Aspects of RDF Data Constraints in the Social, Behavioural and Economic Sciences

Thomas Bosch[1], Benjamin Zapilko[1], Joachim Wackerow[1], and Kai Eckert[2]

[1] GESIS – Leibniz Institute for the Social Sciences, Germany
{firstname.lastname}@gesis.org,
[2] Stuttgart Media University, Germany
eckert@hdm-stuttgart.de

Abstract. For research institutes, data libraries, and data archives, RDF data validation according to predefined constraints is a much sought-after feature, particularly as this is taken for granted in the XML world. Based on our work in the DCMI RDF Application Profiles Task Group and in cooperation with the W3C Data Shapes Working Group, we identified and published by today 82 types of constraints that are required by various stakeholders for data applications. In this paper, we formulate 246 constraints of 53 different types on six different vocabularies (*Disco*, *QB*, *SKOS*, *PHDD*, *DCAT*, *XKOS*) and classify them according to the complexity level of their type and their severity level. For 114 of these constraints, we evaluate the data quality of 15,694 data sets (4.26 billion triples) of research data for the social, behavioural, and economic (*SBE*) sciences obtained from 33 SPARQL endpoints. Based on the results, we formulate several hypotheses to direct the further development of constraint languages.

Keywords: Constraint Validation, Data Quality, RDF

1 Introduction

The social, behavioural, and economic sciences (*SBE*) require high-quality data for their empirical research. For more than a decade, members of the *SBE* community have been developing and using a metadata standard, composed of almost twelve hundred metadata fields, known as the *Data Documentation Initiative (DDI)*, an XML format to disseminate, manage, and reuse data collected and archived for research [11].

In XML, the definition of schemas containing data constraints and the validation of data according to these constraints is commonly used to ensure a certain level of data quality.

With the rise of the Web of Data, data professionals and institutions are very interested in having their data be discovered and used by publishing their data directly in RDF or at least publish accurate metadata about their data to facilitate data integration. Therefore, not only established vocabularies like *SKOS* are used; recently, members of the *SBE* and Linked Data community developed with the *DDI-RDF Discovery Vocabulary (Disco)*[3] a means to expose *DDI* metadata as Linked Data.

For constraint formulation and validation of RDF data, several languages exist or are currently developed, like *Shape Expressions*, *Resource Shapes* or *Description Set Profiles*. *OWL 2* is also used as a constraint language under a closed world assumption. With its direct support of validation via *SPARQL*, *SPIN* is very popular and certainly plays an important role for future developments in this field. It is particularly interesting as a means to validate arbitrary constraint languages by mapping them to *SPARQL* [2]. Yet, there is no clear favorite and none of the languages is able to meet all requirements raised by data practitioners. Further research and development therefore is needed.

In 2013, the W3C organized the RDF Validation Workshop[4], where experts from industry, government, and academia discussed first use cases for RDF constraint formulation and RDF data validation. In 2014, two working groups on RDF validation have been established to develop a language to express constraints on RDF data: the W3C RDF Data Shapes working group[5] and the DCMI RDF Application Profiles task group[6] which among others bundles the requirements of data institutions of the cultural heritage and *SBE* sector and represents them in the W3C group.

Within the DCMI task group, a collaboratively curated database of RDF validation requirements has been created which contains the findings of the working groups based on various case studies provided by data institutions [1]. It is publicly available and open for further contributions[7].

The database connects requirements to use cases, case studies and implementations and forms the basis of this paper. To gain a better understanding about the role of certain requirements for data quality and in order to direct the further development of constraint languages, we collected constraints for commonly used vocabularies in the *SBE* domain, either from the vocabularies themselves or from domain and data experts. All in all, this lead to 246 constraints of 53 different types on six vocabularies. We let the experts classify the constraints according to the severity if they are violated. Furthermore, we classified the type of each constraint (corresponding to a requirement) based on its complexity ranging from types commonly found in vocabulary specifications (e.g., domain, range, and cardinality restrictions), over types that are simply stated using common constraint languages (e.g., language tag cardinality restrictions) to complex types that involve complex data structures or need more sophisticated languages to be easily expressible (e.g., constraints on graph-based structures).

Thomas: in total 82 constraint types

As we do not want to base our conclusions on the evaluations of vocabularies and constraint definitions alone, we conducted a large-scale experiment and evaluated the data quality of 15,694 data sets (4,26 billion triples) of research

[3] http://rdf-vocabulary.ddialliance.org/discovery.html
[4] http://www.w3.org/2012/12/rdf-val/
[5] http://www.w3.org/2014/rds/charter
[6] http://wiki.dublincore.org/index.php/RDF-Application-Profiles
[7] Online at http://purl.org/net/rdf-validation

data for the social, behavioural, and economic (*SBE*) sciences obtained from 33 SPARQL endpoints.

In this paper, we discuss constraints on RDF data in general. Note that the data represented in RDF can be data in the sense of *SBE* sciences, but also metadata about published or unpublished data. We generally refer to both simply as RDF data and only distinguish between data and metadata in the data set descriptions and in the case that it matters for the purpose of this paper. The remainder of the paper is structured as follows.

1.1 Motivation

The data most often used in research within *SBE* sciences is *person-level data* (or more generally *record-unit data*, i.e., data collected about individuals, businesses, and households) in form of responses to studies or taken from administrative registers (such as hospital records, registers of births and deaths). The range of person-level data is very broad - including census, education, health data and business, social, and labor force surveys. This type of research data is held within data archives or data libraries after it has been collected, so that it may be reused by future researchers.

By its nature, person-level data is highly confidential and access is often only permitted for qualified researchers who must apply for access. Researchers typically represent their results as aggregated data in form of multi-dimensional tables with only a few columns; so-called *variables* such as *sex* or *age*. Aggregated data, which answers particular research questions, is derived from person-level data by statistics on groups or aggregates such as frequencies and arithmetic means. The purpose of publicly available aggregated data is to get a first overview and to gain an interest in further analyses on the underlying person-level data. Aggregated data is published in form of CSV files, allowing to perform calculations on the data.

For more detailed analyses, researchers refer to person-level data including additional variables needed to answer subsequent research questions like the comparison of studies between countries. A *study* represents the process by which a data set was generated or collected. Eurostat[8], the statistical office of the European Union, provides research findings in form of aggregated data (downloadable as CSV files) and its metadata at European level that enable comparisons between countries. The variable *formal childcare*[9] captures the measured availability of childcare services in percent over the population in European Union member states by the variables *year*, *duration* (in hours per week), *age* of the child, and *country*. Variables are constructed out of values (of one or multiple datatypes) and/or code lists. The variable *age*, e.g., may be represented by values of the datatype *xsd:nonNegativeInteger* or by a code list of age clusters (e.g., '0 to 10' and '11 to 20').

[8] http://ec.europa.eu/eurostat

[9] Aggregated data and its metadata is available at: http://ec.europa.eu/eurostat/web/products-datasets/-/ilc_caindformal

A very important and representative RDF validation case study within SBE sciences is the comparison of variables between data collections of different countries. Several vocabulary-specific constraints on RDF data are checked for each data collection to determine if variables measuring *age* - collected for different countries (age_{DE}, age_{UK}) - are comparable: (1) variable definitions must be available, (2) for each code a human-readable label has to be specified, (3) code lists must be structured properly, and (4) code lists must either be identical or at least similar. If a researcher only wants to get a first overview over the comparability of variables (use case 1), covering the first three constraints may be sufficient, i.e., the violation of the first three constraints is more serious than the violation of the last constraint. If the intention of the researcher is to perform more sophisticated comparisons (use case 2), however, the user may raise the severity level of the last constraint.

2 Related Work

For data archives, research institutes, and data libraries, RDF validation according to predefined constraints is a much sought-after feature, particularly as this is taken for granted in the XML world. DDI-XML documents, e.g., are validated against diverse XSDs[10]. As certain constraints cannot be formulated and validated by XSDs, so-called secondary-level validation tools like *Schematron*[11] have been introduced to overcome the limitations of XML validation. *Schematron* generates validation rules and validates XML documents according to them. With RDF validation, one can overcome drawbacks when validating XML documents[12]. It cannot be validated, e.g., if each code of a variable's code list is associated with a category (*R-86*). Additionally, it cannot be validated that if an element has a specific value, then certain child elements must be present (*R-71*). A comprehensive comparison of XML and RDF validation, however, is not within the scope of this paper.

A well-formed *RDF Data Cube* is an a RDF graph describing one or more instances of *qb:DataSet* for which each of the 22 integrity constraints[13], defined within the *QB* specification, passes. Each integrity constraint is expressed as narrative prose and, where possible, a SPARQL ASK query or query template. If the ASK query is applied to an RDF graph then it will return true if that graph contains one or more *QB* instances which violate the corresponding constraint [7]. Mader, Haslhofer, and Isaac investigated how to support taxonomists in improving SKOS vocabularies by pointing out quality issues that go beyond the integrity constraints defined in the SKOS specification [9].

[10] http://www.ddialliance.org/Specification/
[11] https://msdn.microsoft.com/en-us/library/aa468554.aspx
[12] http://www.xmlmind.com/xmleditor/_distrib/doc/xmltool/xsd_structure_limitations.html
[13] http://www.w3.org/TR/vocab-data-cube/#wf

3 Common Vocabularies in SBE Sciences

For our evaluation, we examined six different vocabularies commonly used in or developed for the *SBE* sciences which are briefly introduced in the following. For three of them, we analyzed actual data according to constraint violations. For *PHDD* and *XKOS*, there is not yet enough data available so we limit our examination to the vocabularies themselves.

The *RDF Data Cube Vocabulary (QB)*[14] is a W3C recommendation for representing metadata on *data cubes*, i.e. multi-dimensional aggregated data, in RDF [6]. A *qb:DataStructureDefinition* contains metadata of the data collection. The variable *formal childcare* is modeled as *qb:measure*, since it stands for what has been measured in the data collection. The variables *year*, *duration*, *age*, and *country* are *qb:dimensions*. Data values, i.e., the availability of childcare services in percent over the population, are collected in a *qb:DataSet*. Each data value is represented inside a *qb:Observation* which contains values for each dimension[15].

Physical Data Description (PHDD)[16] is a vocabulary to represent data in tabular format in RDF enabling further aggregations and calculations. The data could be either represented in records with character-separated values (CSV) or fixed length. *Eurostat* provides a CSV file, a two-dimensional table (*phdd:Table*) about *formal childcare* which is structured by a table structure (*phdd:TableStructure*, *phdd:Delimited*) including information about the character set (*ASCII*), the variable delimiter (*,*), the new line marker (*CRLF*), and the first line where the data starts (*2*). The table structure is related to table columns (*phdd:Column*) which are described by column descriptions (*phdd:DelimitedColumnDescription*). For the column containing the cell values in percent, the column position (*5*), the recommended data type (*xsd:nonNegativeInteger*), and the storage format (*TINYINT*) is stated.

For more detailed analyses we refer to the metadata on person-level data collected for the series *EU-SILC (European Union Statistics on Income and Living Conditions)*[17]. Where data collection is cyclic, data sets may be released as *series*, where each cycle produces one or more data sets. The aggregated variable *formal childcare* is calculated on the basis of six person-level variables (e.g., *Education at pre-school*[18]) for which detailed metadata is given (e.g., code lists) enabling researchers to replicate the results shown in aggregated data tables. The *DDI-RDF Discovery Vocabulary (Disco)* is a vocabulary to represent metadata on person-level data in RDF. The series (*disco:StudyGroup*) *EU-SILC* contains one study (*disco:Study*) for each year (*dcterms:temporal*) of data collection. *dcterms:spatial* points to the countries for which the data has been collected. The study *EU-SILC 2011* contains eight person-level data sets (*disco:LogicalDataSet*)

---

[14] http://www.w3.org/TR/vocab-data-cube/

[15] The complete running example in RDF is available at: https://github.com/boschthomas/rdf-validation/tree/master/data/running-example

[16] https://github.com/linked-statistics/physical-data-description

[17] http://www.gesis.org/missy/eu/metadata/EU-SILC

[18] http://www.gesis.org/missy/eu/metadata/EU-SILC/2011/Cross-sectional/original#2011-Cross-sectional-RL010

including person-level variables (*disco:Variable*) like the six ones needed to calculate the aggregated variable *formal childcare*.

The *Simple Knowledge Organization System (SKOS)* is reused multiple times to build *SBE* vocabularies. The codes of the variable *Education at pre-school* (number of education hours per week) are modeled as *skos:Concepts* and a *skos:OrderedCollection* organizes them in a particular order within a *skos:memberList*. A variable may be associated with a theoretical concept (*skos:Concept*). *skos:narrower* builds the hierarchy of theoretical concepts within a *skos:ConceptScheme* of a series. The variable *Education at pre-school* is assigned to the theoretical concept *Child Care* which is the narrower concept of *Education* - one of the top concepts of the series *EU-SILC*. Controlled vocabularies (*skos:ConceptScheme*), serving as extension and reuse mechanism, organize types (*skos:Concept*) of descriptive statistics (*disco:SummaryStatistics*) like minimum, maximum, and arithmetic mean. *XKOS*[19] is a SKOS extension to describe formal statistical classifications like the International Standard Classification of Occupations (*ISCO*). and the Statistical Classification of Economic Activities in the European Community *NACE*. *DCAT* enables to represent data sets inside of data collections like portals, repositories, catalogs, and archives which serve as typical entry points when searching for data.

4 Classification of Constraint Types and Constraints

Bosch et al. identified 82 requirements to formulate RDF constraints (e.g. *R-75: minimum qualified cardinality restrictions*); each of them corresponding to an RDF constraint type[20][3]. We published a technical report[21] (serving as first appendix of this paper) in which we explain each requirement/constraint type in detail and give examples for each expressed by different constraint languages. The knowledge representation formalism *Description logics (DL)*, with its well-studied theoretical properties, provides the foundational basis for each constraint type. Therefore, this technical report contains mappings to *DL* to logically underpin each requirement and to determine which *DL* constructs are needed to express each constraint type [3]. We classified both RDF constraint types and RDF constraints to gain better insights into the quality of RDF data with respect to this classification, independent of the used vocabulary. We recently published a technical report[22] (serving as second appendix of this paper) in which we describe 246 constraints of 53 distinct constraint types on six vocabularies [5].

4.1 Classification of RDF Constraint Types

According to the complexity of constraint types, the complete set of *constraint types* encompasses three disjoint *sets of constraint types*:

[19] https://github.com/linked-statistics/xkos

[20] Constraint types and constraints are uniquely identified by alphanumeric technical identifiers like *R-71-CONDITIONAL-PROPERTIES*

[21] Available at: http://arxiv.org/abs/1501.03933

[22] Available at: http://arxiv.org/abs/1504.04479

1. ***Vocabulary Constraint Types***
2. ***Simple Constraint Types***
3. ***Complex Constraint Types***

The modeling languages *RDF*, *RDFS*, and *OWL* are typically used to formally specify vocabularies. *Vocabulary constraint types* denotes the set of constraint types whose constraints can be extracted completely automatically out of formal specifications of vocabularies. As vocabularies have been specified using *RDF*, *RDFS*, and *OWL*, *vocabulary constraints* ensure that the data is consistent with the intended syntaxes, semantics, and integrity of vocabularies' data models. *Minimum qualified cardinality restrictions* (*R-74*), e.g., guarantee that individuals of given classes are connected by particular properties to at least n different individuals/literals of certain classes or data ranges. In *PHDD*, a *minimum qualified cardinality restriction* can be obtained from the OWL definition of this restriction class, ensuring that a *phdd:TableStructure* has (*phdd:column*) at least one *phdd:Column*:

```
[   a owl:Restriction ; rdfs:subClassOf TableStructure ;
    owl:minQualifiedCardinality 1 ;
    owl:onProperty column ;
    owl:onClass Column ] .
```

Simple and *complex constraints* are in contrast to *vocabulary constraints* not explicitly defined within formal specifications of vocabularies. *Simple* and *complex constraints* are defined according to textual descriptions of the intended semantics of vocabularies. *Simple constraint types* is the set of constraint types whose constraints can be easily defined without much effort in addition to the already defined *vocabulary constraints*. *Data property facets* (*R-46*) is an example of a *simple constraint type* which enables to declare frequently needed facets for data properties in order to validate input against simple conditions including min/max values, regular expressions, and string length. The abstract of series/studies, e.g., should have a minimum length which is easily and concisely expressible by SPARQL:

```
SELECT ?study WHERE {
    ?study dcterms:abstract ?abstract . FILTER ( STRLEN ( ?abstract ) = 10 ) . }
```

Complex constraint types encompass constraint types for which the definition of constraints is rather complex and cannot be derived from vocabulary definitions. For assessing the quality of thesauri, e.g., we concentrate on the graph-based structure and apply graph- and network-analysis techniques. An example of such constraints of the constraint type *structure* is that a thesaurus should not contain many orphan concepts, i.e., concepts without any associative or hierarchical relations, lacking context information valuable for search. This *complex constraint* is only expressible by *SPARQL* and not directly understandable:

```
SELECT ?concept WHERE {
    ?concept a [rdfs:subClassOf* skos:Concept] .
```

```
FILTER NOT EXISTS { ?concept ?p ?o .
    FILTER ( ?p IN ( skos:related, skos:relatedMatch, skos:broader, ... ) ) . }
```

Complex constraints are in most cases only expressible by plain *SPARQL* which shows the importance to develop suitable constraint languages.

4.2 Classification of RDF Constraints

A concrete constraint is instantiated from one of the 82 constraint types and is defined for a specific vocabulary. It does not make sense to determine the severity of constraint violations of an entire constraint type, as the severity of the violation of a given constraint depends on the individual context and vocabulary. *SBE* experts determined the default ***severity level*** (*R-158*) for each of the 246 constraints to indicate how serious the violation of the constraint is. We use the commonly accepted classification of log messages in software development and distinguish *informational*, *warning* and *error*. Violations of *informational constraints* point to desirable but not necessary data improvements to achieve RDF representations which are ideal in terms of syntax and semantics of used vocabularies. *Errors* are syntactic or semantic errors which should cause the abortion of data processing. *Warnings* are syntactic or semantic errors which typically should not lead to an abortion of data processing. As the purpose of *vocabulary constraints* is to ensure explicitly stated semantics of vocabularies, their default severity levels are in most cases very strong (*error*) and in average stronger than the severity levels of *simple* and *complex constraints*. As a consequence, violating many *vocabulary constraints* is an indicator for bad (meta)data quality.

Although, we provide default severity levels for each constraint, validation environments should enable users to adapt the severity levels of constraints according to their individual needs. Validation environments should enable users to select which constraints to validate against depending on their individual use cases. For some use cases, validating *vocabulary constraints* may be more important than validating *simple* or *complex constraints*. For other use cases, validating *error constraints* may be sufficient without taking *warning* and *informational constraints* into account. We evaluated the (meta)data quality of large real world data sets represented by multiple and different vocabularies to get an understanding (1) which sets of constraint types (on different levels of complexity) and (2) which sets of constraints (associated with particular severity levels) encompass the constraints causing the most/fewest constraint violations (see section 5).

4.3 Vocabulary Constraint Types

It is a common requirement to narrow down the value space of properties by an exhaustive enumeration of valid values (*R-30/37: allowed values*). *disco:Category Statistics*, e.g., can only have *disco:computationBase* relationships to the values *valid* and *invalid* of the datatype *rdf:langString*. Consider the following *DL* knowledge base $\mathcal{K}$[23]:

$$\mathcal{K} = \{\ \texttt{CategoryStatistics} \equiv \forall\ \texttt{computationBase.}\{\texttt{valid,invalid}\} \sqcap \texttt{langString},$$
$$\texttt{Variable} \equiv \exists\ \texttt{concept.Concept},\ \texttt{Catalog} \sqsubseteq \forall\ \texttt{dataset.Dataset},$$
$$\exists\ \texttt{isStructuredBy.}\top \sqsubseteq \texttt{Table},\ \top \sqsubseteq \forall\ \texttt{belongsTo.Concept}\ \}$$

Existential quantifications (*R-86*) enforce that instances of given classes must have some property relation to individuals/literals of certain types. Variables, e.g., should have a relation to a theoretical concept (*informational*). The variable *Education at pre-school* is associated with the theoretical concept *Child Care*. The default severity level of this constraint is weak, as in most cases research can be continued without having information about the theoretical concept of a variable. A *universal quantification* (*R-91*) contains all those individuals that are connected by a property only to individuals/literals of particular classes or data ranges. Only *dcat:Catalogs*, e.g., can have *dcat:dataset* relationships to *dcat:Datasets*. *Property domain* (*R-25, R-26*) and *range* (*R-28, R-35*) constraints restrict domains and ranges of properties: Only *phdd:Tables*, e.g., can have *phdd:isStructuredBy* relationships and *xkos:belongsTo* relations can only point to *skos:Concepts*.

4.4 Simple Constraint Types

For data properties, it may be desirable to restrict that values of predefined languages must be present for determined number of times (*R-48/49: language tag cardinality*): (1) It is checked if literal language tags are set. Some controlled vocabularies, e.g., contain literals in natural language, but without information what language has actually been used (*warning*). (2) Language tags must conform to language standards (*error*). (3) Some thesaurus concepts are labeled in only one, others in multiple languages. It may be desirable to have each concept labeled in each of the languages that are also used on the other concepts, as language coverage incompleteness for some concepts may indicate shortcomings of thesauri (*informational*) [9]. Percentage values are only valid when they are within the literal range of 0 and 100 (*R-45: literal ranges*; *error*) which is checked for *disco:percentage* standing for the number of cases of a given code in relation to the total number of cases for a particular variable.

It is often useful to declare a given (data) property as the *primary key* (*R-226*) of a class, so that a system can enforce uniqueness and build URIs from user inputs and imported data. In *Disco*, resources are uniquely identified by the property *adms:identifier*, which is therefore inverse-functional ($\texttt{funct identifier}^{-}$), i.e., for each *rdfs:Resource* x, there can be at most one distinct resource y such that y is connected by *adms:identifier*$^{-}$ to x (*error*). Keys, however, are even more general than *inverse-functional properties* (*R-58*), as a key can be a data

[23] A knowledge base is a collection of formal statements which corresponds to *facts* or what is known explicitly. For simplicity reasons, we do not state namespace prefixes.

property, an object property, or a chain of properties [10]. Thus and as there are different sorts of key, and as keys can lead to undecidability, *DL* is extended with the construct *keyfor* (`identifier keyfor Resource`) [8] which is implemented by the *OWL 2 hasKey* construct.

4.5 Complex Constraint Types

Data model consistency constraints ensure the integrity of the data according to the intended semantics of vocabularies. Every *qb:Observation*, e.g., must have a value for each dimension declared in its *qb:DataStructureDefinition* (*error*) and no two *qb:Observations* in the same *qb:DataSet* can have the same value for all dimensions (*warning*). If a *qb:DataSet D* has a *qb:Slice S*, and *S* has an *qb:Observation O*, then the *qb:DataSet* corresponding to *O* must be *D* (*warning*). In many cases, resources must be *members of controlled vocabularies* (*R-32*). If a dimension property, e.g., has a *qb:codeList*, then the value of the dimension property on every *qb:Observation* must be in the code list (*error*). Objects/literals can be declared to be ordered for given properties (*R-121/217: ordering*). Variables, questions, and codes, e.g., are typically organized in a particular order. If codes (*skos:Concept*) should be ordered, they must be members (*skos:memberList*) in an ordered collection (*skos:OrderedCollection*), the variable's code list (*informational*).

It is useful to declare properties to be *conditional* (*R-71*), i.e., if particular properties exist (or do not exist), then other properties must also be present (or absent). To get an overview over a series/study either an abstract, a title, an alternative title, or links to external descriptions should be provided. If an abstract and an external description are absent, however, a title or an alternative title should be given (*warning*). In case a variable is represented in form of a code list, codes may be associated with categories, i.e., human-readable labels (*informational*). The variable *Education at pre-school*, e.g., is represented as ordered code list without any categories. *Context-specific exclusive or of property groups* (*R-11*) constraints restrict individuals of given classes to have properties defined within exactly one of multiple property groups. *skos:Concepts* can have either *skos:definition* (when interpreted as theoretical concepts) or *skos:notation* and *skos:prefLabel* properties (when interpreted as codes), but not both (*error*).

5 Evaluation

5.1 Experimental Setup

In close collaboration with several *SBE* domain experts, we defined 246 constraint of 53 different types on six vocabularies (*Disco*, *QB*, *SKOS*, *PHDD*, *DCAT*, *XKOS*) and classified them according to the complexity level of their type and their severity level. For 114 of these constraints[24], we evaluated the data quality of 15,694 data sets (4.26 billion triples) of *SBE* research data on three

[24] https://github.com/boschthomas/rdf-validation/tree/master/constraints

common vocabularies in *SBE* sciences (*Disco*, *QB*, *SKOS*) obtained from 33 SPARQL endpoints. We distinct two classes of vocabularies: (1) well-established vocabularies (e.g., *QB*, *SKOS*) which are widely adopted and accepted and (2) newly developed vocabularies (e.g., *Disco* which will be published in 2015) which are either recently published or are still in the publication process.

We validated 9,990 / 3,775,983,610 (*QB*), 4,178 / 477,737,281 (*SKOS*), and 1,526 / 9,673,055 (*Disco*) data sets / triples using our validation environment in batch mode. We validated, i.a., (1) *QB* data sets published by the *Australian Bureau of Statistics*, the *European Central Bank*, and the *Organisation for Economic Co-operation and Development*, (2) *SKOS* thesauri like the *AGROVOC Multilingual agricultural thesaurus*, the *STW Thesaurus for Economics*, and the *Thesaurus for the Social Sciences*, and (3) *Disco* data sets provided by the *Microdata Information System*, the *Data Without Boundaries Discovery Portal*, the *Danish Data Archive*, and the *Swedish National Data Service*. We recently published a technical report[25] (serving as third appendix of this paper) in which we describe the evaluation in detail [4]. As we evaluated nearly 10 thousand *QB* data sets, we published the evaluation results for each data set in form of one document per SPARQL endpoint[26].

As *SPARQL* is generally seen as the method of choice to validate RDF data according to certain constraints, we use *SPIN*, a *SPARQL*-based way to formulate and check constraints, as basis to develop a validation environment (available at http://purl.org/net/rdfval-demo)[27] to validate RDF data according to constraints expressed my arbitrary constraint languages[28] [2]. The *RDF Validator* also validates RDF data according to constraints defined for vocabularies such as *Disco*, *QB*, *SKOS*, *PHDD*, *DCAT*, and *XKOS*. The *SPIN* engine checks for each resource if it satisfies all constraints, which are associated with its assigned classes, and generates a result RDF graph containing information about all constraint violations. There is one SPIN construct template for each constraint type[29]. A *SPIN* construct template contains a *SPARQL CONSTRUCT* query which generates constraint violation triples indicating the subject and the properties causing constraint violations, and the reason why constraint violations have been raised. A *SPIN* construct template creates constraint violation triples if all triple patterns within the *SPARQL WHERE clause* match.

5.2 Evaluation Results and Formulation of Hypotheses

Table 1 shows the results of the evaluation, more specifically the constraints and the constraint violations, which are caused by these constraints, in percent. The

[25] Available at: http://arxiv.org/abs/1504.04478
[26] Available at: https://github.com/boschthomas/rdf-validation/tree/master/evaluation/data-sets/data-cube
[27] Source code downloadable at: https://github.com/boschthomas/rdf-validator
[28] SPIN mappings available at: https://github.com/boschthomas/rdf-validation/tree/master/SPIN
[29] For a comprehensive description of the *RDF Validator*, we refer to [2]

constraints and their raised constraint violations are grouped by vocabulary, complexity level of their type, and their severity level. The number of evaluated triples and data sets differs between the vocabularies as we evaluated 3.8 billion *QB*, 480 million *SKOS*, and 10 million *Disco* triples. To be able to formulate hypotheses which apply for all vocabularies, we only use normalized relative values representing the percentage of constraints and constraint violations belonging to the respective class

	Disco		***QB***		***SKOS***		***Total***	
	C	CV	C	CV	C	CV	C	CV
	143	3,575,002	35	45,635,861	35	5,540,988	213	54,751,851
complex	25.9%	18.3%	37.1%	**100.0%**	**37.1%**	21.4%	33.4%	**46.6%**
simple	19.6%	15.7%	8.6%	0.0%	**34.3%**	**78.6%**	20.8%	31.4%
vocabulary	**54.6%**	**66.1%**	**54.3%**	0.0%	**28.6%**	0.0%	**45.8%**	22.0%
info	**52.5%**	**52.6%**	11.4%	0.0%	**60.0%**	41.2%	**41.3%**	31.3%
warning	7.0%	29.4%	8.6%	**99.8%**	14.3%	**58.8%**	10.0%	**62.7%**
error	40.6%	18.0%	**80.0%**	0.3%	25.7%	0.0%	**48.8%**	6.1%

C (constraints), CV (constraint violations)

Table 1: Constraints and constraint violations

As the evaluation is based on three vocabularies, we cannot make valid general statements for all vocabularies, but we can formulate several hypotheses to direct the further development of constraint languages. As these hypotheses cannot be proved yet, they still have to be verified or falsified by evaluating the quality of data represented by further well-established and newly developed vocabularies. Almost 1/2 of all 213 constraints and more than 1/2 of the *Disco* and the *QB* constraints are *vocabulary constraints*. The *SKOS* constraints are nearly to the same extend *vocabulary*, *simple*, and *complex constraints*.

Hypothesis 1 *As a significant amount of 54% of the constraints are either simple or complex constraints, the further development of constraint languages should concentrate on expressing simple and especially complex constraints which up to now in most cases can only be expressed by plain SPARQL.*

Nearly 1/2 of all violations are caused by *complex constraints*, 1/3 by *simple constraints*, and 1/5 by *vocabulary constraints*. The fact that only 1/5 of all violations result from *vocabulary constraints*, even though, 46% of all constraints are *vocabulary constraints*, indicates good data quality for all vocabularies with regard to their formal specifications.

Hypothesis 2 *For all vocabularies, data corresponds to their formal specifications which demonstrates that constraint formulation in general works.*

2/3 of the *Disco* violations result from *vocabulary constraints*, *QB* violations are almost only raised by *complex constraints*, and nearly 80% of the *SKOS* violations are caused by *simple constraints*. For well-established vocabularies, *vocabulary constraints* are almost completely satisfied[30] which indicates good data quality according to their formal specifications. For newly defined vocabularies, however, 2/3 of all violations are raised by *vocabulary constraints* which indicates bad data quality with regard to their formal specifications.

Hypothesis 3 *Data represented by well-established vocabularies corresponds to their formal specifications which demonstrates that constraint formulation in general works. Data represented by newly developed vocabularies, in contrast, does not correspond to their formal specifications.*

It is likely that a newly developed vocabulary is still subject of constant change and that early adopters did not properly understand its formal specification. Thus, published data may not be consistent with its current draft or version. In case newly developed vocabularies turn into well-established ones, data providers are experienced in publishing their data in conformance with these vocabularies and formal specifications are more elaborated. As a consequence, *vocabulary constraints* are satisfied to a greater extend which leads to better data quality.

Hypothesis 4 *A significant amount of 47% of the violations refer to complex constraints that are not easily, concisely, and intuitively expressible by existing constraint languages which confirms the necessity to provide suitable constraint languages.*

In general, *vocabulary constraints* are expressible by either modeling languages (e.g., *RDF*, *RDFS*, *OWL 2*) or constraint languages. *Simple constraints* and especially *complex constraints*, however, are in most cases only expressible by plain *SPARQL*. As almost 1/2 of all violations are caused by *complex constraints*, data quality can be significantly improved when suitable constraint languages are developed which enable to define *complex constraints* in an easy, concise, and intuitive way. 1/2 of all constraints are *error constraints* and 10% of all constraints are *warning constraints*. *Informational constraints* caused 1/3 and *warning constraints* 2/3 of all violations. As the percentage of severe violations is very low for all vocabularies (6.1%), data quality is high with regard to the severity level of constraints.

Hypothesis 5 *The percentage of severe violations is very low, compared to about 2/3 of warning violations and 1/3 of informational violations, which implies that proper constraint languages can significantly improve the data quality beyond fundamental requirements.*

80% of the *QB* constraints are *error constraints*. More than 50% of the *Disco* and *SKOS* constraints, however, are *informational constraints*. 1/6 of the Disco violations are caused by *error constraints* and almost all *QB* violations and 59% of the *SKOS* violations are caused by *warning constraints*. For well-established vocabularies, data quality is high as serious violations rarely appear. For newly developed vocabularies, data quality is worse as serious violations occur partially.

[30] e.g. only 1.777 *QB* violations

Hypothesis 6 *Especially for newly developed vocabularies, constraint languages should be used to a larger extend to meet severe constraints.*

Table 2 shows the relation between the complexity and the severity level of all 213 constraints. More than 1/2 of the *vocabulary constraints* are *error constraints*, more than 3/4 of the *simple constraints* are *informational constraints*, and *complex constraints* are to the same extend *informational* or *error constraints*.

	vocabulary	***simple***	***complex***
info	38.7 %	**76.2 %**	**42.3 %**
warning	6.7 %	7.1 %	13.5 %
error	**54.6 %**	16.7 %	**44.2 %**

Table 2: Relation between complexity and severity level of constraints

Hypothesis 7 *The further development of constraint languages should concentrate on how to express complex constraints as 44% of all complex constraints are also serious ones.*

As *vocabulary constraints* are part of formal specifications of vocabularies, violations of *vocabulary constraints* are more severe than violations caused by *simple constraints*. As *SPARQL* is still used to express *complex constraints* and as 44% of all *complex constraints* are serious, there is an obvious need to develop constraint languages which enable to express *complex constraints*.

6 Conclusion and Future Work

We identified and published by today 82 types of constraints[31] that are required by various stakeholders for data applications. In close collaboration with several *SBE* domain experts, we formulated 246 constraints[32] of 53 different types on six different vocabularies (*Disco*, *QB*, *SKOS*, *PHDD*, *DCAT*, *XKOS*) and classified them according to the complexity level of their type and their severity level.

For 114 of these constraints, we evaluated[33] the data quality of 15,694 data sets (4.26 billion triples) of research data for the *SBE* sciences obtained from 33 SPARQL endpoints. Based on the evaluation results, we formulated several hypotheses to direct the further development of constraint languages. To make

[31] The first appendix of this paper, in which we explain each requirement/constraint type in detail, is available at: http://arxiv.org/abs/1501.03933 [3]

[32] The second appendix of this paper, in which we describe each vocabulary-specific constraint in detail, is available at: http://arxiv.org/abs/1504.04479 [5]

[33] The third appendix of this paper, in which we describe the evaluation in detail, is available at: http://arxiv.org/abs/1504.04478 [4].

valid general statements for all vocabularies, however, the hypotheses still have to be verified or falsified by evaluating the quality of data represented by more than three vocabularies. The main hypotheses are: (1) The percentage of severe violations is very low which implies that proper constraint languages can significantly improve the data quality beyond fundamental requirements. (2) The further development of constraint languages should concentrate on how to express *complex constraints* as 44% of all *complex constraints* are also serious ones. (3) 47% of the violations refer to complex constraints that are in most cases not expressible by existing constraint languages which confirms the necessity to provide suitable constraint languages. (4) As 54% of the constraints are either *simple* or *complex constraints*, the further development of constraint languages should concentrate on expressing *simple* and especially *complex constraints* which up to now in most cases can only be expressed by plain *SPARQL*.

References

1. Thomas Bosch and Kai Eckert. Requirements on rdf constraint formulation and validation. *Proceedings of the DCMI International Conference on Dublin Core and Metadata Applications (DC 2014)*, 2014.
2. Thomas Bosch and Kai Eckert. Towards description set profiles for rdf using sparql as intermediate language. *Proceedings of the DCMI International Conference on Dublin Core and Metadata Applications (DC 2014)*, 2014.
3. Thomas Bosch, Andreas Nolle, Erman Acar, and Kai Eckert. Rdf validation requirements - evaluation and logical underpinning. 2015.
4. Thomas Bosch, Benjamin Zapilko, Joachim Wackerow, and Kai Eckert. An evaluation of metadata and data quality on person-level, aggregated, thesauri, statistical classifications, and rectangular data sets. 2015.
5. Thomas Bosch, Benjamin Zapilko, Joachim Wackerow, and Kai Eckert. Rdf constraints to validate rectangular data and metadata on person-level data, aggregated data, thesauri, and statistical classifications. 2015.
6. Richard Cyganiak, Simon Field, Arofan Gregory, Wolfgang Halb, and Jeni Tennison. Semantic statistics: Bringing together sdmx and scovo. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *Proceedings of the WWW 2010 Workshop on Linked Data on the Web*, volume 628 of *CEUR Workshop Proceedings*, 2010.
7. Richard Cyganiak and Dave Reynolds. The rdf data cube vocabulary. W3C recommendation, W3C, January 2014.
8. Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research*, 23(1):667–726, June 2005.
9. Christian Mader, Bernhard Haslhofer, and Antoine Isaac. Finding quality issues in skos vocabularies. In *Proceedings of the Second International Conference on Theory and Practice of Digital Libraries*, TPDL'12, pages 222–233, Berlin, Heidelberg, 2012. Springer-Verlag.
10. Michael Schneider. OWL 2 Web Ontology Language RDF-Based Semantics. W3C recommendation, W3C, October 2009.
11. Mary Vardigan, Pascal Heus, and Wendy Thomas. Data documentation initiative: Towards a standard for the social sciences. *International Journal of Digital Curation*, 3(1):107–113, 2008.