Linked Data Fusion in ODCleanStore*

Jan Michelfeit and Tomáš Knap

Charles University in Prague, Dept. Software Engineering
Malostranské nám. 25, 118 00 Prague, Czech Republic
michelfeit.jan@gmail.com, tomas.knap@mff.cuni.cz

Abstract. As part of LOD2 project and OpenData.cz initiative, we are developing an ODCleanStore framework enabling management of Linked Data. In this paper, we focus on the query-time data fusion in ODCleanStore, which provides data consumers with integrated views on Linked Data; the fused data (1) has solved conflicts according to the preferred conflict resolution policies and (2) is accompanied with provenance and quality scores, so that the consumers can judge the usefulness and trustworthiness of the data for their task at hand.

The advent of Linked Data [1] accelerates the evolution of the Web into an exponentially growing information space (see the linked open data cloud[1]) where the unprecedented volume of data will offer information consumers a level of information integration and aggregation agility that has up to now not been possible. Consumers can now "mashup" and readily integrate information for use in a myriad of alternative end uses. Indiscriminate addition of information can, however, come with inherent problems, such as the provision of poor quality, inaccurate, irrelevant or fraudulent information. All will come with an associate cost of the data integration which will ultimately affect data consumer's benefit and linked data applications usage and uptake.

To overcome these issues, as part of the *OpenData.cz initiative* and *LOD2 project*[2], we are developing the *ODCleanStore (ODCS) framework*[3] (1) enabling management of Linked Data – data cleaning, linking, transformation, and quality assessment – and (2) providing data consumers with a possibility to consume integrated data, which reduces the costs of the web application development.

The overall picture of ODCS is depicted in Figure 1. ODCS processes *RDF data feeds* (collections of RDF quads, one data feed = one named graph[4]) in the *staging area*; feeds can be uploaded to the staging area by any third-party

* The work presented in this article has been funded in part by EU ICT FP7 under No.257943 (LOD2 project), the Czech Science Foundation (GAČR, grant number 201/09/H057), and GAUK 3110.

[1] http://richard.cyganiak.de/2007/10/lod/

[2] http://opendata.cz, http://lod2.eu

[3] To download the code, please visit http://sourceforge.net/p/odcleanstore

[4] RDF triples can be extended to *quads* (s, p, o, g) where g is the named graph [3] to which the data belongs. When talking about "data in the named graph g", we mean all the quads $(*, *, *, g)$).

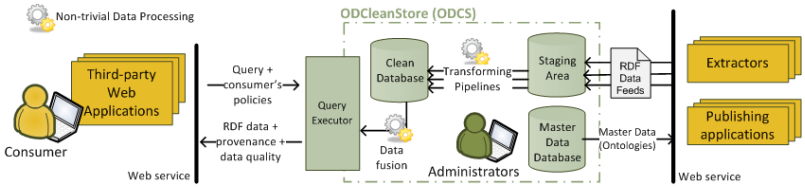

Fig. 1. ODCleanStore Framework

application registered in ODCS, e.g. by various extractors. Based on the identifier of the feed, the appropriate *transforming pipeline* is launched; the pipeline successively executes a defined (and customizable) set of transformers ensuring that data in the processed feed is cleaned, resources deduplicated and linked to already existing resources in the *clean database* or in the linked open data cloud, data is enriched with new resources, arbitrarily transformed, and the quality of the feed *(graph score)* is assessed. When the pipeline finishes, the augmented RDF feed is populated to the clean database together with any auxiliary data and metadata created during the pipeline execution, such as links to other resources or metadata about the feed's graph score.

Data consumers can query (via third-party applications) the clean database to obtain data about the certain resource (e.g. a city, such as the German city "Berlin"). Since the same resource can be described by various sources (feeds), conflicts may arise when integrating data about that city. To solve this, ODCS applies in the *data fusion algorithm* certain conflict resolution policies which resolve data conflicts in the resulting RDF data; these policies can be customized by the consumer. Furthermore, the resulting integrated RDF data is supplemented with provenance metadata (data origin) and quality scores of the integrated quads, so that data consumers can judge the usefulness and trustworthiness of the resulting data for their task at hand; the quality score is influenced by the quality of the feed the triples originate from (graph score) and by the applied conflict resolution policy [4]. The data fusion algorithm runs during query time, because consumers in different situations can have different requirements on the data.

This paper briefly describes the data fusion algorithm in ODCS in Section 1; the algorithm is fully described in [4]. The practical demonstration[5] based on the illustrative examples in Section 1 gives further insight into the work of the data fusion algorithm.

To the best of our knowledge, there is just one another linked data fusion software – Sieve – currently under development [5]. Sieve is part of Linked Data Integration Framework[6]. Differently from our approach, Sieve fuses data while

[5] http://www.ksi.mff.cuni.cz/~knap/iswc12
[6] http://www4.wiwiss.fu-berlin.de/bizer/ldif/

being stored to the clean database and not during execution of queries, thus, provides no data fusion customization during data querying.

1 Linked Data Fusion

Suppose that the clean database of ODCS contains data about the German city Berlin coming from multiple sources – DBpedia, GeoNames, and Freebase[7]. Let us assume that Alice, a data consumer, is an investigative journalist who is writing a story about Berlin; thus, she submits the keyword "Berlin" to the query execution component of ODCS and she would like to get all the information the framework knows about Berlin fused from the available sources.

When fusing data, the data fusion algorithm in ODCS has to deal with *data conflicts*, which happen when two quads have inconsistent object values for a certain subject s and predicate p; such quads are called *o-conflicting quads* and the conflicting object values of these o-conflicting quads are called *conflicting values*. The solution of the conflicts is prescribed by the conflict resolution policies, which may be specified globally or per predicate. We distinguish two types of conflict resolution policies – *deciding* and *mediating*. Deciding policies select one or more values from the conflicting values, e.g., an arbitrary value (ANY), maximum value (MAX), the value with the highest quality (BEST), or all conflicting values (ALL). Mediating policies compute new value, e.g. an average (AVG) of the conflicting values. For example, Alice may specify she would like to receive in the response all the distinct values for the subject representing Berlin and predicate `rdf:type` (deciding conflict resolution policy ALL). On the other hand, she may want to compute for the same subject average value (AVG) for the values of the predicate `geo:lat`, select the best value with the highest quality (BEST) for `rdfs:label` of Berlin, and select maximum value (MAX) from the values of the predicate `dbprop:populationTotal` of Berlin.

When describing the data fusion algorithm within execution of consumer's queries in ODCS, we suppose that the typical pre-fusing processes [2] – *schema mapping* (the detection of equivalent schema elements in different sources) and *duplicate detection* (detection of equivalent resources) has already been done. Therefore, we suppose that (1) proper mappings between ontology elements are available in the master data database in Figure 1, e.g. that `geo:lat` and `fb:location.geocode.latitude` are denoted as equivalent predicates holding latitude of Berlin, and (2) `owl:sameAs` links between resources representing the same entity (the German city Berlin) were created by the proper transformers (linkers) on the transforming pipeline.

The input to the data fusion algorithm is (1) a collection of quads from the clean database to be fused – the quads (x,*,*,*),(*,*,x,*), where x is the URI representing Berlin in some source (2) `owl:sameAs` links between URI resources occurring in the quads (output of the deduplication and schema mapping pre-fusion processes), (3) data fusion settings (including set of selected conflict

[7] Identifiers for the resource Berlin are: http://dbpedia.org/resource/Berlin, http://sws.geonames.org/2950159/, http://rdf.freebase.com/ns/en.berlin

resolution policies), and (4) graph scores of the named graphs (feeds) from which the quads originate. The output is a collection of fused quads enriched with data quality and source named graphs for each fused quad.

The fusion algorithm firstly replaces URIs of resources representing the same concept (i.e. connected by an `owl:sameAs` links) with a single URI and removes duplicate quads[8]. Consequently, quads are grouped to the sets of *comparable quads* – i.e. quads having the same subject and predicate; o-conflicting quads form subset of the corresponding comparable quads. For each set of comparable quads, two steps (Step S1 and S2) are executed: Step S1 chooses and applies a conflict resolution policy determined by the predicate of the comparable quads and Step S2 computes quality of the quads resulting from Step S1. Multiple real-world cases lead us to three factors influencing the computation of the quality of the resulting fused quads (in Step S2): (1) graph scores of the source named graphs containing the processed comparable quads, (2) number of object values within the set of comparable quads which agree on the same object value, and (3) the difference between conflicting values of the comparable and o-conflicting quads. Details of the quality computation are in [4].

2 Conclusions

This paper introduces query-time data fusion algorithm in ODCleanStore – the framework for managing Linked Data. The practical demonstration[9] shows the maturity of the algorithm and demonstrates its features – application of conflict resolution policies and computation of the quality of the fused quads. Full theoretical background behind the data fusion algorithm is in [4].

References

1. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
2. J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1:1–1:41, Jan. 2009.
3. J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, Provenance and Trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.
4. T. Knap, J. Michelfeit, and N. M. Linked Open Data Aggregation: Conflict Resolution and Aggregate Quality. METHOD 2012 : The 1st IEEE International Workshop on Methods for Establishing Trust with Open Data, COMPSAC (to appear), 2012. http://www.ksi.mff.cuni.cz/~knap/files/method.pdf).
5. P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: Linked Data Quality Assessment and Fusion. In *1st International Workshop on Linked Web Data Management (LWDM 2011) at the 15th International Conference on Extending Database Technology, EDBT 2012*, March.

[8] Quads having the same subject, predicate, object, and the named graph.
[9] http://www.ksi.mff.cuni.cz/~knap/iswc12