Supporting Environmental Information Systems and Services Realization with the Geo-Spatial and Streaming Dimensions of the Semantic Web

Emanuele Della Valle[1,2] and Alessio Carenini[2]

[1] Dip. di Elettronica e Informazione, Politecnico di Milano, Milano, Italy
[2] CEFRIEL, Politecnico di Milano, Milano, Italy
email: `emanuele.dellavalle@polimi.it`

Abstract. Environmental Information Systems and Services require flexible discovery and chaining of distributed environmental services to support a large number of concurrent decision processes. The ability to **cope with geo-spatial features** of the environment and to **process in real time huge and possibly noisy data streams** are two critical factors in supporting such decision processes. Solution to separately cope with the two aspects are available. The geo-spatial aspect has been studied for decades in the Geographic Information System (GIS) community. Data Stream Management Systems (DSMS) are the result of a decade of investigation on data stream processing by the database community. However, seamless integrated usage of GIS and DSMS is still a difficult task. Recent developments of the Semantic Web community have been trying to overcome the barriers between these two technologies by proposing to extend the Semantic Web with both a Geo-Spatial and a Streaming dimension. In this paper, these two dimensions of Semantic Web are show-cased for environmental monitoring and management in oil and gas operations.

1 Introduction

Remaining world leader in the oil and gas industry while achieving continuous improvements in environmental performance is one of the objectives for the years 2009-2011 listed by Norwegian Oil Industry Association (OLF)[3][1]. In OLF vision, a number of areas are identified where ICT technology can be used to create smarter solutions. OLF vision calls for:

– better **ICT infrastructure** able to increase communication capability from sensors and controllers to the platform and onshore control rooms;
– better **data integration** solutions able to break the vendor specific silos that make it hard, if at all possible, to correlate data produced by different vendor's equipment; and
– more **intelligent systems** able to interpret the huge amount of real-time sensor data about production, environment and facilities against the even

[3] http://www.olf.no/

larger amount of information that describe wells, templates, processing plants, and pipelines.

For instance, oil operation engineers base their decision processes on real time data acquired from sensors on oil rigs, both on the sea surface and on the seabed. A typical oil production platform is equipped with about 400.000 sensors for measuring environmental and technical parameters. Some of the questions they faces are:

– Given an alarm on a well in progress to drown, how much time do I have given the historical behavior of that well?
– Given this brand of turbine, what is the expected time to failure when the barring starts to vibrate as now detected?
– How do I detect weather events from observation data?
– Which sensors have observed a blizzard within a 100 mile radius of a given location.

Answering these questions requires to process an (almost) "continuous" flow of information – with the recent information being more relevant as it describes the current state of a dynamic system – against a rich background knowledge – with geospatial information playing a central role.

The Semantic Web can provide to the oil industry, and in general the Environmental Information Systems and Services research area, the standard technologies for data integration, but state-of-the-art semantic technologies can only partially support the need for intelligent systems in the oil industry.

In the rest of the paper, we briefly discuss (see Section 2 and 3) recent attempts to add to the Semantic Web the ability to continuously process data flows and to efficiently perform geospatial analysis. We exemplifying their usage for analyzing weather sensor data places all around the oil fields. In particular, in Section 4, we describe how we are developing a solution for chaining different processing units within the LarKC project[4]. Finally, in Section 5 we draw some conclusions.

2 Continuous Processing of Data Streams

Continuous processing of flows of information (namely **data streams**) has been largely investigated in the database community [2]. Specialized Data Stream Management Systems (DSMS) are available on the market and features of DSMS are appearing also in major database products, such as Oracle and DB2.

On the contrary, continuous processing of data streams *together with rich background knowledge* requires specialized reasoners, but work on semantic technologies is still focusing on rather static data. In existing work on logical reasoning, the knowledge base is always assumed to be static (or slowly evolving). There is work on changing beliefs on the basis of new observations [3], but the

[4] http://www.larkc.eu

solutions proposed in this area are far too complex to be applicable to gigantic data streams of the kind illustrated in the oil production example above.

As argued in [4], we strongly believe that there is a need to close this gap between existing solutions for belief update and the actual needs of supporting decision process based on data streams and rich background knowledge. We named this little explored, yet high-impact research area **Stream Reasoning**.

The foundation for complex reasoning over streams and background knowledge has been investigated since 2008 by introducing technologies for wrapping and querying streams in the RDF data format and by supporting simple forms of reasoning. In this paper, we focus on Continuous-SPARQL (shortly C-SPARQL) [5–8].

Listing 1.1 shows an example of C-SPARQL query that detects a blizzard: a severe storm condition lasting for 3 hours or more characterized by low temperatures, strong winds, and heavy snow.

```
1  PREFIX so: <http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#>
2  PREFIX w: <http://knoesis.wright.edu/ssw/ont/weather.owl#>
3  
4  REGISTER STREAM BlizzardDetection COMPUTE EVERY 10m AS
5  CONSTRUCT {
6    ?sensor so:generatedObservation [a w:blizzard] ;
7            so:samplingTime fn:now() .
8  }
9  FROM <http://oilprod.org/weatherStations.rdf>
10 FROM STREAM <http://oilprod.org/weatherObs.trdf>
11             [RANGE 3h STEP 10m]
12 WHERE {
13  ?sensor so:generatedObservation [a w:SnowfallObservation] .
14  { SELECT ?sensor
15    WHERE { ?sensor so:generatedObservation ?o1
16            ?o1 a w:TemperatureObservation ;
17              so:observedProperty w:AirTemperature ;
18              so:result [ so:value ?temperature  ] . }
19    GROUP BY ( ?sensor )
20    HAVING (AVG(?temperature)<"0.0"^^xsd:float) }
21  { SELECT ?sensor
22    WHERE { ?sensor so:generatedObservation ?o2
23            ?o2 a w:WindObservation ;
24              so:observedProperty w:WindSpeed ;
25              so:result [ so:value ?speed  ] . }
26    GROUP BY ( ?sensor )
27    HAVING (MIN(?speed)> "40.0"^^xsd:float)  }
28 }
```

Listing 1.1. Example of C-SPARQL which detects a blizzard

At line 4, the `REGISTER` clause is use to tell the C-SPARQL engine that it should register a **continuous query**, i.e. a query that will continuously compute answers to the query. In particular, we are registering a query that generates as output an RDF stream (i.e., we use `REGISTER STREAM`). The `COMPUTE EVERY` clause states the frequency of every new computation, in the example every 10 minutes. At line 9, the standard SPARQL clause `FROM` is used to load in the default graph the location of all weather stations. At line 10, the C-SPARQL specific clause `FROM STREAM` defines the RDF stream of weather observations. We supposed that the observations are encoded in RDF using the Semantic Sensor Web ontology [9]. Next, line 11 defines the **window** of observation over the RDF stream of weather observations. Streams, for their very nature, are volatile and for this reason should be consumed on the fly; thus, they are observed through a window, including the last elements of the stream, which changes over time. In the example, *the window comprises weather observations produced in the last 3 hour, and the window slides every 10 minutes.* The `WHERE` clause conforms to the under-development SPARQL 1.1 standard [10]. It uses sub-queries and aggregates as defined in [11]. The sub-query from line 14 to 20 checks that the average temperature has been below 0, while the one from line 21 to 27 checks that the minimum wind speed has been above 40 km/h. Finally, selected stations are used to construct the elements of the RDF stream specified in the `CONSTRUCT` clause between line 5 and 8. The xPath function `now()` is used to describe when the blizzard was detected.

As Listing 1.1 illustrates, C-SPARQL enables the encoding of the typical questions an oil operation engineer has to answer. This is possible, because C-SPARQL extends SPARQL with the notions of *window* and of *continuous processing.*

Two approaches alternative to C-SPARQL exist: Streaming SPARQL [12] and Time-Annotated SPARQL (or simply TA-SPARQL) [13]. Both languages introduce the concept of window over stream, but only C-SPARQL brings the notion of continuous processing, typical of stream processing, into the language; all the other proposal still rely on permanent storing the stream before processing it using one-shot queries. Moreover, only C-SPARQL exploits optimization techniques [7] that push, whenever possible, aggregates computation as close as possible to the raw data streams; and only C-SPARQL efficiently supports OWL2-RL entailment regime [8].

3 Efficient Geospatial Analysis

Efficient geospatial analysis methods have been developed over the past half century and most of them are available in Geographic Information Systems (GIS) packages. However, the Semantic Web community has devoted very limited attention to the spatial dimension of data. Available solutions (e.g., Virtuoso [14] or AllegroGraph [15]) offer a limited support if compared to the rich features normally available in a GIS.

The main reason for such a limited support is the tendency to re-implement geospatial analysis algorithm (e.g., R-tree indexes) in RDF repositories. This is neither necessary, nor efficient. Since 2003, several solutions have been conceived and implemented [16–20] addressing the growing need for RDF applications to access the content of non-RDF, legacy databases without having to replicate the whole database into RDF. They provide (in slightly different ways) declarative languages to describe mappings between relational database schemata and RDF-S vocabularies (or more expressive ontological languages). Once the mapping is ready, they can use it to rewrite SPARQL query in SQL. We refer interested readers to [17] and [18] for a comprehensive explanation of the mapping languages of D2RQ and Virtuoso and of the query rewriting algorithms.

In [21] an extension of D2RQ, namely GIS2RDF (G2R), is proposed to treat GIS as virtual RDF graphs by rewriting SPARQL query to GIS query (specifically SQL/MM spatial standard [22]).

Listing 1.2 shows an example of SPARQL query that detects the platforms within oil-fields in which more than 10 blizzards were detected in the last month.

```
SELECT ?oilField ?platform
FROM
WHERE {
   ?oilField ex:hasSurface ?oilFieldSurface .
   ?platform ex:hasSurface ?platformSurface .
   ?sensor grs:point ?sensorPosition ;
           so:generatedObservation [a w:blizzard] ;
           so:samplingTime ?time .
   FILTER (g2r:contains(?oilFieldSurface, ?sensorPosition)
           && g2r:overlaps(?oilFieldSurface, ?platformSurface))
   FILTER(?time >= "2010-10-01T00:00:00Z^^xsd:dateTime")
   FILTER(?time <= "2010-09-01T00:00:00Z^^xsd:dateTime")
} GROUP BY ?oilFieldSurface
HAVING (COUNT(?sensor) > 10)
```

Listing 1.2. Example of SPARQL query requiring geospatial analysis that G2R can efficiently answer using an underlying GIS.

The query in Listing 1.2 is a standard SPARQL 1.1 query that uses two of the extended value testing functions available in G2R: `g2r:contains` and `g2r:overlaps`. `g2r:contains` checks whether the sensor is contained in the area (in the general case a curved polygon) of the oil-field. `g2r:overlaps` tests if the area of the oil platform overlaps the area of the oil-field (in the general case both are a curved polygon).

G2R rewrites the query in Listing 1.2 in the equivalent SQL MM/Spatial query in Listing 1.3 using mappings of the kind declared in Listing 1.4.

```
SELECT o.ID, p.ID,
FROM platform AS p, oilFields AS o, sensors AS s
WHERE  s.generatedObservation = "blizzard" AND
       p.area.ST\_Within(s.position) = 1 AND
       b.area.ST\_Overlaps(o.area) = 1 AND
       s.samplingTime >= "2010-09-01T00:00:00Z" AND
       s.samplingTime <= "2010-10-01T00:00:00Z"
GROUP BY o.ID
HAVING COUNT (s.generatedObservation) > 10
```

Listing 1.3. A SQL MM/Spatial query equivalent to the SPARQL query in Listing 1.2 generated by g2r

In Listing 1.4, note that the extended value testing functions available in G2R, i.e., `g2r:contains` and `g2r:overlaps`, are rewritten in the respective SQL MM/Spatial functions `ST_Within()` and `ST_Overlaps()`.

```
map:area a g2r:SpatialPropertyBridge;
         d2rq:belongsToClassMap map:platform;
         d2rq:property ex:hasSurface;
         g2r:spatialColumn "area";
         d2rq:datatype g2r:Polygon .
```

Listing 1.4. A SQL MM/Spatial query equivalent to the SPARQL query in Listing 1.3 generated by g2r

Moreover, note that the mapping declared in Listing 1.4 allows G2R to map the property `ex:hasSurface` to the spatial column "area" of the GIS, which is a polygon.

4 Combining the Two Approaches with LarKC

C-SPARQL and G2R have not been integrated yet, but we are working on it in the LarKC project. The main goal of LarKC [23] is to develop a pluggable platform for reasoning on massive heterogeneous information integrating techniques from various areas including databases, machine learning, Semantic Web and Geographic Information Systems. LarKC facilitates the processing of a complex SPARQL query by orchestrating various plug-ins that are able to provide partial answer to the query. In the case described in this paper, such plug-ins will be a C-SPARQL engine and G2R, while data integration support can be provided by LarKC datalayer[5].

Once the integration will be complete, we will be able to issue a C-SPARQL query that every 30 minutes determines the geographical area interest by a blizzard by combining the positions of the sensors that have been detecting a blizzard in the last 3 hours (i.e., the RDF stream resulting from the C-SPARQL query in Listing 1.1). To achieve this results (see Listing 1.5) the spatial function `g2r:convexHull` is called to compute the minimal convex polygon that contains all the sensor positions.

```
PREFIX so: <http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#>
PREFIX w: <http://knoesis.wright.edu/ssw/ont/weather.owl#>

REGISTER STREAM BlizzardAreaDetection COMPUTE EVERY 30m AS
CONSTRUCT {
  [] a w:blizzard ;
     ex:hasArea g2r:convexHull(?sensorPoint} .
}
FROM <http://oilprod.org/weatherStations.rdf>
FROM STREAM <http://oilprod.org/BlizzardDetection.trdf> [RANGE 3h STEP 30m]
WHERE {
 ?sensor so:generatedObservation [a w:blizzard] ;
         grs:point ?sensorPosition .
}
```

Listing 1.5. Example of C-SPARQL that requires G2R to be efficiently evaluated

By further processing the results of this query, we can detect which oil platform could be potentially interested in the near future by a blizzard with the C-SPARQL query illustrated in Listing 1.6). This query uses the spatial function `g2r:buffer` to generate a buffer of 20 km around the convex hull previously computed and returns the oil platforms that are placed within this area.

[5] LarKC datalayer is the powerful middleware behind http://factforge.net/, the first successful attempt to assembly independent datasets published on the Semantic Web in a single consistent knowledge base.

```
PREFIX so: <http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#>
PREFIX w: <http://knoesis.wright.edu/ssw/ont/weather.owl#>

REGISTER QUERY PlatformToAlertForPotentialBlizzard COMPUTE EVERY 30m AS
SELECT ?platform
FROM <http://oilprod.org/weatherStations.rdf>
FROM STREAM <http://oilprod.org/BlizzardAreaDetection.trdf> [RANGE 3h STEP
    30m]
WHERE {
 ?blizzard a w:blizzard ;
           ex:hasArea ?blizzardArea .
 ?platform ex:hasSurface ?platformSurface .
 FILTER (g2r:overlaps(g2r:buffer(?blizzardArea, "20"^^g2r:km), ?
     platformSurface))
}
```

Listing 1.6. Another example of C-SPARQL that requires G2R to be efficiently evaluated

5 Conclusion

In this paper we have illustrated the potential usage of C-SPARQL and G2R in the context of oil production. We have shown how to extend the Semantic Web standards, which already facilitate data integration, with the ability to cope with the geospatial features of the environment and to process in real time huge and possibly noisy sensor data streams. We have also reported on the ongoing work, within the LarKC project, in providing a pluggable platform for chaining various systems such as our C-SPARQL Engine and G2R Engine on top of a powerful data integration layer.

The path that leads to systems able to support in real-time the decision making processes of hundreds of concurrent users (e.g., the controllers on the platform and in the onshore control rooms) is still long. However, we trust that the partially implemented infrastructure, described in this paper, is a concrete step in the direction of developing flexible Environmental Information Systems and Services.

Acknowledgements

The work described in this paper has been partially supported by the European project LarKC (FP7-215535). We also thank Titi Roman, Arne Je Berre and Einar Landre for the fruitful discussions that are at the basis of this paper.

References

1. Norwegian Oil Industry Association: Objectives and Goals for the Years 2009-2011. http://www.olf.no/objectives-and-goals/objectives-and-goals-2009-2011-article2911-293.html

2. Garofalakis, M., Gehrke, J., Rastogi, R.: Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
3. Gaerdenfors, P., ed.: Belief Revision. Cambridge University Press (2003)
4. Della Valle, E., Ceri, S., van Harmelen, F., Fensel, D.: It's a Streaming World! Reasoning upon Rapidly Changing Information. IEEE Intelligent Systems **24**(6) (2009) 83–89
5. Della Valle, E., Ceri, S., Barbieri, D.F., Braga, D., Campi, A.: A First Step Towards Stream Reasoning. In: Proc. Future Internet Symposium (FIS). (2008) 72–81
6. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: C-sparql: Sparql for continuous querying. In: WWW. (2009) 1061–1062
7. Barbieri, D.F., Braga, D., Ceri, S., Grossniklaus, M.: An Execution Environment for C-SPARQL Queries. In: Proc. Intl. Conf. on Extending Database Technology (EDBT). (2010)
8. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In Aroyo, L., Antoniou, G., Hyvonen, E., eds.: ESWC. Lecture Notes in Computer Science, Springer (2010)
9. Knoesis Lab: Semantic Sensor Web ontology. http://knoesis.wright.edu/research/semsci/application_domain/sem_sensor/ont/sensor-observation.owl
10. Kjernsmo, K., Passant, A.: SPARQL New Features and Rationale. http://www.w3.org/TR/2009/WD-sparql-features-20090702/
11. Harris, S., Seaborne, A.: SPARQL 1.1 Query. W3C Working Draft 22 October 2009. http://www.w3.org/TR/2009/WD-sparql11-query-20091022 (October 2009)
12. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL – Extending SPARQL to Process Data Streams. In: Proc. Europ. Semantic Web Conf. (ESWC). (2008) 448–462
13. Rodriguez, A., McGrath, R., Liu, Y., Myers, J.: Semantic Management of Streaming Data. In: Proc. Intl. Workshop on Semantic Sensor Networks (SSN). (2009)
14. Erling, O.: RDF Geography With Virtuoso. http://www.openlinksw.com/weblog/oerling/index.vspx?page=&id=1587
15. Franz, Incorporated: Geospatial support in SPARQL queries. http://www.franz.com/agraph/support/documentation/v4/sparql-geo.html
16. Bizer, C., Seaborne, A.: D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs. In: ISWC2004 (posters). (November 2004)
17. Bizer, C., Cyganiak, R., Garbers, J., Maresch, O., Becker, C.: The D2RQ Platform v0.7 - Treating Non-RDF Relational Databases as Virtual RDF Graphs. User Manual and Language Specification. http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/ (August 2009)
18. Erling, O., Mikhailov, I.: Mapping Relational Data to RDF in Virtuoso Virtuoso. http://virtuoso.openlinksw.com/wiki/main/Main/VOSSQLRDF
19. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: light-weight linked data publication from relational databases. In: WWW. (2009) 621–630
20. Team, J.: SquirrelRDF. http://jena.sourceforge.net/SquirrelRDF/
21. Della Valle, E., Qasim, H.M., Celino, I.: Towards treating GIS as virtual RDF graphs. In: WebMGS. (2010)
22. Stolze, K.: SQL/MM Spatial - The Standard to Manage Spatial Data in a Relational Database System. GI **26** (2003) 247–264

23. Fensel, D., van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Della Valle, E., Fischer, F., Huang, Z., Kiryakov, A., il Lee, T.K., Schooler, L., Tresp, V., Wesner, S., Witbrock, M., Zhong, N.: Towards larkc: A platform for web-scale reasoning. In: ICSC, IEEE Computer Society (2008) 524–529