Using SPARQL and SPIN for Data Quality Management on the Semantic Web

Christian Fürber and Martin Hepp

Universität der Bundeswehr München, E-Business & Web Science Research Group
Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
c.fuerber@unibw.de, mhepp@computer.org

Abstract. The quality of data is a key factor that determines the performance of information systems, in particular with regard (1) to the amount of exceptions in the execution of business processes and (2) to the quality of decisions based on the output of the respective information system. Recently, the Semantic Web and Linked Data activities have started to provide substantial data resources that may be used for real business operations. Hence, it will soon be critical to manage the quality of such data. Unfortunately, we can observe a wide range of data quality problems in Semantic Web data. In this paper, we (1) evaluate how the state of the art in data quality research fits the characteristics of the Web of Data, (2) describe how the SPARQL query language and the SPARQL Inferencing Notation (SPIN) can be utilized to identify data quality problems in Semantic Web data automatically and this within the Semantic Web technology stack, and (3) evaluate our approach.

Keywords: Semantic Web, Linked Data, Data Quality Management, SPARQL, SPIN, RDF, Ontologies, Ontology-Based Data Quality Management

1 Introduction

Due to the tight coupling of real-world processes and data, poor data quality may lead to errors in business processes or to wrong decisions, both causing additional costs. Also, data quality can impact product and service quality and the satisfaction of customers and employees [3]. According to *Redman*, the average total costs of poor data quality are as high as 8-12 % of a company's revenues [20]. In 2002, the Data Warehousing Institute estimated that poor data quality costs U.S. companies more than 600 billion US Dollar annually [19]. Those estimates are strong indicators for the significant impact of data quality on business success.

Semantic Web technologies aim to attach data structure, typed links, and axiomatically represented implicit facts to such data that is available on the Web. The goal is to empower computers to better extract, combine, interpret, and reuse the data [4]. A major share of such data originates from existing relational databases and is being lifted by mapping database schema elements to Web ontologies. Ontologies are commonly understood as conceptual models of a domain of interest that (1) aim at representing a model agreed among multiple individuals and organizations, and valid for multiple contexts, and that (2) contain formal axioms to reduce the ambiguity of the conceptual elements [5].

Businesses and public institutions have already started to publish significant amounts of non-toy data on the Web using Web ontologies. For example, BestBuy Inc., one of the largest US retail chains for consumer electronics products, has started to publish its full catalog [6] using the GoodRelations ontology [7]. O'Reilly Media has also begun to expose their products using GoodRelations in the RDFa syntax [24]. In addition to the growing number of data published directly by the owners of the data source, the enterprise OpenLink Software has released a middleware technology called "Sponger cartridges" that creates, on the fly, RDF representations of Amazon, eBay, and other commerce sites using the GoodRelations ontology by accessing vendor-specific APIs [12]. This makes an unprecedented amount of actual business data available on the Web of Linked Data.

However, the process of lifting existing data sources to the RDF data model and Web ontologies like GoodRelations usually replicates existing data quality problems from the original representation. While sophisticated conversion scripts and middleware components can filter out some of the problems, the negative impact of data quality issues will grow on the Web of Data, because the data will be used in more applications and in more different contexts. The amount and impact of any problems will increase accordingly.

In this paper, we describe how data quality problems in Semantic Web data originating from relational databases can be identified and classified, and this within the Semantic Web technology stack. Our approach is motivated by three main assumptions: (1) quality checks based on ontologies are highly reusable, in particular in multiple-source scenarios that utilize ontologies as means for data integration, (2) the application of Semantic Web technologies facilitates the collective emergence of data quality knowledge on the Web, and (3) it is likely that many relational data sources will be exposed to the Semantic Web without previously applying strong quality checks. Our proposal can also be applied to relational databases inside closed settings, e.g. within a single enterprise.

While existing data quality management tools usually hide the rules inside application code or regular expressions, our approach makes the rules much more accessible and easier to maintain, extend, and share, because the rules are kept in the form of a library of SPARQL queries that are human-readable and platform-neutral.

2 Overview of Data Quality Problems

A common, but rather generic definition of high data quality is when it is "fit for use", i.e. that the data meets the required purpose [1]. This popular definition of data quality is based on the subjective perception of data quality by data consumers, encompassing several dimensions, such as accessibility, completeness, and relevance; see [1] for a complete list of established data quality dimensions. Despite the importance of data consumers' perception of data quality, this perspective is not solely sufficient for the development of algorithmic approaches for identifying data quality problems. A more technical understanding of quality is to require data to be "free of defects" [2]. While still rather generic, this allows categorizing data quality problems according to their cause or effect. In the following, we summarize the work of [8-11] and provide a typology of data quality problems (see Table 1).

We trace back the types of quality problems found in literature to four basic types, namely inconsistency, lack of comprehensibility, heterogeneity, and redundancy. In the following sections, we describe these basic types of data quality problems. For a detailed discussion of the original data quality problems, we refer to [8-11].

Our current main interest is to improve the quality of literal values in ontology-based knowledge representations, which have so far not attracted a lot of interest from the formal ontology communities. In this paper, we focus on data quality problems in single-source scenarios, i.e. such within one database. So far, we have developed generally usable identification rules for syntactical errors, missing values, unique value violations, out of range values, and functional dependency violations, all of which are explained in more detail in section 3.

Ontologies also promise significant benefits when data from multiple sources is being combined during retrieval or integration, e.g. as described in [21], but that is part of our ongoing research. Also, we did not yet investigate problems within the conceptual model of ontologies themselves.

Table 1. Common data quality problems in single-source scenarios [8-11]

Data Quality Problem	Basic Type
Word transposition/Syntax violation	Inconsistency
Outdated values	Inconsistency
False values	Inconsistency
Misfielded values	Inconsistency
Meaningless values	Comprehensibility
Missing values	Inconsistency
Out of range values	Inconsistency
Invalid substrings	Inconsistency
Mistyping / Misspelling errors	Inconsistency
Imprecise values	Comprehensibility
Unique value violation	Inconsistency
Violation of a functional dependency	Inconsistency
Referential integrity violation	Inconsistency
Incorrect reference	Inconsistency
Contradictory relationships	Inconsistency
Existence of synonyms	Heterogeneity, Redundancy
Existence of homonyms	Comprehensibility
Approximate duplicate tuples	Redundancy
Inconsistent duplicate tuples	Redundancy, Inconsistency
Business domain constraint violation	Inconsistency
Outdated conceptual elements	Inconsistency

2.1 Representational Inconsistency

Inconsistency subsumes all data quality problems that originate from an actual state σ' of an element E to differ from the required state σ for E. Thereby, the element E can be (1) syntax, (2) the lexical representation of a value, (3) a data type, (4) a schema element, or (5) a relationship. For example, the value for an attribute “date” may require a syntax (E) of state DD/MM/YYYY (σ), but the syntax (E) of a value could actually have the state YYYY/DD/MM (σ’). So if σ’(E) ≠ σ(E), we call σ’ inconsistent to σ. In other words, the actual state of the element is inconsistent to the required state of the element. Functional dependency violations are not fully covered by this formula. According to [13], functional dependencies exist if a value ν_1 of an attribute α_1 requires specific values ν_n of one or more other attributes α_n in the representation. Hence, in contrast to other inconsistency problems, functional dependencies encompass states of more than one element. This also applies for referential integrity violations, incorrect references, inconsistencies among duplicate tuples, and for certain types of business domain constraint violations.

2.2 Comprehensibility

We define comprehensibility as the condition of data to be correctly interpreted by other applications or users. We further break down comprehensibility into ambiguity and vacuity. Ambiguity is if an instance or a schema element can represent two or more meanings that are treated differently by any consumer of the data. A typical case is the usage of homonyms without providing any context. We consider instances or schema elements that have no meaning at all in the presented context as vacuous. It is usually difficult to define hard criteria for comprehensibility, because this property often depends on the amount of context attached to the data and on the amount of background knowledge available to the interpreting agent. Data that is comprehensible within a closed enterprise setting may become incomprehensible when consumed on a Web scale due to the lack of contextual information.

2.3 Heterogeneity

Heterogeneity as a type of data quality problems subsumes all cases in which the representation of identical information varies. Heterogeneity mostly heavily occurs in multiple-source scenarios and can be broken down into *structural* heterogeneity and *semantic* heterogeneity. In cases of structural heterogeneity, the same real-world domain is represented by different schema elements. Semantic heterogeneity also constitutes a difference in the intension of the compared schemata with overlapping elements [9].

2.4 Redundancy

Redundancy problems exist when the same real-world entity or relationship is represented more than once and are not constrained to multiple-source scenarios. Inconsistency problems frequently co-occur with redundancy problems if some of the attribute values of the redundant tuples differ in meaning.

3 Identifying Data Quality Problems with SPARQL and SPIN

In this section, we describe our approach to identify data quality problems in Semantic Web data through the use of the SPARQL Inferencing Notation (SPIN) [17]. First, we describe the architecture of our approach. Next, we show how respective rules for the automatic identification of data quality problems in Semantic Web data can be designed and used. Finally, we evaluate our approach and outline open issues and limitations.

3.1 Architecture for Ontology-Based Data Quality Management (OBDQM)

A key goal of our approach is to handle data quality problems entirely within the Semantic Web technology stack and to employ existing technologies from the Semantic Web community. This allows using the Semantic Web itself for the collective emergence of data quality rules, i.e., users can create, improve, and share knowledge about spotting and curing data quality problems on the Semantic Web.

For the extraction of relational data we use D2RQ[1]. With D2RQ, we can extract data from a relational database into an RDF representation. This step will be optional for data that is already published as RDF. After extracting the data, we can import the data file into TopBraid Composer Free Edition[2] or another environment that supports SPIN[3]. SPARQL[4] is a query language for querying RDF data. SPIN is a framework that utilizes SPARQL to facilitate the definition of constraints and inference rules in ontologies. When applying the constraints on the ontology, SPIN can flag all problematic data elements and list them in a report. The defined rules can be attached to a class of entities in the form of SPARQL queries. With the use of SPARQL query templates in SPIN, it is possible to define generic queries with high reusability [17].

When lifting relational data to RDF automatically, we usually get very simple ontology structures based on the elements of the database. This requires refinements to enable more sophisticated reasoning. For example, the domain and range definitions of properties are usually not available from the extraction. In single-source scenarios, such can be added directly to the extracted ontology using an ontology editor. In multiple-source scenarios, however, it might be more suitable to create a global ontology with mappings to the local database schemata to enable reasoning with a source-independent vocabulary [18]. The mappings from the local database schemata to the global ontology can be created with D2RQ as well. Figure 1 illustrates the basic approach.

To facilitate the identification of data quality problems, we have to create formalized definitions of the expected data quality problem types. For this purpose, we employ SPARQL query templates. The templates have to be customized and can be attached to the class of entities that contains the data to be checked. They are attached using SPIN constraints. After the creation of the query templates and its customization, SPIN can be used to identify, flag, and report each data quality

[1] http://www4.wiwiss.fu-berlin.de/bizer/d2rq/

[2] http://www.topquadrant.com/products/TB_Composer.html#free

[3] http://spinrdf.org/spin.html

[4] http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

problem. Thus, further analysis of the presented data quality problems can be performed by domain experts; alternatively, heuristics for solving the problems can be triggered automatically.

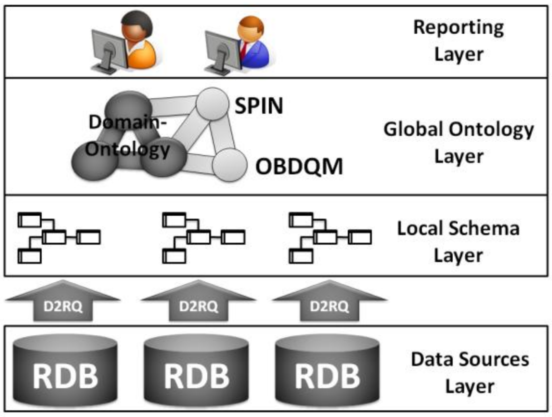


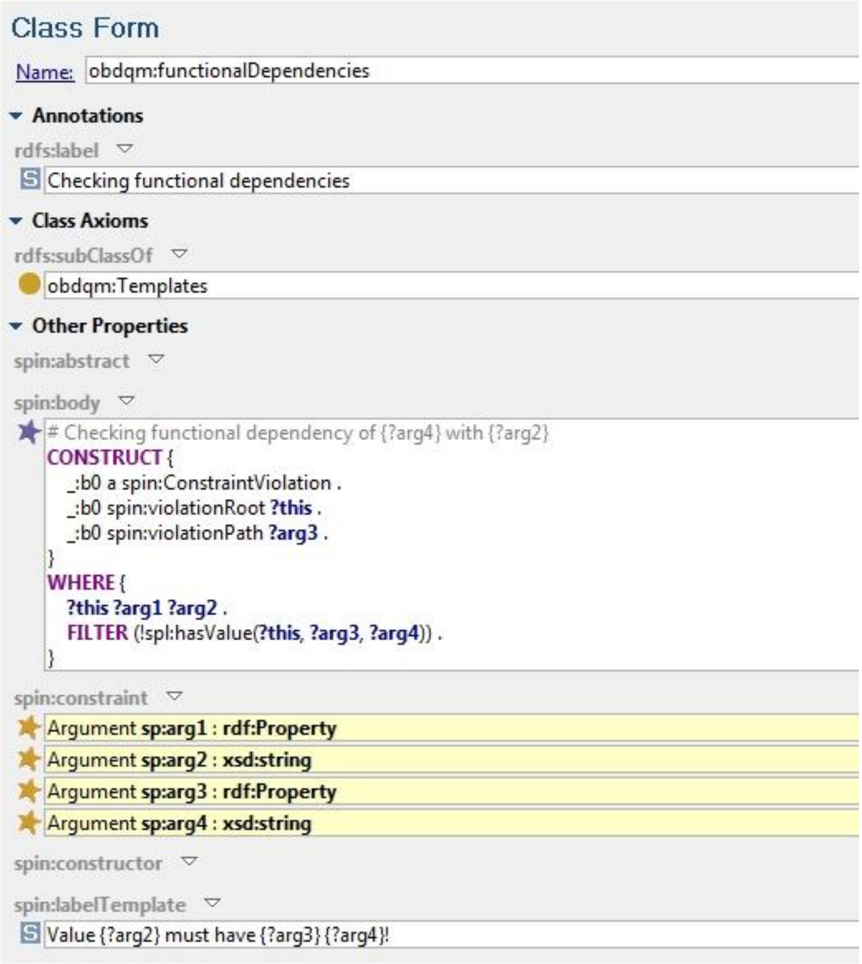
Fig. 1. Architecture of ontology-based data quality management

3.2 Identification of Data Quality Problems with SPIN

With the proposed architecture, we can focus on the main problem, i.e., the automated identification of data quality problems. For that, we define generic SPARQL query templates based on the typology of data quality problems from section 2.

As explained in section 2, inconsistency in the representation exists if the real state of an element does not meet the expected state of an element. Thus, in order to identify data quality problems with SPARQL queries, we need to express nearly all states of an element that do not meet the expected state. In other words, every SPARQL query template used to identify data quality problems has to define either all legal or all illegal states of an element. In the case of OWL object properties, the amount of axioms determines the effort for spotting respective problems. A greater formal account of the ontology, e.g. by including disjointness axioms, simplifies the rules design for data quality management. For OWL datatype properties, the use of intervals, regular expressions, and negations can reduce a lot of manual effort.

We define ASK and CONSTRUCT queries in the property `spin:body` of the generalized query template with variables, as depicted in Figure 2. The element types of the variables have to be defined in the property `spin:constraint` in order to support customization of the query. Typical element types in this case are classes, properties, or literals. Since the answer of ASK queries can only be true or false, they are especially suitable to simply flag data quality problems. When the query result returns true, the problems will be flagged by SPIN tools automatically. If the query contains more than one triple in the WHERE clause, i.e. class, property, and literal combinations, it is recommended to use a CONSTRUCT query statement with the properties `spin:ConstraintViolation`, `spin:violationRoot`, and `spin:violationPath` to correctly flag the data quality problem where it occurs. Finally, we only have to define an error message in the property `spin:labelTemplate` that will be shown with the flagged data quality problem.



Class Form

Name: obdqm:functionalDependencies

▾ Annotations

rdfs:label ▽

S Checking functional dependencies

▾ Class Axioms

rdfs:subClassOf ▽

● obdqm:Templates

▾ Other Properties

spin:abstract ▽

spin:body ▽

```
# Checking functional dependency of {?arg4} with {?arg2}
CONSTRUCT {
   _:b0 a spin:ConstraintViolation .
   _:b0 spin:violationRoot ?this .
   _:b0 spin:violationPath ?arg3 .
}
WHERE {
   ?this ?arg1 ?arg2 .
   FILTER (!spl:hasValue(?this, ?arg3, ?arg4)) .
}
```

spin:constraint ▽

Argument **sp:arg1 : rdf:Property**

Argument **sp:arg2 : xsd:string**

Argument **sp:arg3 : rdf:Property**

Argument **sp:arg4 : xsd:string**

spin:constructor ▽

spin:labelTemplate ▽

S Value {?arg2} must have {?arg3} {?arg4}!

Fig. 2. Properties of a SPARQL query template in TopBraid Composer

For the identification of syntax violations in literals, we use regular expressions in the SPARQL query in order to define the allowed characters in the literal and simply negate it. SPIN tools will flag all literals with syntactical states that do not satisfy the regular expression.

For the identification of functional dependency violations, we focused on defining bilateral dependencies in the generic query template, i.e., a value ν_1 of attribute α_1

requires a certain value ν_2 for attribute α_2. This can be extended to multilateral relationships by using this query template multiple times. In our example we would additionally define that value ν_1 of attribute α_1 also requires a certain value ν_3 for attribute α_3. For instance, we can define that the city "Las Vegas" can only have a corresponding country literal "USA" and a corresponding state literal "NV". This example requires the definition of two customized queries. In a later extension, we will tap existing Linked Open Data resources like DBPedia as references for allowed value combinations, because such resources provide a vast amount of relevant value instances and information about valid combinations.

Table 2. Generalized SPARQL queries for the identification of data quality problems

Data Quality Problem	Generalized SPARQL Query

Missing values	ASK WHERE {
	    ?this ?arg1 "" .
	}

Functional dependency violation	CONSTRUCT {
	    _:b0 a spin:ConstraintViolation .
	    _:b0 spin:violationRoot ?this .
	    _:b0 spin:violationPath ?arg3 .
	}
	WHERE {
	    ?this ?arg1 ?arg2 .
	    FILTER (!spl:hasValue(?this,
	?arg3, ?arg4)) .
	}

Syntax violation (only letters allowed)	ASK WHERE {
	    ?this ?arg1 ?value .
	    FILTER (!regex(str(?value),
	"^([A-Za-z,.])*$")) .
	}

Out of range value (lower limit)	ASK WHERE {
	    ?this ?arg1 ?value .
	    FILTER (?value < ?arg2) .
	}

Out of range value (upper limit)	ASK WHERE {
	    ?this ?arg1 ?value .
	    FILTER (?value > ?arg2) .
	}

Unique value violation	CONSTRUCT {
	    _:b0 a spin:ConstraintViolation .
	    _:b0 spin:violationRoot ?a .
	    _:b0 spin:violationPath ?arg1 .
	}
	WHERE {
	    ?a ?arg1 ?uniqueValue .
	    ?b ?arg1 ?uniqueValue .
	    FILTER (?a != ?b) .
	}

We also created generic queries for the identification of missing values, unique value violations, and out of range values. Missing values can be detected with a simple query searching for empty literal values. This works for numeric and string data types.

The identification of unique values requires a search of equal literals in the same data type property of a different tuple. Since we are again using more than one triple pattern in the WHERE clause, it is necessary to define the location of the potential violation in a CONSTRUCT statement so that the URI of the problem can be reported. Finally, out of range values can only occur with numeric data types. Hence, they can easily be detected with the relational operators "less than" or "greater than", as long as datatypes are properly attached to the RDF literals. For more flexibility, we created two queries to identify out of range values, one for values surpassing the upper limit and one for values below the lower limit. Hence, it is possible to define the legal range either as a single point boundary or in the form of an interval. The limits have to be set during the customization of the queries. All of our queries are summarized in table 2.

4 Related Work

Data quality problems have been addressed by database research for over a decade. In the Semantic Web research community, the problem of data quality is a rather new topic. With growing adoption of the Semantic Web, the diversity of data sources that will be lifted to RDF and the loss of contextual information when reusing data on a Web scale will increase the importance of data quality research for the Web of Linked data. Most existing work from the Semantic Web does not address data quality at the instance level. In particular, errors that are not directly accessible at the logical level received little attention. In the following, we summarize the most relevant previous works.

Ji et al. describe a plug-in called RaDon (Repair and Diagnosis on Ontology Networks) for ontology modeling software, which tries to extend capabilities of existing reasoners to detect inconsistencies [16]. RaDon focuses on logical contradictions when mapping ontologies to each other. The tool assumes that the single ontologies are already consistent and coherent, and focuses on repairing mappings between ontologies. *Hartig* proposed a provenance model for Web data [22]. It is based on the finding that information on the provenance of data can be used to predict the perceived quality of data by its consumers. This provenance model considers data access and data creation. In [23], he complements that work by a framework to extent Semantic Web data with trust values. The trust values are based on subjective perceptions about the query object. Although the latter two approaches provide more transparency about the underlying data sources and its potential trustworthiness, they do not directly identify data quality problems. Hence, they do not provide enough information to spot and repair data quality problems on the instance level.

The approach described by *Wang et al.* in [14] uses a task ontology for describing data cleansing tasks to be performed over existing information systems. Users are required to define a "cleaning" goal, which is translated into queries on the knowledge base to identify adequate cleaning methods. Based on the query results, an appropriate cleansing algorithm can be applied. *Grüning* describes a domain-specific example of data quality management for energy companies that partly uses ontologies [15]. In a training phase, domain experts have to flag data that exhibit data quality problems. Based on the annotations of the training phase, algorithms can be trained to identify and annotate data quality problems automatically. Unfortunately, this excellent approach currently focuses on data quality management for the energy industry only. Moreover, it only considers outlier analysis, redundancies, functional dependency violations, and suspicious timestamps. Although the use of learning algorithms saves a lot of effort, wrongly trained algorithms may limit the impact of that work.

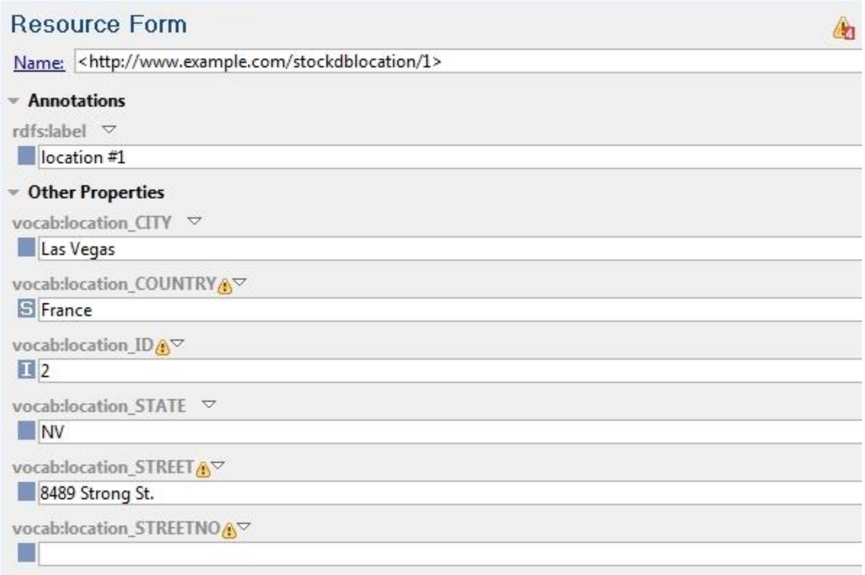
To the best of our knowledge, there is currently no holistic approach that (1) provides a domain-independent data quality management methodology for Semantic Web data, and that (2) handles those problems entirely within the Semantic Web technology stack.

5 Evaluation

In order to evaluate our approach, we created a small sample MySQL database consisting of four tables with data about products and their inventory locations. For the evaluation scenario, we assumed that we want to publish the data on the Semantic Web. The sample data contained errors for different inconsistency problems, e.g. the city “Las Vegas” was wrongly located in the country of “France”. We dumped the database completely into an RDF/XML file using a script from D2RQ. The script created individual ontology classes for each of the database tables, data type properties for each of the table columns, and RDF literals from the attribute values. After the extraction of the data, we refined the raw ontology in TopBraid Composer. First, we changed the `rdf:type` to `owl:Class` for the “table-classes”, defined the four classes as subclasses of `owl:Thing`, and assigned domains and ranges to the datatype properties.

After refining the extracted ontology, we customized the generic queries for our ontology. The exclamation marks in Fig. 3 show the identified data quality problems of a certain tuple. In this example, we defined a functional dependency between the city “Las Vegas” and the country “USA”. Moreover, we defined that the properties `vocab:location_COUNTRY` and `vocab:location_STREET` should only contain letters and that the property `vocab:location_STREETNO` must always have a literal value. Finally, we defined that the property `vocab:location_ID` must only contain unique values. Any new data of this class underlies the same quality checks.

The evaluation based on the sample data shows that the developed generic rules are suitable to identify data quality problems in literals. Since we are at an early stage of research, we have not yet developed algorithms for all data quality problems from table 1. In the future, we will design additional rules for the identification of comprehensibility problems, redundancy problems, and heterogeneity problems. The identification of certain data quality problems, e.g. false values or outdated values, may likely require additional annotations in the ontology. At present, we have no formal evidence about the scalability of our approach. However, existing commercial databases for RDF data, e.g. Virtuoso from OpenLink Software, contain powerful optimizations for regular expressions and scale well up to at least 8 billion triples. We are planning a more formal evaluation on real-world data sets to prove practical applicability.



Resource Form ⚠

Name: <http://www.example.com/stockdblocation/1>

Annotations

rdfs:label
location #1

Other Properties

vocab:location_CITY
Las Vegas

vocab:location_COUNTRY ⚠
S France

vocab:location_ID ⚠
I 2

vocab:location_STATE
NV

vocab:location_STREET ⚠
8489 Strong St.

vocab:location_STREETNO ⚠

Fig. 3. Identification of data quality problems

6 Conclusion and Outlook on Future Work

The proposed approach provides a set of generally usable query templates that allow the identification of data quality problem types, as known from data quality research, in relational database content lifted to RDF, and to native RDF knowledge bases alike, independently of a specific domain or source system. Therefore, it is theoretically suitable for any Semantic Web data before or after its publication on the Web.

So far, we have developed query templates for the identification of syntax errors, missing values, unique value violations, out of range values, and functional dependency violations. Future work will address the development of additional identification rules for other data quality problems. Moreover, we plan to develop correction heuristics for the automated repair of some of the identified data quality problems. It is also planned to evaluate our approach using large-scale real-world data sets to prove the practical applicability. Additionally, we will soon expand the scope of our approach to multi-source scenarios that will be suitable for data quality management of master data distributed in heterogeneous data sources.

7 References

1. Wang, R. Y., Strong, D. M.: Beyond accuracy: what data quality means to data consumers. Journal of Management Information Systems, 12(4), 5-33 (1996)
2. Redman, T. C.: Data quality: the field guide. Digital Press, Boston (2001)
3. Redman, T. C.: Data quality for the information age. Artech House, Boston (1996)
4. Berners-Lee, T., Hendler, J., and Lassila, O.: The Semantic Web. Scientific American, 284(5), 34-43 (2001)
5. Uschold, M., & Gruninger, M.. Ontologies: Principles, Methods, and Applications. The Knowledge Engineering Review, 11(2), 93-155 (1996)
6. BestBuy catalog in RDF: http://products.semweb.bestbuy.com/sitemap.xml
7. Hepp, M.: GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In: Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Acitrezza, Italy, 332-347 (2008)
8. Oliveira, P., Rodrigues, F., Henriques, P. R.: A Formal Definition of Data Quality Problems. In: International Conference on Information Quality (2005)
9. Leser, U., and Naumann, F.: Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen, dpunkt-Verlag, Heidelberg (2007)
10. Oliveira, P., Rodrigues, F., Henriques, P.R., and Galhardas, H.: A Taxonomy of Data Quality Problems, In: Proc. 2nd Int. Workshop on Data and Information Quality (in conjunction with CAiSE'05), Porto, Portugal (2005)
11. Rahm, E., Do, H.-H.: Data Cleaning: Problems and Current Approaches. IEEE Data Engineering Bulletin 23(4), 3-13 (2000)
12. OpenLink Software: Sponger Technology http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtSponger
13. Olson, J.: Data quality: the accuracy dimension. Morgan Kaufmann Publishers, San Francisco (2003)
14. Wang, X., Hamilton, H. J., Bither, Y.: An ontology-based approach to data cleaning. Regina: Dept. of Computer Science, University of Regina (2005)
15. Grüning, F.: Datenqualitätsmanagement in der Energiewirtschaft. Oldenburger Verlag für Wirtschaft, Informatik und Recht, Oldenburg (2009)
16. Ji, Q., Haase, P., Qi, G., Hitzler, P., & Stadtmüller, S.. RaDON -- Repair and Diagnosis in Ontology Networks. In: 6th European Semantic Web Conference on The Semantic Web: Research and Applications (2009)
17. Knublauch, H.: SPIN – SPARQL Inferencing Notation, http://spinrdf.org/, retrieved on Dec 04th (2009)
18. Alexiev, V., Breu, M., de Bruin, J., Fensel, D., Lara, R., & Lausen, H.: Information integration with ontologies : experiences from an industrial showcase. Jon Wiley & Sons, Ltd., Chichester (2005)
19. Eckerson, W.: Data Quality and the Bottom Line: Achieving Business Success through a Commitment to High Quality Data. Report of The Data Warehousing Institute (2002)
20. Redman, T. C.: The impact of poor data quality on the typical enterprise. Communications of the ACM, 41, 79-82 (1998)
21. Kedad, Z., Métais, E.: Ontology-Based Data Cleaning. In: Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems-Revised Papers (2002)
22. Hartig, O.: Provenance Information in the Web of Data. Linked Data on the Web (LDOW'09) Workshop at the World Wide Web Conference (WWW) (2009)
23. Hartig, O.: Querying Trust in RDF Data with tSPARQL. Paper presented at the 6th Annual European Semantic Web Conference (ESWC2009) (2009)
24. O'Reilly catalog in RDF: http://oreilly.com/catalog/9780596007683