Semantic Data Integration for the SMT Manufacturing Process using SANSA Stack

Mohamed Nadjib Mami ✉ [1], Irlán Grangel-González [2], Damien Graux [1,3], Enkeleda Elezi[1], and Felix Lösch[2]

[1] IAIS, Fraunhofer Gesellschaft, Germany
[2] Bosch Corporate Research, Renningen, Germany
[3] ADAPT Centre, Trinity College Dublin, Ireland
{mami,s6enelez}@cs.uni-bonn.de, damien.graux@adaptcentre.ie,
{irlan.grangelgonzalez|felix.loesch}@de.bosch.com

Abstract. In this article, we report on our successful integration of Semantic Web techniques in a large Industry 4.0 context. We deploy the SANSA Stack to enable the uniform access to Surface-Mount Technology (SMT) data. An ergonomic visual user interface is proposed to help non-technical users coping with the various concepts underlying the process and conveniently interacting with the data.

1 Introduction

Semantic Data Integration is one of the prominent applications of Semantic technologies whose value has been showcased in a lot of use cases in Industry 4.0 [2,4]. Schemata of the data are mapped to high-level ontologies using so-called mapping languages. This allows to build a general schema against which queries can be posed uniformly using SPARQL query language [3]. Using this schema, semantic conflicts are resolved and data can be queried on-the-fly without requiring the conversion of the full data to RDF: a data model representing the world in triples (subject, property, objects) *e.g.*, (*:machine153, msmt:smdMachineName, "ABD3"*), where *msmt* is the ontology in which *smdMachineName* is defined. This project is an application of these techniques in an industrial application for accessing Surface Mount Technology (SMT) data.

Technically, SMT is a process for mounting electronic components, *e.g.*, microchips, resistors, or capacitors on printed-circuit boards. Several sub-processes are involved in producing these boards and are executed by specialized machines. In particular, we focus this effort on the Surface Mount Devices (SMD) and the Automatic Optical Inspection (AOI). The SMD places the electronic components on top of the printed-circuit boards, and the AOI inspects the boards for any error that could have occurred, *e.g.*, misplaced or bad solder components (see [9] for an example of error detection method). In SMT, both SMD and AOI are subprocesses generating large amounts of data. These data comprise semantic interoperability conflicts, *e.g.*, same objects in real life that are named differently in SMD and AOI. To properly explore this data, these conflicts demand to be

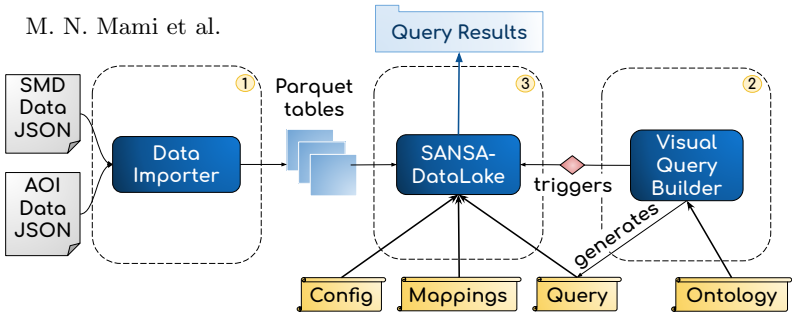

Fig. 1. General Architecture of the proposed solution.

resolved. To that end, we propose a solution that is composed of several components that extract SMT data, transfer it to an efficient format, and query it in a scalable manner. The solution also includes a user-friendly graphical interface that supports non-experts to construct queries and trigger their execution.

2 A SANSA-based solution to access SMT data

To enable and facilitate exploring the SMT data (SMD and AOI in particular), we propose a solution that consists of three main components (cf. Figure 1).

2.1 Data Importer

Data generated by the SMT process is stored in JSON format. Since JSON format is not optimal for large-scale processing, we used a more efficient format called Parquet[4]. Parquet is a columnar file format that is well-suited for analytical queries, a prevalent class of queries in industrial applications. The Data Importer is created to convert SMD data to Parquet tables. Being columnar by nature, data in Parquet is stored by columns instead of rows. This means that if a query requests three columns, only those columns are accessed instead of the entire row. This also makes Parquet compression-friendly, since data of the same column are homogeneous, in contrast to data of the same row. Data Importer is built to create Parquet tables starting from parsing JSON files. Several tables are created, capturing any encountered foreign-primary key relations.

2.2 SANSA DataLake

SANSA-DL [6,7] is an extensible software solution that allows to query heterogeneous and large data sources using SPARQL. It is part of the SANSA Stack[5] [5], a distributed framework for large scale RDF querying, inference and machine learning. SANSA-DL decomposes SPARQL queries into sub-queries, each one extracting an entity from the data. Relevant entities are detected based on a

[4] `https://parquet.apache.org/`
[5] Scalable Semantic Analytics Stack `http://sansa-stack.net/`

```
<#MachineMap>
rml:logicalSource [
 rml:source: "../machine.parquet" ;
 nosql:store nosql:parquet ] ;
rr:subjectMap [
 rr:template "http://uri/../{MID}" ;
 rr:class msmt:SMDMachine ] ;
rr:predicateObjectMap [
 rr:predicate msmt:smdMachineName ;
 rr:objectMap [rml:reference "Name"]]
```

Fig. 2. Mapping an entity using RML.

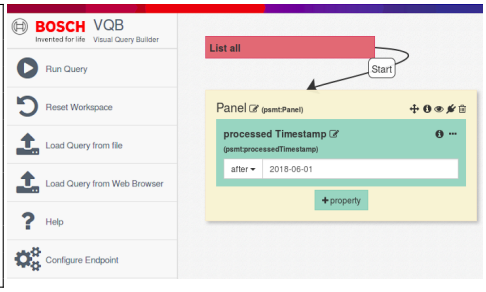

Fig. 3. Visual Query Builder.

set of user-defined mappings in a language called RML [1]. Relevant entities are loaded into Spark's *in-memory* tabular structures [11], which can be filtered, joined, grouped or sorted following the input query. Data (entities) are internally stored following a partitioning scheme similar to the Property Tables [10]. A table is created for each group of triple patterns sharing the same subject in the SPARQL query. The table attributes correspond to the properties of the triple group. These tabular data structures are not materialized, but only used during query execution and then cleared. Practically, SANSA-DL is built on the OBDA [8] principles, including ontology, a unique query language and mappings.

Ontology. In order to develop a virtual general schema on top of the data sources, SANSA-DL needs to describe data schemata using ontology classes and properties. For example, *Machine* and *Failure* are classes and *hasPanel* and *smdMachineName* are properties of the class *Machine*. The links between data schemata and ontology classes and properties are defined in form of *mappings* using RML language. This virtual general schema abstracts away schemata difference, and thus, resolves semantic conflicts found across the data sources.

SPARQL Query. The main purpose of using a unique query language in SANSA-DL is to bring together (join) various data sources and, thus, derive cross-data knowledge and insights. SPARQL is the *de facto* query language for RDF data. A SPARQL query extracts data by matching its triple templates against the stored RDF data, *e.g.*, (*?m fsmt:hasPanel "ABD3"*) matches all the triples where the property *fsmt:hasPanel* is equal to the object *"ABD3"*. The SPARQL query uses ontology properties and classes.

Mappings. In SANSA-DL, data is described (metadata) in terms of entities, *i.e.*, collections of data sharing the same structure and schema, *e.g.*, a table in a relational database or CSV file, a collection in a document database, *etc.* An entity has one or more attributes, *e.g.*, a column in a table or CSV, a field in a document, *etc.* A data source has one to many entities. In order for a data source to be queried using SANSA-DL, the schema of its containing entities

has to be mapped onto ontology terms. For this matter, RML mapping language is used. The way an entity is mapped using RML is shown in Figure 2. `MachineMap` denotes an entity mapping, `rml:source` refers to where the source is located, `nosql:store` refers to the type of data source where the entity is stored, `rr:class` specifies the class of the entity, `rml:reference` and `rr:predicate` respectively refer to what entity attribute to map to what ontology property.

2.3 Visual Query Builder (VQB)

VQB is a Web application (see Figure 3) that is built[6] to lower the complexity of querying RDF data by stakeholders who are not familiar with Semantic technologies. SPARQL queries are visualized in a UML-like model, where classes are implemented as boxes and their properties inside of them as sub-boxes and relations between classes as arrows. We further developed VQB by improving the query graph construction and adding a feature to connect it to SANSA-DL. The connection is ensured using Apache Livy[7] and an intermediate Web service. Given a VQB-built query, the Web service uses Livy to remotely trigger SANSA-DL execution of the query using Spark. It then returns the response to VQB for it to display in a grid-like manner with pagination.

3 Conclusion

The solution operated internally over the SMT data (approx. 60GB). Data Importer generated Parquet tables that are approximately 13% more compact than the original data size. The VQB used the underlying SMT ontology to generate the use case queries and trigger their execution by SANSA-DL. The queries were run and finished between 0.27 and 130s depending on their complexity. As lessons learned, the incorporation of Semantic technologies has proven their effectiveness at bridging Industrial disparate data sources together. Further, the use case domain was complex, so only with the help of an interactive visual query builder that data can be conveniently exploited by non-SPARQL experts. SANSA-DL allowed to mediate over and query disparate data sources uniformly using SPARQL even if the supported fragment did not cover all the use case queries. Thanks to its resiliency and scalability, the current solution can be deployed in a much larger environment, and thus pave the way for future large-scale semantic adoptions in the Industry 4.0.

Acknowledgments

This work was supported by the EU H2020 projects BETTER (GA 776280) and QualiChain (GA 822404), and by the ADAPT Centre for Digital Content Technology (`http://www.adaptcentre.ie/`) funded under the SFI Research Centres Programme (Grant 13/RC/2106) and co-funded under the European Regional Development Fund.

[6] Initially implemented by Lukas Leipert `https://github.com/leipert/vsb`

[7] `https://livy.incubator.apache.org`

References

1. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: A generic language for integrated RDF mappings of heterogeneous data. In: LDOW (2014), `https://rml.io`
2. Grangel-González, I.: A Knowledge Graph Based Integration Approach for Industry 4.0. Ph.D. thesis, Universitäts-und Landesbibliothek Bonn (2019)
3. Harris, S., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 query language. W3C recommendation **21**(10) (2013)
4. Kharlamov, E., Grau, B.C., Jiménez-Ruiz, E., Lamparter, S., Mehdi, G., Ringsquandl, M., Nenov, Y., Grimm, S., Roshchin, M., Horrocks, I.: Capturing industrial information models with ontologies and constraints. In: The Semantic Web - ISWC - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, Proceedings, Part II. pp. 325–343 (2016)
5. Lehmann, J., Sejdiu, G., Bühmann, L., Westphal, P., Stadler, C., Ermilov, I., Bin, S., Chakraborty, N., Saleem, M., Ngomo, A.C.N., et al.: Distributed semantic analytics using the SANSA stack. In: International Semantic Web Conference. pp. 147–155. Springer (2017)
6. Mami, M.N., Graux, D., Scerri, S., Jabeen, H., Auer, S., Lehmann, J.: Squerall: Virtual ontology-based access to heterogeneous and large data sources. In: International Semantic Web Conference. pp. 229–245. Springer (2019)
7. Mami, M.N., Graux, D., Scerri, S., Jabeen, H., Auer, S., Lehmann, J.: Uniform access to multiform data lakes using semantic technologies. Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services (iiWAS), Munich, Germany, 2-4 December (2019)
8. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Journal on Data Semantics X. Springer (2008)
9. Tavakolizadeh, F., Soto, J., Gyulai, D., Beecks, C.: Industry 4.0: mining physical defects in production of surface-mount devices. 17th Industrial Conference on Data Mining (2017)
10. Wilkinson, K.: Jena property table implementation. SSWS (2006)
11. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., et al.: Apache spark: a unified engine for big data processing. Communications of the ACM **59**(11), 56–65 (2016)