A Visual Query System for Stream Data Access over Ontologies

Ahmet Soylu[1(✉)], Martin Giese[2], Rudolf Schlatte[2], Ernesto Jimenez-Ruiz[3], Özgür Özçep[4], and Sebastian Brandt[5]

[1] Norwegian University of Science and Technology, Gjøvik, Norway
ahmet.soylu@ntnu.no
[2] University of Oslo, Oslo, Norway
{martingi,rudi}@ifi.uio.no
[3] University of Oxford, Oxford, UK
ernesto.jimenez-ruiz@cs.ox.ac.uk
[4] University of Lübeck, Lübeck, Germany
oezcep@ifis.uni-luebeck.de
[5] Siemens AG, Munich, Germany
sebastian-philipp.brandt@siemens.com

Abstract. In this demo, we present an ontology-based visual query system, namely OptiqueVQS, extended for a stream query language called STARQL in the context of use cases provided by Siemens AG.

Keywords: Visual query formulation · Ontology · Streams · Sensors · OBDA

1 Motivation

Siemens runs several service centers for power plants, each responsible for remote monitoring and diagnostics of many thousands of gas/steam turbines and associated components such as generators and compressors. Diagnosis engineers working at the service centers are informed about any potential problem detected on site. They access a variety of raw and processed data with pre-defined queries in order to isolate the problem and to plan appropriate maintenance activities. For diagnosis situations not initially anticipated, new queries are required, and an IT expert familiar with both the power plant system and the data sources in question (e.g., up to 2.000 sensors in a part of appliance and static data sources) has to be involved to formulate these queries. Thus, unforeseen situations may lead to significant delays of up to several hours or even days.

By enabling diagnosis engineers to formulate complex queries on their own with respect to an expressive domain vocabulary, IT experts will not be required anymore for adding new queries, and manual preprocessing steps can be avoided. Yet they rarely have technical skills and knowledge on formal languages for querying streams. Ontology-based visual query formulation is promising as ontologies provide higher level abstractions closer to end users' understanding

H. Sack et al. (Eds.): ESWC 2016 Satellite Events, LNCS 9989, pp. 161–166, 2016.
DOI: 10.1007/978-3-319-47602-5_32

(cf. [4,9]). Moreover, ontology-based data access (OBDA) technologies extend the reach of ontology-based querying from triple stores to relational databases (cf. [12]).

C-SPARQL [1], SPARQLstream [2], CQELS [5], and STARQL [6] are notable examples of semantic stream query languages. Although several visual tools exist for SPARQL (cf. [9]), the work is very limited for semantic stream query languages. An example is SPARQL/CQELS visual editor designed for Super Stream Collider framework [7]. However, it follows the jargon and structure of the underlying formal language closely and is not appropriate for end users.

In this demo, we present an ontology-based visual query system, namely OptiqueVQS [8], extended for a stream query language, called STARQL [6].

2 System Overview

OptiqueVQS is developed as a part of an OBDA system, Optique [4], which employs data virtualisation to enable in-place querying of legacy relational data sources over ontologies. This is realised through a set of mappings describing the relationships between the terms in the ontology and their representations in the data sources, and a query rewriting mechanisms for translating SPARQL to target language (e.g., SQL) [3,6]. In this demo, the focus is on visual query formulation and the underlying OBDA framework is out of scope. The following overviews the STARQL query language and OptiqueVQS with stream querying.

2.1 STARQL

STARQL [6] provides an expressive declarative interface to both historical and streaming data. In STARQL, querying historical and streaming data proceeds in an analogous way and in both cases the query may refer to static data. The relevant slices of the temporal data are specified with a window. In the case of historical data, this is a window with fixed endpoints. In the case of streaming data, it is a moving window and it contains a reference to the developing time NOW and a sliding parameter that determines the rate at which snapshots of the data are taken. The contents of the temporal data are grouped according to a sequencing strategy into a sequence of small graphs that represent different states. On top of the sequence, relevant patterns and aggregations are formulated in the HAVING-clause, using a highly expressive template language. In Fig. 1, an example STARQL query is given, which asks for a train with turbine named "Bearing Assembly", and queries for the journal bearing temperature readings in the generator. It uses a simple "echo" to display the results.

2.2 OptiqueVQS

OptiqueVQS[1] [8] is widget-based and supports three-shaped conjunctive queries. In Fig. 2, an example query is shown as a tree in the upper widget (W1), representing typed variables as nodes and object properties as arcs. Typed variables

[1] Demo video: https://youtu.be/TZTxujz5hCc.

```
PREFIX ns1 : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ns2 : <http://www.siemens.com/ontology/gasturbine/>
CREATE PULSE WITH FREQUENCY = "PT1s"^^xsd:duration

CREATE STREAM S_out AS
SELECT { ?_val0 ?Train_c1 ?Turbine_c2 ?Generator_c3 ?BearingHouse3_c4
  ?JournalBearing_c5 ?TemperatureSensor_c6 }
FROM STREAM measurement
  [NOW - "PT10s"^^xsd:duration, NOW]->"PT1s"^^xsd:duration
WHERE {
  ?Train_c1 ns1:type ns2:Train.
  ?Turbine_c2 ns1:type ns2:Turbine.
  ?Generator_c3 ns1:type ns2:Generator.
  ?BearingHouse3_c4 ns1:type ns2:BearingHouse3.
  ?JournalBearing_c5 ns1:type ns2:JournalBearing.
  ?TemperatureSensor_c6 ns1:type ns2:TemperatureSensor.
  ?Train_c1 ns2:hasTurbine ?Turbine_c2.
  ?Train_c1 ns2:hasGenerator ?Generator_c3.
  ?Generator_c3 ns2:hasBearingHouse3 ?BearingHouse3_c4.
  ?BearingHouse3_c4 ns2:hasJournalBearing ?JournalBearing_c5.
  ?JournalBearing_c5 ns2:isMonitoredBy ?TemperatureSensor_c6.
  ?Turbine_c2 ns2:hasName "Bearing Assembly"^^xsd:string.
}
SEQUENCE BY StdSeq AS seq
HAVING  EXISTS i IN seq
  ( GRAPH i { ?TemperatureSensor_c6 ns2:hasValue ?_val0 } )
```

Fig. 1. An example diagnostic task in STARQL.

can be added to the query by using the list in the bottom-left widget (W2). If a query node is selected, the faceted widget (W3) at the bottom-right shows controls for refining the corresponding typed variable, e.g. constraining a data property. Once a restriction is set on a data property or a data property is selected for output (i.e., using the eye icon), it is reflected in the label of the corresponding node in the query graph. The user can always jump to a specific part of the query by clicking on the corresponding variable-node in W1.

In W3, dynamic properties (i.e., whose extensions are time dependent) are colored in blue and as soon as one is selected OptiqueVQS switches to STARQL mode. A stream button appears on top of the W1 and lets the user configure parameters such as slide (i.e., frequency at which the window content is updated/moves forward) and window width interval. If the user clicks on the "Result Overview" button, a template selection widget (W4) appears for selecting a template for each stream attribute, which is by default "echo" (see Fig. 3). W4 is normally used for displaying example results in the SPARQL mode. The example query depicted in Fig. 2 and Fig. 3 represents the query example given in Fig. 1 with the exception that a "range" template is selected. The user can register the query in W4 by clicking on the "Register query" button.

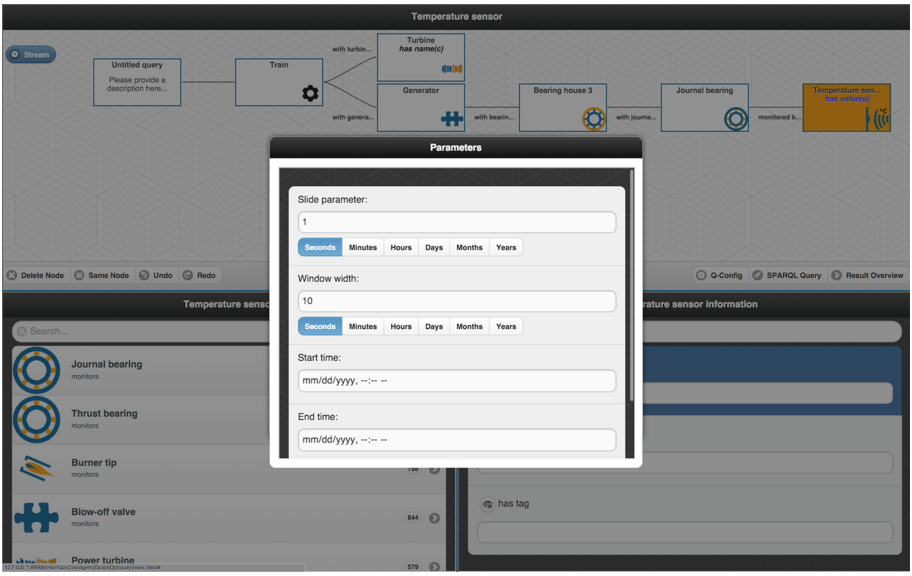

Fig. 2. OptiqueVQS with stream querying.

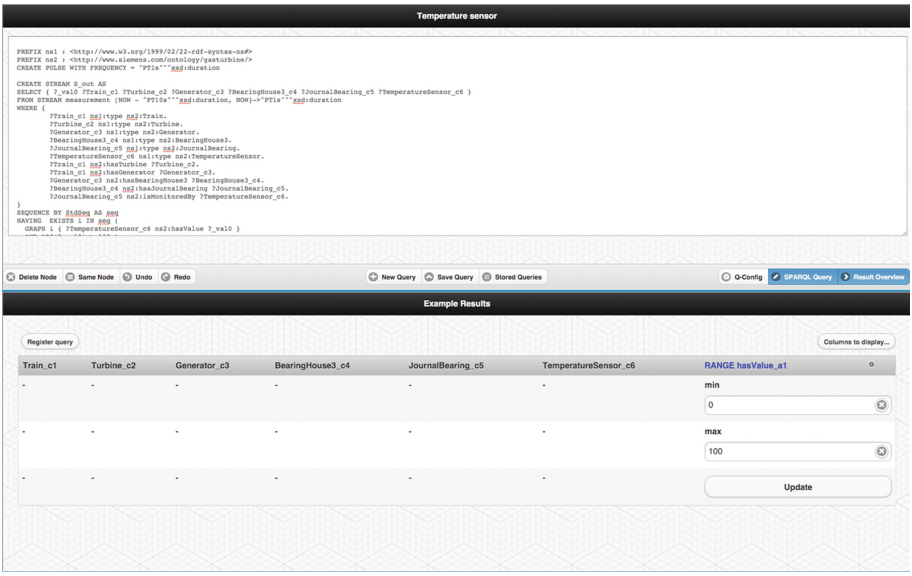

Fig. 3. OptiqueVQS with stream querying – template selection.

Several user experiments have been conducted over OptiqueVQS and the results suggest that OptiqueVQS is viable tool for end-user querying (e.g., [8, 10,11]).

3 Demonstration Scenario

We will demonstrate OptiqueVQS over anonymised Siemens' relational stream data gathered from numerous gas and steam turbines and a set of representative diagnostic tasks. OptiqueVQS will run over the Optique platform and attendees will be able to formulate, register, and execute stream queries.

Acknowledgments. This research is funded by the "Optique" (FP7 - 318338).

References

1. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: SPARQL for continuous querying. In: Proceedings of the 18th International Conference on World Wide Web (WWW 2009), pp. 1061–1062. ACM (2009)
2. Calbimonte, J.-P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 96–111. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17746-0_7
3. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: answering SPARQL queries over relational databases. Semantic Web (in press)
4. Giese, M., Soylu, A., Vega-Gorgojo, G., Waaler, A., Haase, P., Jimenez-Ruiz, E., Lanti, D., Rezk, M., Xiao, G., Ozcep, O., Rosati, R.: Optique-zooming in on big data access. IEEE Comput. **48**(3), 60–67 (2015)
5. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25073-6_24
6. Özçep, Ö.L., Möller, R., Neuenstadt, C.: A stream-temporal query language for ontology based data access. In: Lutz, C., Thielscher, M. (eds.) KI 2014. LNCS, vol. 8736, pp. 183–194. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11206-0_18
7. Quoc, H.N.M., Serrano, M., Phuoc, D.L., Hauswirth, M.: Super stream collider: linked stream mashups for everyone. In: Proceedings of the Semantic Web Challenge at ISWC 2012 (2012)
8. Soylu, A., Giese, M., Jimenez-Ruiz, E., Vega-Gorgojo, G., Horrocks, I.: Experiencing OptiqueVQS - a multi-paradigm and ontology-based visual query system for end-users. Univers. Access Inf. Soc. **15**(1), 129–152 (2016)
9. Soylu, A., Giese, M., Kharlamov, E., Jimenez-Ruiz, E., Zheleznyakov, D., Horrocks, I.: Ontology-based end-user visual query formulation: why, what, who, how, and which? Univers. Access Inf. Soc (2016, in press). doi:10.1007/s10209-016-0465-0
10. Soylu, A., Giese, M., Schlatte, R., Jimenez-Ruiz, E., Ozcep, O., Brandt, S.: Domain experts surfing on stream sensor data over ontologies. In: Proceedings of the 1st International Workshop on Semantic Web Technologies for Mobile and Pervasive Environments (2016)

11. Soylu, A., Kharlamov, E., Zheleznyakov, D., Jimenez-Ruiz, E., Giese, M., Horrocks, I.: Ontology-based visual query formulation: an industry experience. In: Bebis, G., et al. (eds.) ISVC 2015. LNCS, vol. 9474, pp. 842–854. Springer, Heidelberg (2015). doi:10.1007/978-3-319-27857-5_75
12. Spanos, D.E., Stavrou, P., Mitrou, N.: Bringing relational databases into the semantic web: a survey. Semant. Web **3**(2), 169–209 (2012)