Semantic Queries Supporting Crisis Management Systems

Manfred Schenk, Tobias Hellmund,
Philipp Hertweck and Jürgen Moßgraber

Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Karlsruhe, Germany
Email: {manfred.schenk, tobias.hellmund,
philipp.hertweck, juergen.mossgraber}@iosb.fraunhofer.de

Abstract—**In crisis management, it is crucial to have up-to-date data available to assess an ongoing situation correctly. This information originates from different sources, such as human observations, various sensors or simulation algorithms with heterogeneous geographic scope. The aggregation of such data is conducted by decision support systems to disburden the end-users from automatable tasks. By using semantic technologies for the integration, these systems can benefit from the expressional power of semantic queries. Therefore, all data that is available in the system should also be accessible for these queries. In the following, we present an approach how sensor data can be accessed through semantic queries and how geospatial knowledge can be integrated in a Decision Support System.**

Keywords–decision support; semantic access to sensor observations; GeoSPARQL; geospatial semantic queries.

I. INTRODUCTION

Being a specialized sub-category of a Decision Support System (DSS), crisis management systems are designed to support authorities in handling a crisis by providing collected and analyzed information as well as simulation results. While in the past the lack of information was one of the biggest challenges, the situation has changed now. Since the advent of the Internet of Things (IoT) placing great numbers of connected sensors the amount of available data has increased [1]. This results in new challenges: The decision makers have to be protected from information overflow [2] and the input from different heterogeneous sources has to be integrated and automatically analyzed.

Why is geospatial knowledge important in crisis management? A crisis situation usually will be limited to a certain area in most cases. When a high level of water in a river causes flooding of the land along the riverbanks, this flooding will be limited to a certain area surrounding the river. The knowledge about this area allows reducing the amount of data which has to be considered in the crisis management workflows. Thus, the location of a sensor or its observation can be used for filtering data that is of no value for the current crisis; the location of a human observer can be used to estimate the validity of a given statement; the geometry of an urban district can be used to determine if the district is affected by some event.

Our approach is to use semantic technologies to integrate information from different sources and then apply reasoning to draw conclusions from the gathered information focusing on the semantics of geospatial data. While Kontopoulos et al. presented the overall approach in [3] and [4], this paper concentrates on two parts of the approach: firstly, accessing of time series data, e.g. sensor observations and, secondly, exploiting geospatial knowledge.

The paper is structured as follows: Related Work introduces preliminaries and gives a brief overview of recent work. In Section 3 - Using geospatial knowledge, the motivation for the semantic integration of geospatial information in a crisis situation given. It is described how geospatial semantics can be integrated in a machine-understandable manner. Further, geoSPARQL [5] with possible applications is introduced. Section 4 - Semantic Access to Sensor Data - gives different approaches how sensors and their geospatial semantics can be integrated into a DSS. We present our approach for the integration and retrieval of such data with the standard SensorThings API [6] and ONT-D2RQ [7]. Section 5 - Use-Case Application describes the application of the developed functionalities within the project beAWARE [8]. We give an exemplaric query and analyze its architecture. Section 6 evaluates the approach on a qualitative base. Section 7 concludes our findings and gives an outlook on future tasks.

II. RELATED WORK

Parts of the work presented here are based on the Open Geospatial Consortium (OGC) standard GeoSPARQL [5]. Battle and Kolas [9] provide a good introduction into GeoSPARQL. Zhang et al. [10] have already considered GeoSPARQL useful in the area of crisis management. While they concentrated on performance improvements of the GeoSPARQL implementations, our focus is how to bring all parts together. Nishanbaev et al. provided a survey of geospatial semantic web for cultural heritage [11]. While the survey focused on cultural heritage, major parts of it are also valid for the topic of crisis management.

The integration of databases into semantic systems has been adressed by Bizer and Seaborne [12] in their description of the D2RQ project. Hert et al. [13] also provided a comparison of several relational database to RDF mapping languages. Santipantakis et al. [14] also described the use of database to ontology mapping systems for the maritime domain. A discussion on the integration of sensors into a crisis management system can be found in [15]. The beAWARE ontology, which was used for the evaluation of our approach has been presented in [3]. The ontology integrates aspects of the domain sensors and observations, as well as metadata for geospatial information. Hence, we continue using this ontology.

III. Using geospatial knowledge

As already stated in the introduction, locations of events, places, buildings, creatures, etc., play an important role in crisis management. Locations can be expressed in different ways: Technical documentation will most likely provide coordinates of objects like sensors or buildings. In contrast, people tend to use symbolic locations, e.g. ”The fire at the cathedral in Paris” or ”the traffic accident at the crossing of St. James Street and Independance Avenue”. Sometimes, these symbolic locations are also called well-known places. Therefore, all those different types of location descriptions should be supported and there should be some automatic mapping between them. The resolving of symbolic locations is done with the help of gazetteers. A gazetteer is a geographical dictionary providing several information about the recorded well-known places, e.g. the location or geometry, the population, etc. A popular example available on the internet is GeoNames [16]. Some of those gazetteers can even be accessed via SPARQL and therefore can be integrated into a semantic crisis management system.

In some use-cases, the amount of data can be further restricted to increase the usability of the system. If some events are visualized on a map, only the events that are located inside the visual part of the map are of interest. Therefore, the semantic query used for populating the map should utilize the information about the map’s viewport. In this case, we have two geospatial restrictions for the data: process only data which is located inside the area which is affected by the crisis situation, process only data which is located inside the area visualized by the current viewport.

There are two preconditions for the use of such geospatial knowledge as part of semantic queries: First, the collected data has to be in relation with a location or geometry. This can either be explicit, e.g. the documented location of a sensor, or the geospatial information can be inferred, e.g. some textual statement mentions a well-known place. The second requirement is geospatial support of the semantic query engine. The GeoSPARQL extension from the OGC addresses this requirement. It provides SPARQL extension functions for geographic information. According to the standard document [5], GeoSPARQL provides the following features:

- An RDF/OWL vocabulary for representing spatial information consistent with the Simple Features model [17]
- A set of SPARQL extension functions for spatial computations
- A set of RIF (Rule Interchange Format [18]) rules for query transformation (not in the scope of this paper)

The vocabulary defines top-level spatial vocabulary components, as well as geometry vocabulary and topological relation vocabulary. Since the definition of relations can follow different approaches, GeoSPARQL supports three of these *relation families*: The *Simple Features* family follows the OpenGIS Simple Features specification [17], the *Egenhofer* family follows the formal definition of binary topological relationships by Egenhofer [19] and the *RCC8* family follows the Region Connection Calculus by Randell et al. [20]. The SPARQL extension functions can be divided into topological and non-topological query functions. The topological functions contain relations like *equals*, *intersects*, *touches*, *contains*, *overlaps*, etc. and are defined for each of the different relation families mentioned above. The non-topological functions contain relations like *distance*, *buffer*, *convexHull*, etc.

Geometries can have different numbers of dimensions: points (0-dimensional), lines (1-dimensional) and areas (2-dimensional). The function *equals* can be applied to all geometry types and expresses that two instances of SpatialObject are topologically equal, i.e. their interiors intersect and no part of the interior or boundary of one geometry intersects the exterior of the other. The function *intersects* can be also be applied to all geometries and states that both geometries have at least one point in common. The function *touches* can be applied to all geometries with a dimension greater than zero and expresses that both geometries have at least one boundary point in common, but no interior points. The function *overlaps* can be applied only to geometries with the same dimension. It states that they have some but not all points in common and the intersection of their interiors has the same dimension as the geometries themselves. The function *distance* returns the shortest distance in units between any two points in the two geometries as calculated in the spatial reference system of the first geometry. The function *buffer* returns a geometric object that represents all points whose distances from the given geometry are less than or equal to the given radius value measured in the given units. The calculations are made within the spatial reference system of the given geometry. Finally, the function *convexHull* returns a geometric object that represents all points in the convex hull of the given geometry.

The following enumeration lists examples of competency questions from a crisis management system, which could benefit from the integration of geospatial knowledge:

1) Which is the area with most people involved in an incident?
2) Which is the area with the highest density of incidents?
3) Will the playground be affected by the estimated flood zone?
4) Is there enough shelter capacity for all people affected by a specific incident type within a certain radius?
5) Where are the nearest sensors for a specific incident?

How can GeoSPARQL be used for answering those questions? In the following examples, the functions from the *Simple Features* relation family are used. The snippets are simplified to show only the GeoSPARQL part of the whole query to keep it short. For the first question, the locations of all incidents have to be collected and then the number of persons involved has to be aggregated for the different areas. In this case, the *contains* respectively the *sfContains* function determines which locations belong to which area as shown in the following SPARQL snippet.

```
SELECT ?area ?incidentLocation
WHERE {
?area a geo:wktLiteral;
geo:sfContains ?locLiteral.
?incidentLocation geo:asWKT ?locLiteral.
}
```

The second question is similar to the first one. If the geometries of the playground and the estimated flood zone are known, the *intersects* or *overlaps* functions will help answering question three.

```
SELECT ?playground ?estFloodZone
WHERE {
?playGround geo:hasGeometry ?playGeom.
?estFloodZone geo:hasGeometry
    ?estFloodGeom.
?playGeom geo:asWKT ?playWkt.
?estFloodGeom geo:asWKT ?estFloodWkt.
FILTER(geof:overlaps(?playWkt,
    ?estFloodWkt)
}
```

Question four is more complex. At first, the affected area has to be determined. With the help of the *buffer* function the area for potential shelters is calculated. After that, the matching shelters can be associated with those areas by using the *contains* function and their capacity can be summed up per area.

```
SELECT ?affectedZone ?incidentLocation
WHERE {
?incidentLocation geo:asWKT ?wktIncident.
BIND (geof:buffer(?wktIncident,
    uom:metre) AS ?affectedZone)
}
```

```
SELECT ?affectedZone ?shelterLocation
WHERE {
?shelterLocation geo:asWKT ?wktShelter.
?affectedZone geo:sfContains ?wktShelter.
}
```

The *distance* function will help answering the last question if the location of the sensors and the incident location is known.

```
SELECT ?sensorLocation ?incidentLocation
    ?distance
WHERE {
?sensorLocation geo:asWKT ?wktSensor.
?incidentLocation geo:asWKT ?wktIncident.
BIND (geof:distance( ?wktSensor,
    ?wktIncident, uom:metre) as ?distance)
}
ORDER BY DESC(?distance)
```

By using the functions mentioned above together with the standard SPARQL features, queries that are more complex can be built for the given questions. If some locations are provided as well-known places, they will have to be mapped to coordinates by some gazetteer services before they can be used together with the GeoSPARQL functions. If those gazetteers provide a SPARQL interface, the mapping can be done as part of a federated query, meaning that some parts of a query are sent to remote SPARQL endpoints and will be executed there.

IV. Semantic Access to Sensor Data

Changes in the environment can be observed by sensors through a large number of parameters. Since these sensors are usually connected to the internet, a large amount of information is generated, which can be used as possible input for decision support systems. Since we focus on the semantics of the available data, the information should be semantically integrated to make it accessible through semantic queries.

This integration could be done in several ways (in the following sections, the term *sensor management system* is used with the following meaning: *A system that manages sensor measurements and provides standardized access to the data measured by them.*):

- Sensors store their data directly into an ontology or use some generic adapter for this task
- An existing sensor management system is enhanced by a SPARQL interface
- The SPARQL queries are mapped to some query interface of an existing sensor management system
- The database of an existing sensor management system is accessed via some adaption layer which provides a SPARQL interface

The first option is enhancing each sensor in a way it stores its data as described by the used ontology. However, since this approach would require changes to each sensor it would counteract the idea to use existing sensors as input. Even if there were some generic adapters available for this task, another problem would still exist: While some raw observation data might not be suitable for the user of a Decision Support System, the results of some simulations or forecasts based on these observations are of greater value. Adding such simulation and processing functionality to these adapters also would make them less generic and increase their complexity. Furthermore, it could lead to massive concurrent write access to the ontology, causing a permanent index rebuild according to Pan et. al [21]. Therefore, this approach was not continued any further.

Another solution would be to integrate a SPARQL interface into an existing sensor management system. This would leave the task of collecting and storing the sensor data to an existing software, which follows a popular standard and therefore enables the use of a great number of already existing sensors. The OGC SensorThings API [6] is such a standard from the OGC, which provides a unified way to interconnect Internet of Things devices, data, and applications over the web. From the list of available implementations, the FROST-Server [22] is used for the current research. It uses a PostgreSQL relational database for the actual storage of the sensor data. Since the FROST-Server is an open-source implementation, integrating an additional SPARQL interface should be feasible. The second part of the SensorThings API standard already specifies how simulation and processing tasks can interact with an implementation of the standard.

Some small changes to this second solution leads to the third possibility: While the SPARQL interface of the previous solution would have direct access to the internals of the FROST-Server, in this approach it would be decoupled by only using the existing query interface of the server. This approach would not require changes to the implementation of the FROST-Server, but the mapping between SPARQL and the query interface would have to be implemented from scratch.

A fourth approach would be to access the underlying database of the FROST-Server via some adaption layer instead of enhancing the server itself. This would keep the server

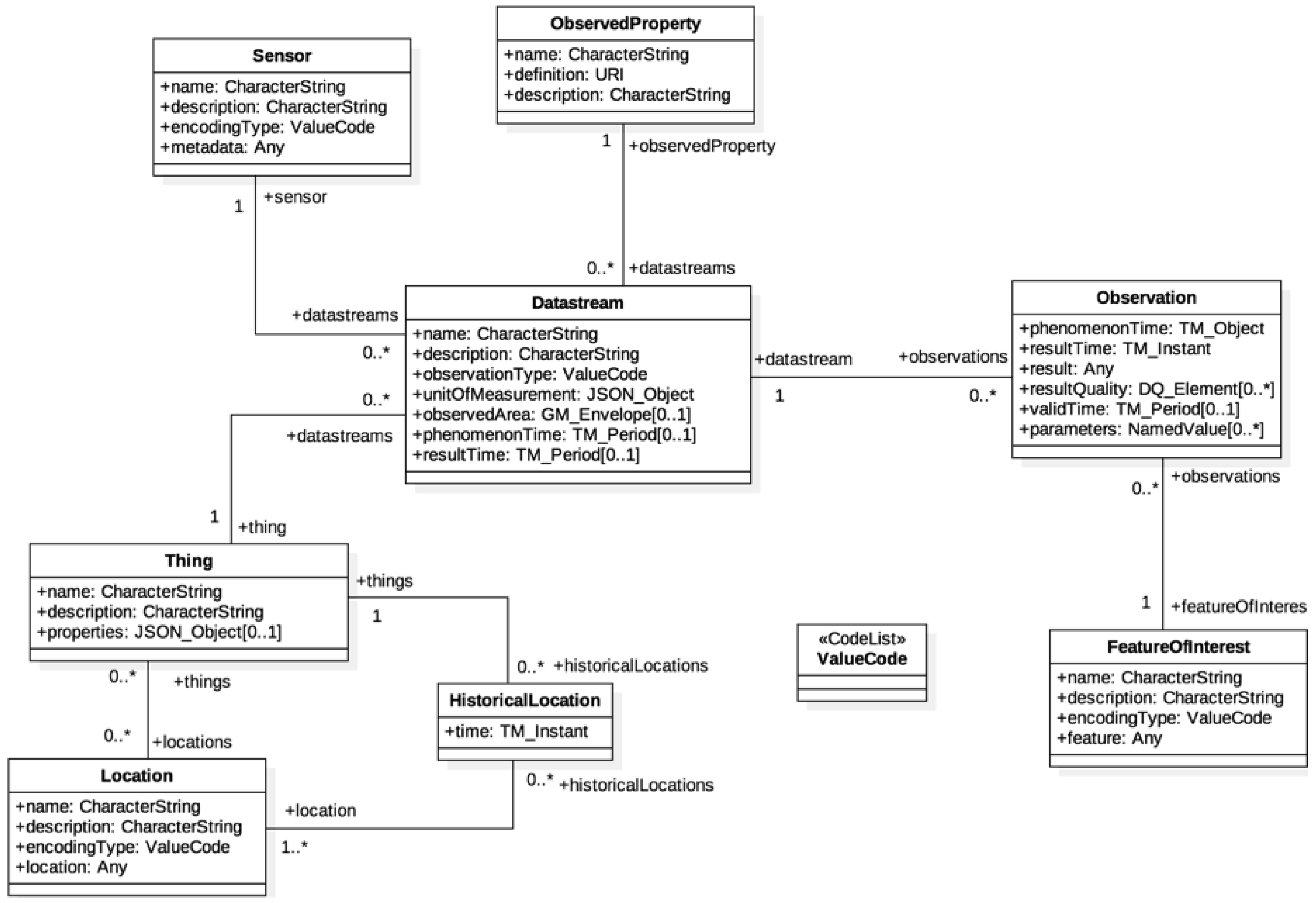

Figure 1. SensorThings Datamodel (from the OGC SensorThings API standard)

implementation simple by keeping out functionality, which is only used in some use cases. By using this approach, the original problem *Semantic access to sensor data* is transformed into the problem *Semantic access to relational databases*. For the transformed problem, there are already multiple solutions available: one of them, the D2RQ project [23] provides a generic implementation for accessing relational databases as virtual, read-only RDF graphs [12] [13]. The original project has been extended in the meantime by the community as part of the project ONT-D2RQ [7] to support OWL in addition to RDF.

Since the fourth approach promises to reach the goal with only a minimum of software development, it has been chosen and the idea of integrating a SPARQL interface directly into a sensor management system has been postponed.

The D2RQ-Server requires a mapping between the database tables and an ontology. As a first attempt, a simple ontology is created, representing the data model of the SensorThings API (see Figure 1). Parts of this ontology are integrated into the beAWARE Ontology, presented in [3]. The entities of this model are directly mapped to tables in the implementation of the FROST-Server. Therefore, table mapping means entity-mapping in our case. Each of the main entities is represented by an OWL class. The relations between entities are modeled as ObjectProperties. Finally, DatatypeProperties are used for the attributes of an entity. Since the relations of the SensorThings data model are not directed, there has to be an inverse ObjectProperty defined for each ObjectProperty. The actual mapping for ONT-D2RQ is defined in a configuration file.

Using the new modular Semantic Sensor Network Ontology (SSN, [24] would have been an alternative to the creation of an own simple ontology, but the complexity of the mapping would have been higher and some aspects of the SensorThings API which are not covered by the SSN would have required some additional extensions as well.

V. Use-Case Application

The integration of geospatial knowledge and sensor observation data with the help of ontologies (as introduced in Section 3 and 4) has been evaluated as part of the research project beAWARE: "Enhancing decision support and management services in extreme weather climate events" [8]. The goal is to develop an integrated solution to manage climate-related crises in all phases, starting with early warning before, managing during and recovery after the event.

A great number of sensors has been connected to the system for the surveillance of water levels, temperatures,

```
1  PREFIX webgenesis: <http://arqext.webgenesis.de/>
2  PREFIX beaw: <http://beaware-project.eu/beAWARE/#>
3  PREFIX sth: <http://www.iosb.fraunhofer.de/frost#>
4  PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
5  PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>
6
7  SELECT ?location ?jsonLoc  ?wktLocInc ?wktLoc ?distance
8  WHERE {
9          {SELECT ?location ?jsonLoc ?wktLocInc WHERE {
10         SERVICE <http://localhost:2020/sparql/> {
11             ?location a sth:Location;
12                         sth:LOCATIONS_LOCATION ?jsonLoc.
13         }
14                 ?wktLocInc webgenesis:convertGeoJSON2WKT ?jsonLoc.
15         }}
16     {{
17         ?incidentReportLocation beaw:latitude  _:lat;
18                                 beaw:longitude _:lon.
19         ?wktLoc webgenesis:convertLatLon2WKT (_:lat _:lon) .
20     }}
21   BIND ( geof:distance( ?wktLoc, ?wktLocInc, uom:metre) AS ?distance) .
22 }
```

Figure 2. A geoSPARQL-enhanced SPARQL query retrieving geospatial information

humidity, etc. The sensor observation data is stored inside the FROST-Server via the SensorThingsAPI interface. For the SPARQL access to these observations, the D2RServer from ONT-D2RQ has been chosen. It also supports OWL, whereas the original D2RQ implementation only supported RDF.

Since the different sources for crisis management system do not necessarily use the same representation for geospatial location data, we have implemented additional custom SPARQL extensions for converting between those representations, e.g. the *webgenesis:convertGeoJSON2WKT* property function converts the GeoJSON geometries delivered by the FROST-Server to a WKT representation required by GeoSPARQL.

Figure 2 shows a SPARQL query combining GeoSPARQL with information from within the FROST-Server. This example is constructed from several subqueries and uses the federated queries feature of SPARQL:

1) The subquery shown in lines 10 to 13 is sent to the SPARQL-endpoint of the ONT-D2RQ Server and processed there. This endpoint will then return all locations which have the DatatypeProperty sth:LOCATIONS_LOCATION set.
2) In the next step, those results are converted from GeoJSON to WKTLiterals by the use of custom property functions. This is the purpose of the subquery shown in lines 9 to 15.
3) For the incident locations, there is also a previous conversion step to have the data available as WKTLiterals. This conversation step is shown in lines 16 to 20.
4) Finally, the converted locations are used as part of a GeoSPARQL query to determine the distance between those observation locations and incident locations from the beAWARE ontology.

Similar SPARQL queries have been developed for the competency questions of the beAWARE decision support system. The query results are presented to the decision maker by means of different visualizations ranging from a simple list to geographical maps. The applicability of the approach used for the beAWARE decision support system was validated by two large-scale trials up to now an in one up-coming third trial. The evaluation report [25] of the first trial is publicly available on the project website [8].

VI. Evaluation

The feasibility of our approach could be shown, but also some problems were identified which have to be addressed before the implementation can be used in production: The performance of the mapping between the relational database and the semantic representation is not fast enough. This causes network timeouts of the SPARQL requests and degrades the usability of the system, if the user has to wait several minutes for the results of queries which were expected to be simple queries. Since the author of the SPARQL queries does not know how these queries will be mapped to SQL queries by the D2RQ implementation, the mapping may produce non-optimal SQL statements. The original D2RQ implementation already provided some optimizations to address this deficiency when handling large datasets, but it seems that not all of them are still present in the ONT-D2RQ implementation. Since both implementations provide different feature sets and the common features of both are not sufficient for our system, a direct comparison with a defined dataset was not possible.

The use of federated queries has been identified as another bottleneck: Since the variable bindings of such a query have to be transferred to the remote endpoint via the network and after the execution, the results will traverse the network again, some delay caused by the network has to be considered. In particular, the missing support of *LIMIT* and *OFFSET* restrictions for the federated queries makes the situation even more badly. Even in the cases where only a few results are needed from the remote SPARQL endpoint, that endpoint has to process the whole dataset and return a possibly large number of results, despite the fact that most of these results will then be thrown away by some *LIMIT* or *OFFSET* clause of the surrounding query.

In November 2019, the last beAWARE pilot will be conducted. Here, a quantitative benchmark will be performed. With these results, measures to improve the performance will be developed.

VII. Conclusion

In this paper, we presented the usage of GeoSPARQL for the integration of geospatial knowledge into a Decision Support System for crisis management based on semantic technologies. Competency questions that are of interest for such DSS were introduced, as well as their formal representations as geoSPARQL enhanced SPARQL queries. To access sensor data on a semantic base, the data must be semantically integrated. We introduced four possible ways how this could be achieved and how sensor observation data could be accessed from within such a system. In the Use-Case Application section, we discussed how these parts were combined within the EU-funded project beAWARE. As further task, performance optimization has been identified. In November, the last beAWARE pilot will take place in Valencia, where quantitative measures will be implemented to evaluate the systems performance. Finally, the evaluation of the approach within the project has been presented.

Acknowledgment

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 700395 and Grant Agreement No 700475.

References

[1] T. Usländer et al., "The trend towards the internet of things: what does it help in disaster and risk management?" Planet@ Risk, vol. 3, no. 1, 2015, pp. 140–145.

[2] M. van den Homberg, R. Monné, and M. R. Spruit, "Bridging the information gap: mapping data sets on information needs in the preparedness and response phase," Technologies for Development, 2018, p. 213.

[3] E. Kontopoulos, P. Mitzias, J. Moßgraber, P. Hertweck, H. van der Schaaf, D. Hilbring, F. Lombardo, D. Norbiato, M. Ferri, A. Karakostas et al., "Ontology-based representation of crisis management procedures for climate events." in ISCRAM, 2018, pp. 1064–1073.

[4] E. Kontopoulos, P. Mitzias, S. Dasiopoulou, M. J., S. Mille, P. Hertweck, T. Hellmund, A. Karakostas, S. Vrochidis, L. Wanner, and I. Kompatsiaris, "Applying semantic web technologies for decision support in climate-related crisis management," in Proceedings Citizen Observatories for natural hazards and Water management. 2nd Int. Conf. on Citizen Observatories for natural hazards and Water Management (COWM 2018). COWM, 2018.

[5] M. Perry and J. Herring, "Ogc geosparql-a geographic query language for rdf data," OGC implementation standard., 2012, last access date: 2019-09-05. [Online]. Available: https://www.opengeospatial.org/standards/geosparql

[6] S. Liang, C.-Y. Huang, T. Khalafbeigi et al., "Ogc sensorthings api-part 1: Sensing," OGC R Implementation Standard. Available online: http://docs. opengeospatial. org/is/15-078r6/15-078r6. html (accessed on 14 April 2018), 2016.

[7] A database to owl mapper. Last access date: 2019-09-05. [Online]. Available: https://github.com/avicomp/ont-d2rq

[8] beware project homepage. Last access date: 2019-09-05. [Online]. Available: https://beaware-project.eu/

[9] R. Battle and D. Kolas, "Geosparql: enabling a geospatial semantic web," Semantic Web Journal, vol. 3, no. 4, 2011, pp. 355–370.

[10] C. Zhang, T. Zhao, L. Anselin, W. Li, and K. Chen, "A map-reduce based parallel approach for improving query performance in a geospatial semantic web for disaster response," Earth Science Informatics, vol. 8, no. 3, 2015, pp. 499–509.

[11] I. Nishanbaev, E. Champion, and D. A. McMeekin, "A survey of geospatial semantic web for cultural heritage," Heritage, vol. 2, no. 2, 2019, pp. 1471–1498.

[12] C. Bizer and A. Seaborne, "D2rq-treating non-rdf databases as virtual rdf graphs," in Proceedings of the 3rd international semantic web conference (ISWC2004), vol. 2004. Proceedings of ISWC2004, 2004.

[13] M. Hert, G. Reif, and H. C. Gall, "A comparison of rdb-to-rdf mapping languages," in Proceedings of the 7th International Conference on Semantic Systems. ACM, 2011, pp. 25–32.

[14] G. Santipantakis, K. I. Kotis, and G. A. Vouros, "Ontology-based data integration for event recognition in the maritime domain," in Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics. ACM, 2015, p. 6.

[15] J. Moßgraber, D. Hilbring, H. van der Schaaf, P. Hertweck, E. Kontopoulos, P. Mitzias, I. Kompatsiaris, S. Vrochidis, and A. Karakostas, "The sensor to decision chain in crisis management." in ISCRAM, 2018, pp. 754–763.

[16] Geonames gazetteer. Last access date: 2019-09-05. [Online]. Available: https://www.geonames.org

[17] Opengis simple features specification for sql. Open GIS Consortium and others. Last access date: 2019-09-05. [Online]. Available: http://www.opengeospatial.org/standards/sfs (1999)

[18] M. Kifer and H. Boley, "Rif overview," W3C working draft, W3C, 2013, last access date: 2019-09-05. [Online]. Available: http://www.w3.org/TR/rif-overview

[19] M. J. Egenhofer, "A formal definition of binary topological relationships," in International conference on foundations of data organization and algorithms. Springer, 1989, pp. 457–472.

[20] D. A. Randell, Z. Cui, and A. G. Cohn, "A spatial logic based on regions and connection." KR, vol. 92, 1992, pp. 165–176.

[21] Z. Pan, T. Zhu, H. Liu, and H. Ning, "A survey of rdf management technologies and benchmark datasets," Journal of Ambient Intelligence and Humanized Computing, vol. 9, no. 5, 2018, pp. 1693–1704.

[22] Fraunhofer open source sensorthings api server. Last access date: 2019-09-05. [Online]. Available: https://github.com/FraunhoferIOSB/FROST-Server

[23] Accessing relational databases as virtual rdf graphs. Last access date: 2019-09-05. [Online]. Available: https://github.com/d2rq/d2rq

[24] Semantic sensor network ontology. W3C. Last access date: 2019-09-05. [Online]. Available: https://www.w3.org/TR/vocab-ssn/ (2017)

[25] F. Lombardo, D. Norbiato, M. Ferri, I. Vourvachis, M. Meliadis, A. Karakostas, I. Koulalis, T. Hellmund, and I. Koren, "D2.4 evaluation report of the 1st prototype," in beAWARE H2020 project deliverable, last access date: 2019-09-05. [Online]. Available: https://beaware-project.eu/wp-content/uploads/2019/01/D2.4_-beAWARE_Evaluation-report-of-P1_v0.5.pdf