Provenance Policies for Subjective Filtering of the Aggregated Linked Data

Tomáš Knap
Department of Software Engineering
Charles University in Prague
Prague, Czech Republic
Email: tomas.knap@mff.cuni.cz

***Abstract*—As part of LOD2.eu project and OpenData.cz initiative, we are developing an ODCleanStore framework (1) enabling management of governmental linked data and (2) providing web applications with a possibility to consume cleaned and integrated governmental linked data; the provided data is accompanied with data provenance and a quality score based on a set of policies designed by the governmental domain experts. Nevertheless, these (objective) policies fail to express subjective quality of the data as perceived by various data consumers and different situations at their hand. In this paper, we describe how consumers can define their own situation-specific policies based on the idea of filtering certain data sources due to certain aspects in the data provenance records associated with these sources. In particular, we describe how these policies can be (1) constructed by data consumers and (2) applied as part of the data consumption process in ODCleanStore. We are persuaded that provenance policies are an important mechanism to address the subjective dimension of data quality.**

Keywords-provenance; provenance policies; linked data; linked open data; data aggregation; data quality

I. INTRODUCTION

Allover the world, governments are connecting to the uprising trend of publishing governmental data as open data [6]; open data is original non-aggregated machine readable data which is freely available to everyone, anytime, and for whatever purpose. As a result, citizens paying the government are able to see and analyze the performance of the government by observing the raw data or using third-party applications visualizing the data; companies can use the data to run their business.

Cannot we do more than just opening the data to simplify the data exploration and creation of applications on top of open data? If global identifiers were used in the form of HTTP URLs for the data exposed as open data, data could be published on these URLs and data consumers could then use the current Web infrastructure to obtain relevant information about any resource by simply inserting the HTTP URL of the resource to the browser. Furthermore, if the open data was represented as RDF (Resource Description Framework) [15] triples or quads (quads are RDF triples with a forth field representing the context – named graph [11] – to which the triple belongs), we could link data (e.g., the public contract) to other data (e.g., the supplier or price) and, thus, create a huge web of interconnected data. The idea described is precisely the idea of *linked data* [9].

The advent of linked data [9] 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* [4] and *LOD2 project* [2], we are developing the *ODCleanStore framework* [3] (1) enabling management of linked data – RDF data cleansing, linking, transforming, and quality assessing – and (2) providing consumers with a possibility to consume cleaned and integrated RDF data supplemented with data quality and provenance metadata.

The overall picture of ODCleanStore is depicted in Figure 1; data filtering module is depicted in red in Figure 1 to denote that it is not part of the current ODCleanStore state of the art and is discussed as one of the contributions further in this paper. ODCleanStore processes RDF *data feeds* and stores it to the *staging database*; feeds can be uploaded to the staging area by any third-party application registered to ODCleanStore. The RDF data feed is a set of named graphs including the RDF *data graph* (the main named graph of the feed) and graphs holding descriptive and provenance metadata. Based on the pipeline identifier within the feed's descriptive metadata, ODCleanStore engine launches the particular *data processing pipeline* containing an execution of the sequence of *data transformers (data processing units)*, which may normalize the data, deduplicate the data against the *raw data mart* or link the data to the data in the raw data mart, assess the quality of the data, or execute an arbitrary data transformation. After executing all the transformers on the given pipeline, the data feed is stored to the raw data mart together with any auxiliary data

 ISBN: 978-1-61208-247-9

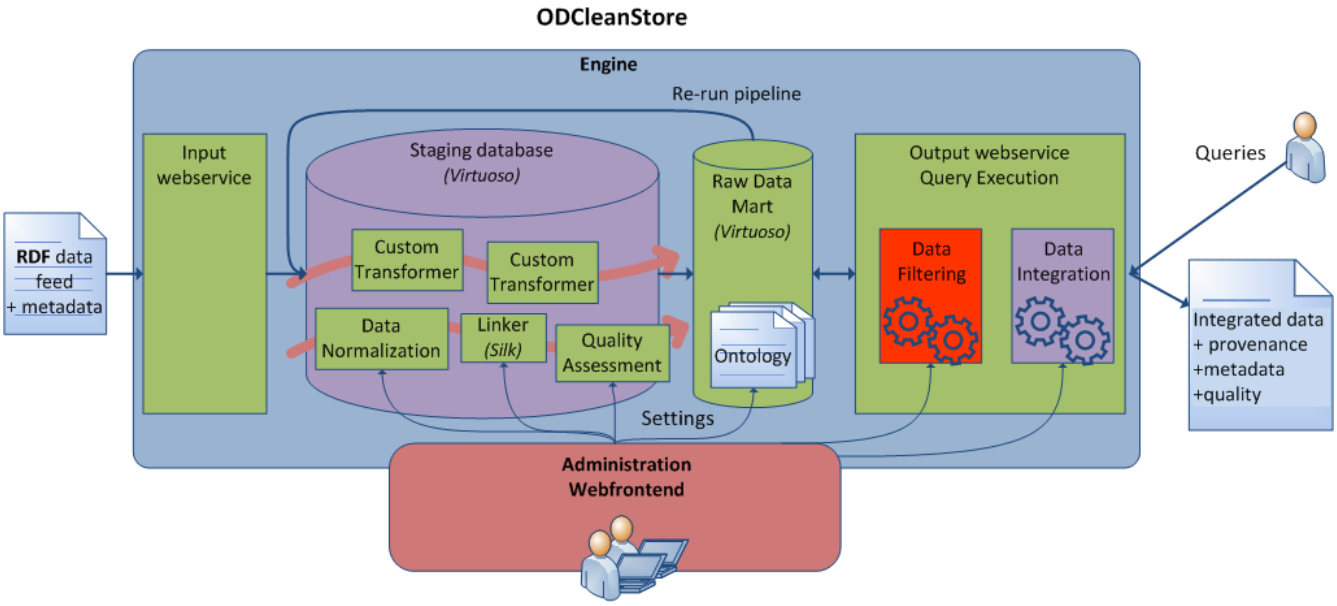

Figure 1. ODCleanStore Framework with Provenance Data Filtering Component

created during the pipeline execution, such as links to other resources or metadata about the feed's quality.

Consumers can query the raw data mart (using the *output web service*). In further text, we highlight two types of queries: *uri* queries and *keyword* queries. Suppose that the set Q_q contains quads from the raw data mart relevant to the consumer's query q; for an uri query q, Q_q contains the quads $(x,*,*,*)$,$(*,*,x,*)$, where x is the URI in the consumer's query or URI being `owl:sameAs` with the URI in the consumer's query; for a keyword query q with a sequence of keywords kw, Q_q contains the quads $(*,*,l,*)$, where the literal l contains the keywords kw. Sample keyword query may be: "Give me all you know about the Ministry of Finance of the Czech Republic".

Since the same resource can be described by various sources, using different schemas (vocabularies), data integration is necessary to provide the data consumer with an integrated view on the data. Data integration consists of three phases [10] – *schema mapping* (the detection of equivalent schema elements in different data graphs), *duplicate detection* (detection of equivalent resources) and *data fusion* (fusion the descriptions of equivalent resources).

Data integration module in ODCleanStore addresses all the integration phases – it takes into account the mappings between different schemas stored in the raw data mart and links deduplicating resources generated by the proper transformers on the data processing pipelines and then fuses the data. When fusing data about the same resource from multiple sources (data graphs), data conflicts may arise and should be solved. Thus, as part of the data fusion phase, data integration module in ODCleanStore applies certain conflict handling strategies which resolve, ignore, or avoid the data conflicts in the resulting RDF data. Provided the conflict handling strategy is to resolve the conflicts, consumer can also specify the *conflict resolution policies* driving the data fusion.

Furthermore, the resulting RDF data outputted by the data integration module in a form of quads is supplemented with a *quality score* influenced by the quality of the feed the data originates from (which is quantified on the transforming pipeline by the quality assessment transformer), the score of the data publisher (e.g., a domain http://wikipedia.org), and by the applied conflict resolution policies [16].

Thus, the ODCleanStore framework is able to addressed the objective part of the data quality by employing quality assessment transformer. Nevertheless, the information quality must be always considered w.r.t the specific (subjective) requirements of the consumer [8, 17, 20] for his particular task at hand. For example, the consumer might want to prefer data from the Czech Business Register when looking for data about companies, or use only sources verified by his boss.

These needs are partially supported – the resulting data is accompanied also with *data provenance* of the data graphs (sources) the data originates from, providing the necessary contextualization for the information consumer to analyze the (subjective) quality of the information [12, 13, 19]. However, the manual examination of such provenance and manual filtering of the resulting data based on its provenance is rather tedious work for information consumers.

Therefore, in this paper, we describe the concept of *provenance policies* which will enable the data consumer to express his subjective preferences for certain data sources

 ISBN: 978-1-61208-247-9

based on the provenance data associated with these sources; such policies must be taken into account automatically during the query execution - as a result, only data from certain sources appear in the result on the consumer's query q. We describe in the paper the format of the provenance policies and how they are applied during the query execution as part of the data filtering module in ODCleanStore to refine the data being provided to data consumers.

II. Data Provenance

We use in this paper the definition of provenance, which is based on the definition introduced by W3C Provenance Group [7] and takes some aspects from the "provenance as annotations" approach [19]: "Provenance of a resource r is a record that contains resources, agents, or processes (and their properties contextualizing them) involved in producing and delivering or otherwise influencing the resource r. "

W3C Provenance Group is defining the core provenance terms for tracking provenance on the Web; in [12], we described a W3P provenance model for the Web.

In further text, *provenance graph* is the named graph $g^p \in G$ (set of triples belonging to that graph) containing provenance information about the data graph $g \in G$; data feed inserted to the staging area always contains the data graph g and may contain provenance graph g^p.

```
<http://source.com/1> a prov:Entity ;
  dc:creator <http://purl.org/knap#me> ;
  dc:created "2011-11-18"^^xsd:date ;
  dc:source <http://source.com/2> ;
  p:hadPrimarySource <...> .
```

Listing 1. Sample Provenance Graph

Listing 1 depicts the sample provenance graph holding provenance data about the data graph `<http://source.com/1>`. As you can see, the provenance graph holds (1) the creator of the source, (2) the creation time, and (3) the primary sources from which that source was obtained (e.g., extracted); in Listing 1, the prefix `dc:` stands for `http://purl.org/dc/terms/`, `p:` stands for `http://www.w3.org/ns/prov#`.

III. Provenance Policies

We define a provenance policy $p \in P$ as a tuple $(cond, weight)$, where $cond \in C$, C is a set of all valid `GroupGraphPatterns` in the SPARQL language [5]; function $w(p)$, $w : P \rightarrow [-1, 1] \setminus \{0\}$ quantifies the $weight$ of the policy p, $w(p) \in (0, 1]$ determines a *positive policy* and $w(p) \in [-1, 0)$ determines a *negative policy* p.

A provenance policy $p = (cond,\ weight) \in P$ can be successfully applied to the provenance named graph g^p if and only if a SPARQL query "ASK FROM NAMED g^p WHERE $\{cond\}$" returns $true$. The successful application is expressed as $a(p, g^p) = true$; otherwise, if the policy was not successfully applied, $a(p, g^p) = false$; $a : P \times G \rightarrow \{true,\ false\}$. If $a(p, g^p) = true$, then the policy p changes the *provenance score* of the graph g according to $weight$. Positive policy always increases the provenance score, negative policy decreases.

The condition $cond \in C$ of a policy $p = (cond,\ weight) \in P$ may use the variable `odcs:graph` which is replaced by the particular data graph g being processed before the query is sent to the underlying SPARQL engine; full URL for the odcs prefix is: http://ld.opendata.cz/infrastructure/odcleanstore/. Suppose a condition $cond$ = {`odcs:graph dc:creator <http://purl.org/knap#me>`}; such condition is matching all the graphs g^p containing the triple with the subject being the URI of the graph g, the predicate `dc:creator` and the object `<http://purl.org/knap#me>`, i.e., all graphs created by the agent `<http://purl.org/knap#me>`.

We define a function $s_{prov} : G \times \mathcal{P}(P) \rightarrow (0, 1]$ quantifying the *provenance score* of the graph g based on the weights of the policies $P_a = \{p \in P \mid a(p, g^p) = true\}$ successfully applied to g^p as:

$$s_{prov}(g, P_a) = min\{\frac{\prod_{p \in P_a}(1 + w(p))}{C}, 1\}$$

The constant $C \in \mathbb{N}$ defines the upper boundary for the influence of the positive policies; if $\prod_{p \in P_a}(1 + w(p)) > C$, the provenance score $s_{prov}(g, P_a)$ is equivalent to the case when $\prod_{p \in P_a}(1 + w(p)) = C$. The constant C should be set based on the average proportion of positive and negative policies and the average absolute number of positive policies applied to the graphs. Furthermore, the default provenance score of any graph to which no policy was successfully applied should be equal to $1/C$.

IV. Applying Provenance Policies

The application of provenance policies is part of the data filtering component depicted in Figure 1. The data filtering component is executed during query execution after fetching the quads, Q_q, relevant for the uri or keyword query q from the raw data mart. An output of the data filtering component is the collection of quads, $\widetilde{Q_q} \subseteq Q_q$, which is created as defined further in Algorithm 1; such output is used as the input to the data integration component in Figure 1. The quality score computed in the data integration module of ODCleanStore [16] also takes into account the provenance score s_{prov} computed for the graphs involved in $\widetilde{Q_q}$.

The inputs to Algorithm 1 are (1) the quads, Q_q, being fetched as a result of the consumers query q, (2) policies, $P_c \subset P$, defined by the consumer c executing the query, (3) constraints, $F_q \subset F$, customizing the behavior of the algorithm for the given query q, and (4) the desired threshold, $\kappa \in [0, C]$, for the provenance score. The output is the refined set of quads, $\widetilde{Q_q}$, $|Q_q| \geq |\widetilde{Q_q}|$, belonging to data graphs g having the provenance score above the threshold κ. The algorithm can enforce the following constraints F on

Algorithm 1 Provenance Policies Application

Input: Q_q, $P_c \subseteq P$, $F_q \subseteq F$, κ
Output: $\widetilde{Q_q} = applyProvPolicies(Q_q, P_c, F_q, \kappa)$
1: $\widetilde{Q_q} \leftarrow \emptyset$
2: $G_q \leftarrow \{g \mid \exists(*,*,*,g) \in Q_q\}$
3: **for all** graphs $g \in G_q$ **do**
4: $P_a \leftarrow \emptyset$, $flagResult \leftarrow true$
5: **for all** policies $p \in P_c$ **do**
6: **if** $a(p, g^p)$ **then**
7: $P_a \leftarrow P_a \cup \{p\}$
8: **end if**
9: **end for**
10: **for all** flags $f \in F$ **do**
11: $flagResult \leftarrow flagResult \wedge eval(P_a, f)$
12: **end for**
13: **if** $flagResult$ **then**
14: **if** $s_{prov}(g, P_a \geq \kappa_{pp}$ **then**
15: $\widetilde{Q_q} \leftarrow \widetilde{Q_q} \cup \{(*,*,*,g)\}$
16: **end if**
17: **end if**
18: **end for**
19: **return** $\widetilde{Q_q}$

the provenance graphs of the data graphs whose triples are included in $\widetilde{Q_x}$:

- NoNeg - Negative policy shall not be successfully applied to the provenance graph.
- ExistsPos - A positive policy shall be successfully applied to the provenance graph.
- PosMajority - Number of positive policies successfully applied to the provenance graph shall prevail over the number of negative policies.
- PolMandatory - At least one policy shall be successfully applied to the provenance graph.

In Lines 3 – 18 of Algorithm 1, the provenance policies are successively applied to the graphs $g \in G_q = \{g \mid \exists(*,*,*,g) \in Q_q\}$. In Lines 5 – 9, the set P_a of successfully applied policies is constructed; based on that, in Lines 10 – 12, the function $eval$, $eval : \mathcal{P}(P) \times F \rightarrow \{true, false\}$, progressively checks the satisfaction of all the constraints F_q w.r.t. to the set of polices P_a and directly influences the construction of $\widetilde{Q_q}$. The function $eval(P_a, f)$ is defined as follows for the set of successfully applied policies $P_a \subset P_c \subset P$ and the constraint $f \in F$:

$$eval(P_a, NoNeg) = \begin{cases} true\,\{\nexists p \in P_a : w(p) < 0\} \\ false\,\{otherwise\} \end{cases}$$

$$eval(P_a, ExistsPos) = \begin{cases} true\,\{\exists p \in P_a : w(p) > 0\} \\ false\,\{otherwise\} \end{cases}$$

$$eval(P_a, PosMajority) = \begin{cases} true\left\{|\{p|w(p) > 0\}| > \frac{|P_a|}{2}\right\} \\ false\,\{otherwise\} \end{cases}$$

$$eval(P_a, PolMandatory) = \begin{cases} true\,\{|P_a| > 0\} \\ false\,\{otherwise\} \end{cases}$$

In Lines 14 – 16, if all the flags F_q are satisfied for the given P_a, the algorithm tests whether the condition on the threshold $\kappa \in [0, C]$ of the provenance score is satisfied; if yes, the quads of the data graph g are added to $\widetilde{Q_q}$ in Line 15. Otherwise, the quads associated with the processed graph g are not included in $\widetilde{Q_q}$.

The time complexity of Algorithm 1 is $O(|Q_q| + |G_q||P_c|O(a(p, g^p)))$, where $O(|Q_q|)$ yields from loading the quads to the memory and $O(a(p, g^p))$ is the time complexity of applying a single policy $p \in P_c$ to the provenance graph g^p. Space complexity is at least $O(|Q_q|)$, because the quads has to be loaded to the memory, but depends also on the SPARQL queries being executed as part of the policy application.

V. Related Work

Researchers have developed and investigated various policy languages to describe trust, quality, and security requirements on the Web, such as [8, 14]; a variety of access control mechanisms generally based on policies and rules have been developed, such as [18]. In linked data framework WIQA (Web Information Quality Assessment Framework) [8], users can specify policies in the form of RDF graph patterns using the WIQA-PL policy language; they can filter the information in their local storage according to the selected policy, and get justifications for "why" a given information satisfies a set of policies. WIQA-PL and our policy language are both based on the SPARQL language; the grammar for WIQA-PL is not aligned with the latest SPARQL specification. A WIQA-PL policy enables to define which information is filtered positive; ODCleanStore supports both positive and negative filtering. WIQA does support the provision of justifications by extending the SPARQL language with the construct EXPL; in ODCleanStore, justifications are represented by the list of policies being applied to the resulting data.

VI. Conclusions and Future Work

In this paper, we described the concept of provenance policies – motivation for provenance policies, how these policies can be constructed, and how they can be applied as part of the data filtering component in ODCleanStore when the query is prepared for the data consumer. We are persuaded that provenance policies are an important mechanism to address the subjective dimension of data quality on the Web.

 ISBN: 978-1-61208-247-9

Future work involves the evaluation of the provenance policies' usability. To that end, we will develop a linked data browser [9] using the ODCleanStore's output web service's results and supporting data consumers with simple user interface to (1) manage provenance policies and (2) select iteratively relevant provenance policies which should be applied to the data resulting from the given query. Future work also involves examining the reasoning possibilities within provenance graphs, which affects the efficiency of provenance policies application.

VII. Acknowledgments

The work presented in this article has been funded in part by GAUK 3110, project of the internal grant agency of the Charles University.

References

[1] Linked Open Data Cloud. http://richard.cyganiak.de/2007/10/lod/ (Online, retrieved: January, 2013).

[2] LOD2 Project. http://lod2.eu (Online, retrieved: January, 2013).

[3] ODCleanStore. http://sourceforge.net/p/odcleanstore (Online, retrieved: January, 2013).

[4] OpenData.cz Initiative. http://opendata.cz (Online, retrieved: January, 2013).

[5] SPARQL Query Language - Group Graph Pattern. http://www.w3.org/TR/rdf-sparql-query/#rGroupGraphPattern (Online, retrieved: January, 2013).

[6] The Open Data Handbook. http://opendatahandbook.org/en/ (Online, retrieved: January, 2013).

[7] W3C Provenance Working Group. http://www.w3.org/2011/prov (Online, retrieved: January, 2013).

[8] C. Bizer and R. Cyganiak. Quality-driven Information Filtering Using the WIQA Policy Framework. *Web Semantics*, 7(1):1–10, 2009.

[9] 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.

[10] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1:1–1:41, Jan. 2009.

[11] 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.

[12] A. Freitas, T. Knap, S. O'Riain, and E. Curry. W3P: Building an OPM based provenance model for the Web. *Future Generation Comp. Syst.*, 27(6):766–774, 2011.

[13] O. Hartig. Provenance Information in the Web of Data. In *Linked Data on the Web (LDOW 2009), http://events.linkeddata.org/ldow2009/papers/ldow2009_paper18.pdf*, April 2009.

[14] L. Kagal, T. Finin, and A. Joshi. A Policy Based Approach to Security for the Semantic Web. In *The SemanticWeb - ISWC 2003, Florida, USA*, pages 402–418, 2003.

[15] G. Klyne and J. J. Carroll, editors. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. World Wide Web Consortium, Feb. 2004.

[16] T. Knap, J. Michelfeit, and M. Necaský. Linked Open Data Aggregation: Conflict Resolution and Aggregate Quality. In *COMPSAC Workshops, Izmir, Turkey*, pages 106–111, 2012.

[17] S. A. Knight and J. Burn. Developing a Framework for Assessing Information Quality on the World Wide Web. *Informing Science Journal*, 8:159–172, 2005.

[18] K. Lawrence and C. Kaler. WS-Trust Specification. Technical report, 2007, http://docs.oasis-open.org/ws-sx/ws-trust/200512.

[19] L. Moreau. The Foundations for Provenance on the Web. *Found. Trends Web Sci.*, 2(2–3):99–241, Feb. 2010.

[20] F. Naumann and C. Rolker. Assessment Methods for Information Quality Criteria. In *Proceedings of the International Conference on Information Quality*, pages 148–162, 2000.

 ISBN: 978-1-61208-247-9