Integrating heterogeneous data sources using rule-based reasoning, backward-chaining and custom built-ins

Barry Nouwt[1] and Jack Verhoosel[1]

[1] TNO, Kampweg 55, 3769 DE, Soesterberg, The Netherlands
barry.nouwt@tno.nl,jack.verhoosel@tno.nl

Abstract. In this paper, we investigate the challenge of integrating heterogeneous, distributed data sources using semantic technologies to enable the answering of complex user queries on the combination of available data. Our solution uses a common OWL ontology that contains the concepts that represent the data in all the distributed sources providing maximum querying flexibility towards the user. Retrieval of specific data from distributed sources is done using a set of *customized* rules and built-in functions that call the data sources APIs triggered when a user issues a SPARQL query. We validated our system with forward and backward ruling strategies in a horticultural application domain. The most important lessons learned are that the approach is very flexible in the sense that only a few customized rules enable the querying of a large set of concepts. Using backward-reasoning rules largely improves the performance of the system, because the reasoner only executes those rules that it deems necessary to answer the user query.

Keywords: heterogeneous data sources, common ontology, rule reasoner

1 Introduction

In various business domains and sectors in society, data sources are distributed and heterogeneous by nature, because they come into existence separately and independently. As a consequence, combining and analyzing data from these heterogeneous, distributed data sources is a challenge. We have developed and studied a semantic technology solution that assumes that data is kept at distributed data sources and is only retrieved when needed to answer a user query. Our solution uses as a key instrument a common OWL ontology that contains the concepts that represent the data used from the distributed sources. This common ontology is made available via a triple store with a SPARQL engine that provides maximum querying flexibility on all the concepts in the common ontology. Retrieval of specific data from distributed sources is done using a set of customized rules defined in terms of the concepts and relations in the ontology as well as built-in functions that call the data sources APIs. The execution of these rules is triggered by a reasoner when a user issues a SPARQL query on the ontology and necessary data needs to be retrieved. In this paper, we discuss the validation of our system in an application domain with the usage of forward and backward ruling strategies for retrieving data using the Apache Jena's generic rule reasoner.

2 Related work

This paper continues the research described in our previous work [1]. Here we overcome the challenge of context-information by integrating data sources at the reasoner-level instead of the graph-level. We use Apache Jena's GenericRuleReasoner that uses its own rule syntax, but very similar W3C proposed rule languages are SWRL and SHACL. This work relates to research on data/ontology integration [2][6][7] that pursues similar goals, but uses other technologies. Other relevant work is the extensive research on applying semantic technologies to web services. On the one hand this research focused on giving semantic descriptions of the inputs/outputs of web services thereby enabling automatic discovery [3]. On the other hand, the research focused on how to automatically compose multiple web services together [4]. Our work can benefit from semantics descriptions of web services, however, many of today's web services do not include them. For now, we use mappings of a combination of custom Java code and reasoning rules, but in a future version we want to utilize semantic descriptions. Regarding web service composition, our approach is similar to rule-based 'planning' techniques. The main difference, however, is that we use these techniques to answer *dynamic* SPARQL queries, where each query potentially requires a different composition of web services.

3 Rule reasoning, built-ins and their application

For our research we use Apache Jena's Generic Rule Reasoner, because it integrates nicely with semantic technologies, like RDF and SPARQL. The rule reasoner uses RDF triples as its facts and also the syntax of the rules is tailored to working with these triples: *[mortality-rule: (?x rdf:type :Man) -> (?x rdf:type :Mortal)]*. This reasoner supports both the forward data-driven and backward goal-driven ruling mechanisms, but also another important feature called built-in functions. Built-in functions can be included in the head or body of rules to perform actions that do not easily fit the rule-based paradigm, such as calculations or string manipulation. For example, in the rule *[rule:(?n rdf:type :Man)(?n :cmLength ?o) -> CmInch(?o ?p)(?n :inchLength ?p)]* the built-in function *CmInch(?o ?p)* handles the conversion from cm to inch, where ?o is bound to a Man's length in cm, and the ?p will be bound by the built-in function to the converted length of the Man in inches. Apache Jena's rule reasoner keeps a register of standard built-in functions and allows developers to register custom built-in functions that can be used to realize the link with external data sources.

We have used the horticultural application domain to validate our solution as there are many stakeholders that maintain data about food products that is however not transparently available to all stakeholders in the supply chain. Therefore, we developed the HortiCube platform that contains our solution using the Common Horticultural Model (CHM) described in [5]. The HortiCube platform provides access to data sources:

1. Apple/pear yearly *production* forecast figures per variety per EU country
2. Apple/pear monthly *stock* per variety per EU country
3. UN Comtrade *import/export* data between countries in the EU.

The first two data sources are available via an Apache Jena triplestore and the Comtrade data is available via an external API provided by the UN. For each of these data sources we designed a specific ontology and mapped it to the CHM using *owl:subClassOf* and *owl:subPropertyOf* constructs. We added specific individuals for all countries in the world and most of the existing apple/pear varieties produced and stored in these countries. The actual production, stock and export data is however *not* present in the CHM, but available at the data sources indicated. We describe the use of the rule-reasoner for accessing an API of an external data source for answering the question: *"What is the production and stock of apples in The Netherlands in 2014 and how much apples have been exported to Belgium in that year?"*. This question is described as a SPARQL query on the combination of the data sources to be used, especially the external Comtrade data source rest-API. When firing this query onto the SPARQL engine, the rule-based reasoner used existing RDFS rules to reason over the mapping between CHM and Comtrade:

```
[rdfs6:(?p rdfs:subPropertyOf ?q),notEqual(?p,?q) -> [(?a ?q ?b) <- (?a ?p ?b)] ]
[rdfs9:(?x rdfs:subClassOf ?y),notEqual(?x,?y) -> [(?a rdf:type ?y) <- (?a rdf:type ?x)]]
```

In addition, we defined a generic forward-rule and a custom build-in function to get access to the Comtrade API:

```
[comtradeExportForwardRule:
(?exporting rdf:type comtrade:Country)(?importing rdf:type comtrade:Country)
(?exporting comtrade:hasCountryEnglishName ?exportingvalue)
(?importing comtrade:hasCountryEnglishName ?importingvalue)
(?product rdf:type comtrade:Apple)(?product comtrade:hasComtradeCode ?productvalue)
(?year rdf:type comtrade:Year)(?year comtrade:hasYear ?yearvalue)
comtradeExport(?exportingvalue, ?importingvalue, ?productvalue, ?yearvalue, ?export)
   ->
(?uri rdf:type comtrade:Export)(?uri comtrade:hasExportTradeValue ?export)
(?uri comtrade:hasReportingArea ?exporting)(?uri comtrade:hasPartnerArea ?importing)
(?uri comtrade:hasCommodity ?product)(?uri comtrade:hasExportPeriod ?year)]
```

The body of the rule checks whether there are Comtrade individuals of countries *?exporting* and *?importing*, an apple *?product* and a *?year* in the CHM. In addition, the body of the rule contains a built-in function *comtradeExport* that takes these and returns the specific *?export* value as result by executing an HTTPS-call to the Comtrade API. The head of the rule is meant to create a *comtrade:Export* individual in the CHM and add the retrieved *?export* value to it as well as the *?exportingcountry*, *?importingcountry*, *?product* and *?year* for which the export value was requested.

In the forward chaining, data-driven modus the rule reasoner executes the rule for each combination of triples found that satisfy the body, which can lead to unacceptable execution times. Therefore, we also used the backward chaining, goal-driven modus by simply swapping the head and body of the rule. In that case, the reasoner will *only* try to prove the overall goal, based on the restrictions in the SPARQL query, which yields an enormous performance improvement. An important restriction of the Apache Jena rule reasoner is that it allows not more than one clause in the head. As a consequence, we split the backward-rule into six rules, one for each of the goals in the head.

Finally, we evaluated our system with respect to the design of the SPARQL query. We found out that the sequence of execution of the backward-rules by the reasoner very much depends on the order of the statements in the query. It turned out that using a "filter-as-soon-as-possible" policy for the ordering of the statements is best. In this way, the reasoner is instructed to first reduce the solution space by filtering on specific values in the triples. The other statements are then used to construct the result of the query.

4 Lessons learned and future work

We have learned various lessons with our experiments. First, we found that in some scenarios the goal-driven modus of the reasoner is preferable over the data-driven modus. In our test scenario, the goal-driven modus only called the external data source once, while the data-driven modus exhaustively requested information from the external data source for every combination of possibilities of data in the body of the rules. The performance difference between the two modi changes depending on both the size of the answer and the data. Second, the ordering of the clauses in the SPARQL query can have a considerable impact on query answering time, because it filtering and scoping of query results should be done first in the query. A similar behavior occurs with the order of the clauses in rules, where the location of the built-in can cause incorrectly derived triples. Both the user and the designer should be aware to correctly order the clauses in queries and rules. Third, designing a system like we describe in this paper consists of creating the following artifacts: (1) multiple ontologies, (2) mappings between the ontologies, (3) different rules to integrate the external data sources, (4) custom built-ins for each external data source. Designing such a system requires effort, but enables generic, flexible access to distributed data sources.

Although our solution efficiently and effectively operates, there are some possibilities for future work, such as cascading of OWL and rule-reasoners, aggregation of built-in calls, backward ruling with multiple heads, missing fact indications and parallelism in the reasoning process.

References

1. B. Nouwt, "Tight integration of Web APIs with Semantic Web.," in *SALAD Workshop, Semantics 2017*.
2. A. Halevy, A. Rajaraman and J. Ordille, "Data integration: the teenage years.," *In Proceedings of the 32nd international conference on Very large data bases (pp. 9-16).,* vol. VLDB Endowment., 2006.
3. M. Klusch, P. Kapahnke, S. Schulte, F. Lecue and A. Bernstein, "Semantic web service search: A brief survey.," *KI-Künstliche Intelligenz,* vol. 30, no. 2, pp. 139-147, 2016.
4. A. L. Lemos, F. Daniel and B. Benatallah, "Web service composition: a survey of techniques and tools.," *ACM Computing Surveys (CSUR),* vol. 48, no. 3, p. 33, 2016.
5. J. Verhoosel, M. van Bekkum and Y. Verwaart, "Semantic interoperability for data analysis in the food supply chain", *International Journal on Food System Dynamics*, vol. 9, no. 1, pp. 101-111, 2018
6. N. F. Noy, "Semantic integration: a survey of ontology-based approaches.," *ACM Sigmod Record,* pp. 65-70, 2004.