Fig. 4 contains the results of the query. It is a detail of a map displayed in OpenLayers, having URM technical land usage layers in the background (with the river Moldau on the left-hand side). The colors correspond to the result list in Fig. 3, thus the parks are blue, and the places of worship are red. The display of boroughs was disabled, to make the map more legible.

6 Conclusion and Future Work

The designed ontology based system, OnGIS, is suitable for distributing queries over spatial data to multiple data sources with spatial restrictions among them.

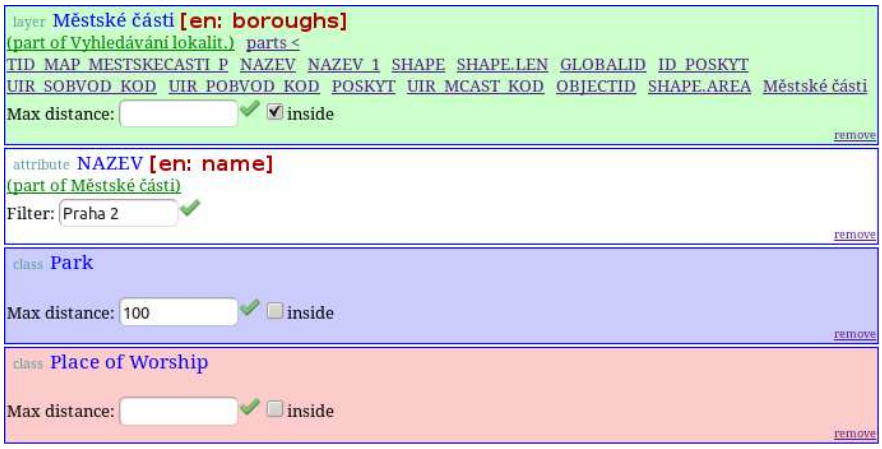

Fig. 3. OnGIS having a few search results added. We are looking for places of worship, thus the "Place of Worship" OpenStreetMap class found by keyword "worship". For the park restriction, we simply sought for "park", which resulted in "Park" OpenStreetMap class. To filter by borough name, it is easier to find boroughs layer first, and then its name attribute (since there are a lot of name attributes of many layers). So we added "Městské části" (meaning boroughs), which is from URM data. Then its parts can be shown in the boroughs item. One of the parts is "název" (meaning name).

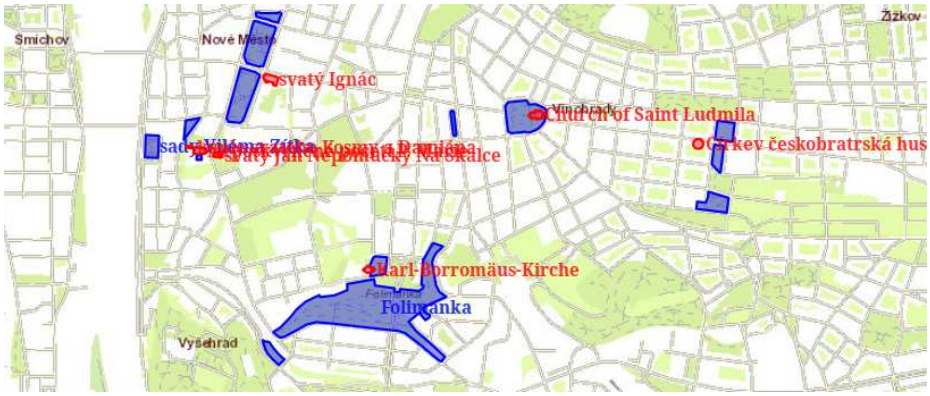

Fig. 4. OnGIS displaying a map with the search results. The blue areas are parks, and the red spots are places of worship (note that they are all close to the parks, as requested by the query), with both appearing only in the borough "Praha 2".

It is based on the idea that the domain-specific data structures linked to the system are described by domain-specific ontologies, that are annotated with general OnGIS annotations, which are the entry points for accessing the data from OnGIS.

A strong aspect of the system is that it can perform filtering by spatial relations between objects — efficiently within one database data source, or even between heterogeneous data sources.

Its prototype supports a few GIS services at the moment: WMS and ArcGIS servers, and spatially-enabled relational databases (specifically Postgres with PostGIS extension).

There are many ways of extending OnGIS. An obvious one is supporting other GIS services via plugins, e.g. plugin for WFS servers (which allow for spatial queries and displaying features). Also a plugin for querying RDF repositories via SPARQL endpoints, possibly supporting GeoSPARQL would help using many data sources (naturally the ones in Linked Data — DBpedia, etc.) would greatly extend the system. The OwlgresMM plugin will be extended to fully support GeoSPARQL to allow compatibility with other systems.

Using non-spatial data would require non-spatial links between objects, which requires an appropriate extension of OnGIS GUI to support it. Also the spatial relation restrictions need extending, with an appropriate semantics designed. Complete query correctness and soundness verification is also necessary.

The current OnGIS query input, iterative addition of single query objects, can be substituted by entering a query via a piece of text with simple rigid structure (an expression not in a natural language, but very restricted) defining objects, and their restrictions and relations. Some users may find it more intuitive and as a quicker way of entering a query. To give the terms in the expression specific meanings, the terms could be picked from a list of suggestions after typing a few first letters (such suggestion lists are widely used by web search engines).

Filtering data based on a spatial relation within a spatially enabled relational database is a computationally expensive operation. Experiments show, that Postgres, when asked to find objects from one table within a distance to objects from another table, uses sequential scan on one table, and spatially indexed access on the other (using the index for object from the sequential scan from the former table). This can take a long time for otherwise unrestricted query over large tables. A solution to alleviate the problem may be to implement seeded tree algorithm according to [21].

Acknowledgments. This work was supported by the CTU research funding and the research programme no. MSM 6840770038 funded by the Czech Ministry of Education.

References

1. World Wide Web Consortium: OWL 2 Web Ontology Language: Profiles, OWL 2 QL, http://www.w3.org/TR/owl2-profiles/#OWL_2_QL. (2009)

2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. of Artificial Intelligence Research **36** (2009) 1–69
3. Acciarri, A., Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005). (2005)
4. Corona, C., Ruzzi, M., Savo, D.F.: Filling the gap between OWL 2 QL and QuOnto: ROWLKit. In: Description Logics '09. (2009)
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The Mastro system for ontology-based data access. Semantic Web Journal **2**(1) (2011) 43–53
6. Battle, R., Kolas, D.: Enabling the geospatial semantic web with Parliament and GeoSPARQL. Semantic Web Journal, to appear
7. Bizer, C., Seaborne, A.: D2RQ - treating non-RDF databases as virtual RDF graphs. In: ISWC2004 (posters). (November 2004)
8. Telang, A., Chakravarthy, S., Huang, Y.: Information integration across heterogeneous sources: Where do we stand and how to proceed? In: COMAD, Computer Society of India / Allied Publishers (2008) 186–197
9. Levy, A.Y., Rousset, M.C.: Combining horn rules and description logics in CARIN. Artif. Intell. **104**(1-2) (1998) 165–209
10. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Query-answering algorithms for information agents. In: AAAI-96. (1996)
11. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: C-OWL: Contextualizing ontologies. In: ISWC. Lecture Notes in Computer Science, Springer Verlag (October 2003) 164–179
12. Baglioni, M., Masserotti, M.V., Renso, C., Spinsanti, L.: Improving geodatabase semantic querying exploiting ontologies. In: GeoSpatial Semantics, Springer (2011)
13. Codescu, M., Horsinka, G., Kutz, O., Mossakowski, T., Rau, R.: DO-ROAM: Activity-oriented search and navigation with OpenStreetMap. In: GeoSpatial Semantics, Springer (2011)
14. Wessel, M., Möller, R.: Flexible software architectures for ontology-based information systems. Journal of Applied Logic – Special Issue on Empirically Successful Computerized Reasoning (2009)
15. Zhao, T., Zhang, C., Wei, M., Peng, Z.R.: Ontology-based geospatial data query and integration. In: GIScience. Volume 5266 of Lecture Notes in Computer Science., Springer (2008) 370–392
16. Zhang, C., Zhao, T., Li, W.: The framework of a geospatial semantic web-based spatial decision support system for digital earth. Int. J. Digital Earth **3**(2) (2010) 111–134
17. Baader, F., Calvanese, D., McGuinness, D.L., Patel-Schneider, P., Nardi, D.: The description logic handbook: theory, implementation, and applications. Cambridge University Press (2003)
18. Stocker, M., Smith, M.: Owlgres: A scalable OWL reasoner. In: OWLED. Volume 432 of CEUR Workshop Proceedings., CEUR-WS.org (2008)
19. Šmíd, M.: Using databases for description logics. Master's thesis, Czech Technical University in Prague, Faculty of Electrical Engineering (2009)
20. Open Geospatial Consortium: OGC GeoSPARQL — A Geographic Query Language for RDF Data, http://www.opengeospatial.org/standards/geosparql. (2012)
21. Lo, M.L., Ravishankar, C.: The design and implementation of seeded trees: an efficient method for spatial joins. Knowledge and Data Engineering, IEEE Transactions on **10**(1) (jan/feb 1998) 136 –152