Quality Assurance of RDB2RDF Mappings

Patrick Westphal, Claus Stadler, and Jens Lehmann

Department of Computer Science, University of Leipzig
Augustusplatz 10
04109 Leipzig
{pwestphal|cstadler|lehmann}@informatik.uni-leipzig.de

Abstract

Today, the *Web of Data* evolved to a semantic information network containing large amounts of data. Since such data may stem from different sources, ranging from automatic extraction processes to extensively curated knowledge bases, its quality also varies. Thus, currently research efforts are made to find methodologies and approaches to measure the data quality in the Web of Data. Besides the option to consider the actual *data* in a quality assessment, taking the *process* of data generation into account is another possibility, especially for extracted data. An extraction approach that gained popularity in the last years is the mapping of relational databases to RDF (RDB2RDF). By providing definitions of how RDF should be generated from relational database content, huge amounts of data can be extracted automatically. Unfortunately, this also means that single errors in the mapping definitions can affect a considerable portion of the generated data. Thus, from a quality assurance point of view, the assessment of these RDB2RDF mapping definitions is important to guarantee high quality RDF data. This is not covered by recent quality research attempts in depth and is examined in this report. After a structured evaluation of existing approaches, a quality assessment methodology and quality dimensions of importance for RDB2RDF mappings are proposed. The formalization of this methodology is used to define 43 metrics to characterize the quality of an RDB2RDF mapping project. These metrics are also implemented for a software prototype of the proposed methodology, which is used in a practical evaluation of three different datasets that are generated applying the RDB2RDF approach.

Table of Contents

1 Introduction

Today, more than 20 years after Tim Berners-Lee first published his ideas of a 'linked information system' [1], this vision of an *information web* evolved into a mature medium for information access, communication, entertainment and commerce. Moreover this *World Wide Web (WWW)* today is the *major* medium for all kinds of information exchange. Initially, this information network mainly grounded on the idea of *hypertext documents* that allow the linking to all kinds of related information, possibly other hypertext documents. This *Web of Documents* is currently being extended to also serve as a *Web of Data*. Inside, data is provided and stored using so called *Semantic Web technologies*, which not only allow a database-style access but further come with linking and inference capabilities which make this web a *Semantic Web*.

Besides the technological foundations, bootstrapping such a linked data network requires data. Even though a vast amount of datasets is already part of such networks, most of the data nowadays is stored in relational database systems [2,3], of which only a few also provide means for data access via Semantic Web technologies. Since the logical foundations of relational databases and Semantic Web data endpoints are comparable with regards to the underlying semantics [4], it seems natural to build converters that are able to transform the one into the other. This was also considered by Tim Berners-Lee back in 1998 [5] and a working group was founded under the umbrella of the *World Wide Web Consortium*[1] to standardize languages and approaches to map relational databases and its schemas to the *Resource Description Framework (RDF)* [6] – the predominant Semantic Web data model. One such language is the *RDB to RDF Mapping Language (R2RML)* [7] providing means to define mappings between relational data and data expressed in RDF. This mapping process is in the following referred to as *RDB2RDF* mapping. Apart from R2RML, there are further languages that can be used to define RDB2RDF mappings. One example is the *Sparqlification Mapping Language (SML)*[2], designed to be easy to read and write by human beings, and which is convertible to R2RML and vice versa.

To make this Web of Data *valuable* and *usable*, not only the data generation and preparation needs consideration, but also its quality evaluation and evolution [8]. Being discussed in the context of many different domains within the last decades, the *data* or *information quality* assessment became a current research topic for Semantic Web data. Since it may stem from many different sources, ranging from crowdsourced user input or automatic extraction processes to extensively curated knowledge bases, the consideration of data quality in the Semantic Web context is of importance [9]. Accordingly, common and generally accepted perceptions, like viewing data quality as a multi-dimensional concept expressing the data's *'fitness for use'* [10], were applied to the Semantic Web context to derive relevant quality aspects.

This report focusses on the combination of the two introduced fields, i.e. how the findings of current data quality research can be applied and extended to be used in the RDB2RDF context.

[1] http://www.w3.org/2001/sw/rdb2rdf/
[2] http://sml.aksw.org

1.1 Motivation

The mapping of relational data to RDF is an ongoing topic which led to recent standardizations [7,11] as well as commercial, free-to-use and free software tools (cf. Section 2.1). Currently there are many prominent RDF datasets that contribute to the Web of Data, which were generated applying RDB2RDF mechanisms. Examples are *LinkedGeoData*[3], a Linked Data mirror of the *OpenStreetMap*[4] project, the RDF version of the data provided by the *PanLex*[5] project, and *LinkedBrainz*[6], which was mapped to RDF from the *MusicBrainz*[7] database. But although a considerable amount of RDF data is generated from relational databases, quality considerations of RDB2RDF mappings are, to the best of our knowledge, not discussed extensively, yet. Even though general Semantic Web-related quality assessment methodologies and tools could be applied to the data generated by RDB2RDF mappings, such attempts would not take the actual data source and the transformation process into account. Accordingly, there might be errors introduced during the data conversion, that are not detectable by just assessing the RDF output. Moreover, since the mapping of relational data to RDF is a mass generation approach [12], a single mapping error might have a great impact on the resulting data. Thus, it is of crucial importance to develop theoretical concepts and means to make such errors detectable, and the overall quality of RDB2RDF mappings assessable.

1.2 Goal

In this report the RDB2RDF mapping process *and* the resulting data and schemas are to be examined from a data quality point of view, deriving criteria for high quality RDB2RDF mappings. Based on a structured evaluation of existing approaches an assessment methodology shall be developed that suits the RDB2RDF process, considering characteristics of the input data, the actual mapping configuration and the generated output. Using a formalization of the proposed methodology, actual metrics are to be defined. Besides this, a software prototype shall be developed, implementing this assessment methodology and the proposed metrics for the Sparqlify RDB2RDF mapping tool. The prototype shall further be used to run data quality assessments on real world RDB2RDF mapping projects to detect actual quality errors and get some initial feedback on which deficiencies are likely to occur in RDB2RDF settings.

1.3 Structure of this Report

First, Section 2 describes the state of the art with respect to RDB2RDF mapping and data quality. These findings are used in Section 3 to develop an approach to derive quality aspects of significance for RDB2RDF mappings and a methodology to assess them, as well as actual metrics. In Section 4 the assessment tool is introduced, followed by an evaluation of assessment runs on different RDB2RDF mapping projects in Section 5. The conclusions are drawn in Section 6.

[3] `http://linkedgeodata.org`
[4] `http://www.openstreetmap.org`
[5] `http://panlex.org`
[6] `http://linkedbrainz.org`
[7] `http://musicbrainz.org`

1.4 Conventions

In this report URIs are represented by their qualified names, if possible, for brevity and to ease the reading. The prefixes used, are given in Table 1.

Prefix	Namespace
dbr	http://dbpedia.org/resource/
dcterms	http://purl.org/dc/terms/
dc	http://purl.org/dc/elements/1.1/
ex	http://ex.org/
foaf	http://xmlns.com/foaf/0.1/
geodata	http://sws.geonames.org/
owl	http://www.w3.org/2002/07/owl#
rdfs	http://www.w3.org/2000/01/rdf-schema#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rr	http://www.w3.org/ns/r2rml#
sioc	http://rdfs.org/sioc/ns#
void	http://rdfs.org/ns/void#
xsd	http://www.w3.org/2001/XMLSchema#

Table 1: Prefix definitions of qualified names used in this report

2 State of the Art

This section presents the current state of the art of the two major topics of this report: RDB2RDF mappings and data quality. First, the main ideas, standards and tools of the RDB2RDF approach are introduced. In the second part of this section, different models and definitions for data quality are described. Afterwards, methodological aspects of the process of determining the quality of data are considered.

2.1 RDB2RDF

Since nowadays most of the data stored digitally resides in relational databases [12], they are an important data source for the Web of Data [13]. To utilize such data, one can distinguish between three different approaches. First, a snapshot of the relational data can be converted to the RDF data model and then made available via SPARQL or as Linked Data. The advantage of this Extract Transform Load (ETL) approach is that the RDF data is accessible in a certain serialization format which allows further processing. Nonetheless, this approach is not optimal in cases where the database to convert is frequently updated since the conversion has to be performed repeatedly to stay up-to-date.

A second approach applies an on-the-fly conversion, using a service that translates SPARQL queries against a virtual RDF graph to SQL queries against the considered database. Therefore a special configuration is needed that defines, how the virtual RDF graph is derived from the underlying relational data, or conversely, what the basic specifications for the query translations are. Such a service can deliver current data but requires some overhead for every query to translate.

Finally, these two techniques can be combined. In this case, the converting software may pre-process the SPARQL query, generate SQL queries that return intermediate results to derive the final result from. However, this approach may require quite a lot of main memory to hold the intermediate results.

All these techniques are referred to as *relational database to RDF mapping* (abbreviated *RDB to RDF* or *RDB2RDF* mapping) approaches [14]. Apart from the differences concerning the point in time when to perform the conversion and on which portion of data, they all have in common, that the conversion rules are applied to single tuples retrieved via an SQL query [12]. Thus, the RDB2RDF quality considerations hold for all of the introduced approaches.

These approaches can be applied following different strategies and using different languages to express the transformation rules. The two basic strategies – retrieving the mapping configuration automatically and defining them by hand – are introduced in the following. To support the manual mapping definitions, several languages evolved. Two of them – the *RDB to RDF Mapping Language* and the *Sparqlification Mapping Language* – are also covered in the next section. Afterwards, current RDB2RDF implementations are considered.

RDB2RDF Strategies and Mapping Languages

Relational Database	RDF
table	class
row	resource typed with its table class
column	property
column value	literal value assigned via column property
foreign key relation	column property considered as object property; object is the resource representing the target row

Table 2: Example translation patterns of the direct mapping approach

Direct Mapping Since the conceptions of the relational model and the Description Logics have some structural similarities, they can be used to automatically derive a generic RDB2RDF mapping. Regarding, e.g. relations as 'containers' for entities that share the same relational structure, this notion can be compared to concepts in Description Logics. Following this idea, the W3C RDB2RDF working group published a W3C Recommendation [11] for the generic generation of an RDF schema, based on the schema structures of a relational database. Some example translation patterns are shown in Table 2. In its entirety this recommendation provides means to convert any input database to RDF without manual mapping definitions.

Being based completely on tables and its schema definitions, the resulting ontology is a *schema ontology* reflecting the database structure. This has to be distinguished from a *domain ontology* that is actually *modeled* by a knowledge engineer. Thus even though direct mapping has the advantage of working automatically, it is not able to generate ontologies that reflect the nature of a certain domain, or to map the database at hand to a given vocabulary or ontology.

RDB to RDF Mapping Language Besides this automatic approach, modeling a domain ontology by hand requires a language to express the corresponding mapping definitions. One such language, that recently became a W3C Recommendation [7], is the *RDB to RDF Mapping Language (R2RML)*. It provides means to express so called *triples maps*, that determine how a certain triple template is filled with relational data. Such definitions are expressed as RDF graphs using the Turtle serialization. The main structure of such a triples map is depicted in Figure 1. The logical table, the actual data is retrieved from, is defined via the `rr:logicalTable` property. Its value is a blank node that can either refer to an existing relational table or view of the underlying database via `rr:table`, or contain a custom SQL query assigned via the `rr:sqlQuery` property. The definitions of how to generate RDF nodes, i.e. RDF resources or literals, are expressed in *term maps*. These generate resources or literals based on column values of the underlying logical table or constant expressions. The column values may be used as is, or referenced in a custom template string. Depending on where such term maps are used, they are called *subject map*, *predicate map* or *object map*, with corresponding predicate and object maps being grouped in one *predicate object map*. Moreover it is possible to relate subject or predicate object maps to RDF graphs, which is also represented using a term map. Such term maps are then called *graph map*. An example of an RDB2RDF

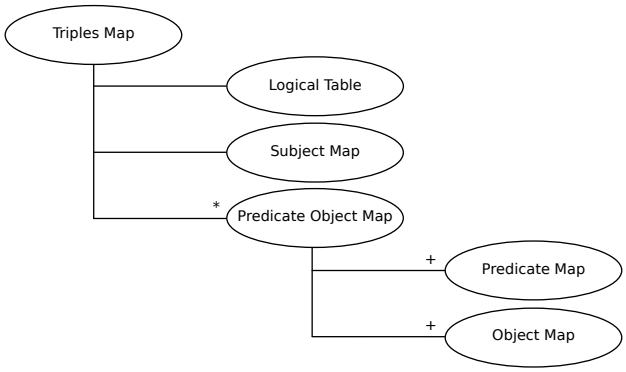

Fig. 1: General structure of an R2RML triples map. Parts marked with '+' and '*' may appear at least once and zero or multiple times, respectively.

```
<#DepartmentTriplesMap>
    rr:logicalTable [ rr:tableName "DEPT" ];
    rr:subjectMap [
        rr:template "http://ex.org/department{DEPTNO}";
        rr:graphMap [
            rr:constant ex:DeptGraph
        ];
    ];
    rr:predicateObjectMap [
        rr:predicateMap [
            rr:constant ex:location
        ];
        rr:objectMap [
            rr:column "DNAME"
        ];
        rr:graphMap [
            rr:constant ex:DeptGraph
        ];
    ].
```

Listing 1.1: R2RML triples map example

mapping defined in R2RML is given in Listing 1.1. Besides this, R2RML provides further features like references between triples maps and a well defined datatype handling.

Sparqlification Mapping Language The *Sparqlification Mapping Language (SML)* is a mapping language, designed to be intuitive and expressive [15]. Since there are tools to convert SML to R2RML and vice versa[8], both languages are of equal expressiveness whereas SML is terser and requires less syntactical noise. The main entities defined with SML are *view definitions*. Such a view definition is shown in Listing 1.2[9]. The actual view definition is declared by the `Create View ... As` keywords in line 1. The remainder of a view definition is structured in three parts. The `From` directive (line 10-12) defines the logical table based on a physical table or view contained in the considered

[8] `https://github.com/AKSW/Sparqlify-Extensions`
[9] Prefix definitions in Listing 1.2 are omitted for brevity

```
1  Create View employee As
2    Construct {
3      ?empl ex:worksAt ?dept .
4      ?dept rdfs:label ?dnme .
5    }
6    With
7      ?empl = uri(ex:employee, '/', ?emp_id)
8      ?dept = uri(ex:dept, '/', ?dpt_id)
9      ?dnme = plainLiteral(?name)
10   From
11     [[SELECT emp.emp_id AS emp_id, emp.dept_id AS dpt_id, dept.name AS
           name
12       FROM emp JOIN dept ON emp.dept_id=dept.id]]
```

Listing 1.2: Example of a view definition in SML

Argument 1	Argument 2	Argument 3	Argument 4
type:			
0... blank node	value expression	*empty*	*empty*
1... URI		*empty*	*empty*
2... plain literal		language tag	*empty*
3... typed literal		*empty*	datatype

Fig. 2: Overview of the arguments of SML term constructors

database, or a custom SQL query (denoted by the opening and closing double brackets). An RDF quad pattern is defined in the `Construct` part by means of URI, blank node or literal constants (e.g. `ex:worksAt`) and variables (e.g. `?empl`, `?dept`). This quad pattern has the same purpose as in a SPARQL CONSTRUCT query: It is used to create triples, replacing the variables with matching RDF nodes. The actual bridge between the logical table and the quad pattern is given in the `With` part. There, the variables used in the quad pattern are defined via term constructor expressions (line 7-9), where the term constructor expressions refer to columns of the logical table (e.g. `?emp_id`, `?dpt_id`). Such term constructor expressions can be seen as generic quaternary functions that require the type of the RDF node to return and further expressions that are evaluated to represent a URI, blank node or literal value as well as the datatype (in case of a typed literal) and language tag (in case of a plain literal). An overview of the possible arguments and their meanings is given in Figure 2. Accordingly, the typed literal `"42"^xsd:int` can be created using the term constructor function $tc(3, 42, \epsilon, xsd : int)$, with ϵ being the empty input. A more schematic view of the SML view system, introducing the SML terminology as defined in [15], is given in Figure 3.

Apart from the structures mentioned above, a view definition may also contain a `Constraint` clause, used to give constraint hints to improve the query performance. Since the `Constraint` clause does not contribute to the actual semantics as far as the RDB2RDF mapping is concerned, it is not considered in the following.

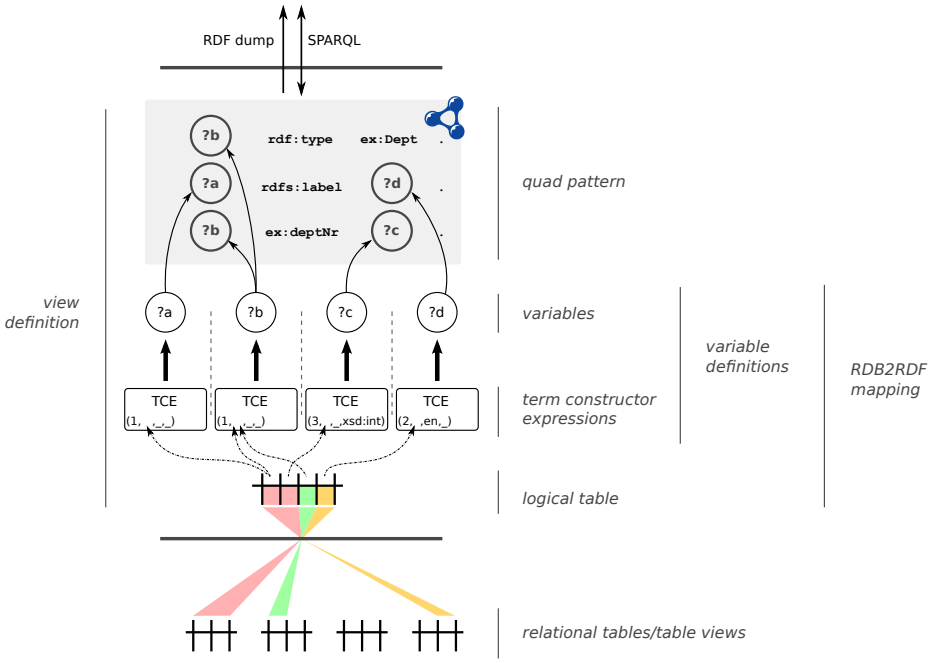

Fig. 3: The SML view definition system

RDB2RDF Tools Currently there is a wide range of RDB2RDF tools. A selection of implementations mentioned in [12] and [16] comprises *Ultrawrap*[10], *D2RQ*[11], *Virtuoso* [17] and *Spyder*[12]. Further state of the art solutions are *-ontop-*[13], *Asio Semantic Bridge for Relational Databases*[14], *SparqlMap* [18] and *Sparqlify*[15]. An overview of the tools is given in Table 3. There, the different ways to actually define an RDB2RDF mapping are shown. Besides graphical tools like *Snoggle*[16] and the mapping languages already introduced, further custom languages to express transformation rules can be found. The direct mapping approach, applied by the Ultrawrap tool, is also contained.

Besides this, the tools differ e.g. with regards to the license used, the supported features, and their maturity. The Virtuoso triple store, which also has RDB2RDF capabilities, can be considered to be the most mature of the introduced tools. If the performance of the query engine is of major importance, the -ontop- system proofed to be very efficient[17]. A tool with a wide range of features is the Sparqlify SPARQL to SQL rewriter,

[10] `http://capsenta.com/ultrawrap`
[11] `http://d2rq.org/`
[12] `http://www.revelytix.com/content/spyder`
[13] `http://ontop.inf.unibz.it/`
[14] `http://bbn.com/technology/knowledge/asio_sbrd`
[15] `http://sparqlify.org`
[16] `http://snoggle.semwebcentral.org/`
[17] see `http://ontop.inf.unibz.it/?page_id=74` for more details

	License	Mapping Definition	Version
Asio Semantic Bridge for Relational Databases	proprietary; free of charge for non-profit use	graphical (Snoggle)	—
D2RQ	Apache 2.0 License	D2RQML	—
-ontop-	Apache 2.0 License	Quest Mapping syntax, R2RML	1.10
SparqlMap	GNU Lesser General Public License	R2RML	0.6.1
Sparqlify	Apache 2.0 License	Sparqlification Mapping Language	0.6.6
Spyder	proprietary; free of charge	R2RML	2.1.2
Ultrawrap	proprietary	Direct Mapping	—
Virtuoso	GNU General Public License	Virtuoso Meta Schema Language	7.0.0

Table 3: Overview of RDB2RDF tools

which e.g. supports the integration of the Linked Data publishing tool *Pubby*[18] and the SPARQL browser *SNORQL*[19], provides an experimental web administration interface[20] and SPARQL UPDATE support[21].

2.2 Data Quality

This section covers the state of the art with regards to *data quality*. Besides some historical remarks, the term 'data quality' is introduced and different theoretical models are presented. An established proceeding is to break down the quality of the considered data into measurable quality *dimensions*. This approach is further discussed, followed by an overview of common data quality assessment methodologies.

Overview Quality is considered not just since the so called *Information Age*. First publications on this subject go back to the 1940s [10], mainly considering quality from an economic and management perspective. These considerations led to current standardizations like the ISO 9000 standards familiy [19] and are also subject of published best practices of local administrations and governments, e.g. in the education sector [20,21,22] or the international financial marked [23].

Other than the approaches, dealing with quality of processes, products and services, *data quality* refers to the quality of stored information[22]. The understanding of data in this context, is based on the definition in [25], stating that data *"represent real world objects, in a format that can be stored, retrieved, and elaborated by a software procedure, and communicated through a network"*. There are different categorizations of data [26,25], whereas the focus of this report is clearly on *structured data*, neglecting other categorizations.

Even though the data quality domain differs from the economic and management perspective, general quality definitions are still applicable. Phrases like *'freedom from deficiencies'* or *'fitness for use'* [10] are also suitable definitions for data quality and are used widely in the literature. In the following section further definitions and models are introduced, giving a deeper understanding of data quality.

Data Quality Models and Definitions *"While fitness for use captures the essence of quality, it is difficult to measure quality using this broad definition."* [27] Thus, further research was done to find more suitable data definitions and models. One approach examined the weaknesses and strengths of the definitions of quality in general as *'excellence'*, *'value'*, *'conformance to specifications'* and *'meeting or exceeding consumer expectations'* [28]. The outcome of this study was that none of them alone can describe quality satisfyingly since it often depends on the underlying use case.

[18] `http://www4.wiwiss.fu-berlin.de/pubby/`
[19] `https://github.com/kurtjx/SNORQL`
[20] `https://github.com/AKSW/Sparqlify`
[21] `https://github.com/AKSW/Sparqlify-Extensions`
[22] Though there are publications, e.g. [24], pointing out the difference between *data* and *information*, this distinction is not made in this report.

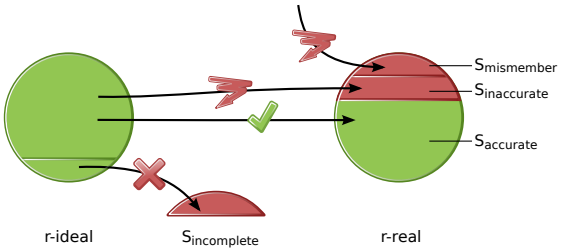

Fig. 4: Accuracy model of PARSSIAN et al. [29]

Other approaches to find models that describe quality, especially *data quality*, in a more formal way, often only refer to certain aspects of quality. One such model is described by PARSSIAN et al. [29,30] and depicted in Figure 4. There, the process of capturing a model (*r-real*) of the real world (*r-ideal*) is shown[23]. During this process information gets lost (information flow to $S_{incomplete}$), corrupted (information flow to $S_{inaccurate}$) or is added without describing the considered domain (information flow to $S_{mismember}$). Hence, this model concentrates on the accuracy and completeness of a system, describing the real world, and has similarities to SHANNON's model of a noisy channel [31].

The model created by WAND and WANG [32] uses the notion of an *information system*, being a *"representation of a real world system as perceived by the user"* and describes the mapping problems between the real world and its representation. Certain

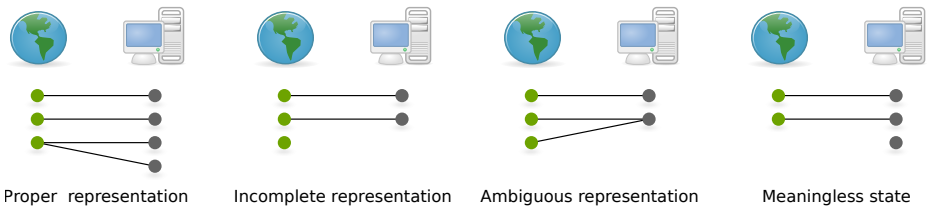

Fig. 5: Quality model of WAND and WANG [32]. In each of the depicted mappings the left hand side represents the real world system states and the right hand side the mapped states in the information system.

quality aspects are derived by looking at possible mapping deficiencies as shown in Figure 5.

In both models quality is described as the difference between the domain that should be expressed and the expression itself. The model of WAND and WANG moreover explicitly considers *data quality* as the extend to which the real world system can be modeled without errors. There are also other sources sharing this definition, e.g. [33,34], or

[23] Within the model *r-ideal* is used to denote a system that models the real world *ideally*, i.e. accurately and completely, whereas *r-real* refers to a *real* system that might have errors.

ORR [35] stating that *"Data quality is the measure of the agreement between the data views presented by an information systems [sic] and the same data in the real-world"*. The question that remains, is how this difference can be determined and measured.

Other models share a more process oriented view, regarding the data quality of an *information product*, being created in a chain of production steps [36,37]. Such models should provide means to examine *why* a particular piece of information has a certain quality and which are the most influencing processing steps as far as data quality is concerned. These approaches consider data quality as a composite phenomenon and also try to develop calculation models to compute quality by means of a composition algebra [38].

Even though these models provide different views of quality, they all face the same problem of making quality tangible and computable. Since quality is a broad concept and depends on the use case, it was increasingly considered as *multi-dimensional* in the sense that is has many different aspects that can be examined separately. This idea is introduced in the following section.

Data Quality Dimensions Since data quality touches many different aspects it can be decomposed into different *data quality dimensions*. Depending on the use case, only a subset of all known dimensions needs consideration, breaking the problem of measuring quality down into smaller pieces. Even though regarding data quality as a multi-dimensional concept is a common view in the literature and an enormous amount of dimensions were proposed (cf. Section C), *"there is no agreement on the set of dimensions characterizing data quality"* [39]. Another issue is that there is also no consensus on what particular dimensions mean, leading to multiple definitions of single dimensions. This situation to some extent reflects the circumstance that quality is often considered in connection with a certain use case or application domain. Besides these differing views of quality dimensions there are also different approaches to actually infer them from a given use case. These are categorized into *theoretical*, *empirical* and *intuitive* approaches [39,25].

In a theoretical approach the modeled system is considered in a more abstract way, deriving a formal model to detect and describe quality issues. An example of such an approach is the quality model of WAND and WANG. Due to the presented abstraction, viewing the development of an *information system* as a mapping problem of the real world, several artifacts can be derived that are of interest. These are design deficiencies referring to the errors shown in Figure 5 (incomplete representation, ambiguous representation and meaningless state) and operation deficiencies standing for inappropriate behaviour of the system. With these deficiencies at hand one can define quality dimensions as shown in Table 4. The process oriented models mentioned in the previous section are theoretical approaches as well.

The empirical approach does not consider formal models but takes stakeholder opinions into account. In most cases such approaches are based on a user survey as in the method of WANG and STRONG [40]. There, a survey performed in multiple steps led to a shortlisted catalogue of 19 quality dimensions grouped in four categories shown in Figure 6.

Dimension	Description
Accuracy and Precision	*"inaccuracy implies that the information system represents a real world state different from the one that should have been represented."*
Reliability	indicates *"whether the data can be counted on to convey the right information; it can be viewed as correctness of data."*
Timeliness and Currency	refers to *"the delay between a change of the real-world state and the resulting modification of the information system state."*
Completeness	is *"the ability of an information system to represent every meaningful state of the represented real world system."*
Consistency	inconsistency of data values occurs if there is more than one state of the information system matching a state of the real-world system; therefore *"inconsistency would mean that the representation mapping is one-to-many."*

Table 4: Quality dimensions derived from the quality model of WAND and WANG [32]

When following an intuitive approach, data quality dimensions are defined *"according to common sense and practical experience"* [25]. A concrete example of this approach is given by REDMAN [41]. The corresponding data quality dimensions are listed in Table 5.

These three approaches and their prerequisites are summarized in Figure 7. Apart from the dimensions presented for the three approaches, an overview of all dimensions introduced in the considered literature, can be found in Section C. To ease the understanding, the dimension definitions or descriptions were normalized using a shared vocabulary for formulae, and consolidated in case multiple dimensions share the same meaning.

Quality Assessment Methodologies Having defined the data quality dimensions to assess, a certain methodology for the actual assessment has to be applied. *Quality assessment* here means *"evaluating if a piece of information meets the information consumer's needs in a specific situation"* [42], or, if possible, *"assigning numerical and categorical values to [data quality] dimensions"* [43]. As with data quality dimensions, there are several different approaches proposed (cf. [44] and [25] for current methodology surveys). Nonetheless, many of the methodologies mentioned in the literature share the same general key quality evaluation activities [45,46,34]. These are to first specify the context-dependent quality requirements, followed by the identification and distinction of actual quality problems. After that the current quality status is assessed, analyzing the evaluation results afterwards. To present a selection of more concrete

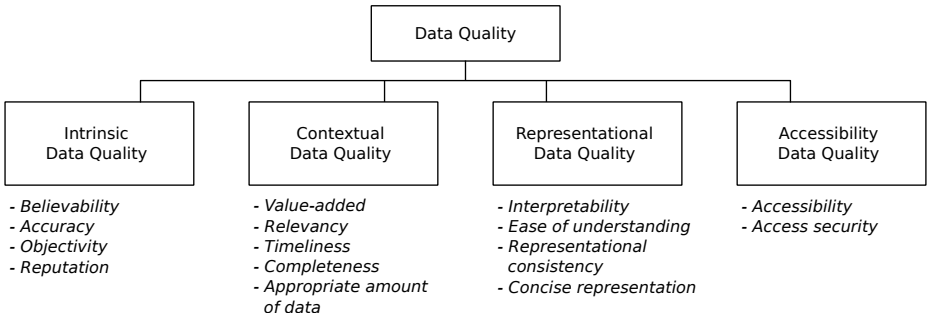

Fig. 6: Quality dimensions according to WANG and STRONG [40]

Type	Dimension	Description
Data value	Accuracy	*"Distance between v and v′, considered as correct"*
	Completeness	*"Degree to which values are present in a data collection"*
	Currency	*"Degree to which a datum is up to date"*
	Consistency	*"Coherence of the same datum, represented in multiple copies, or different data to respect integrity constraints and rules"*
Data format	Appropriateness	*"One format is more appropriate than another if it is more suited to the user needs"*
	Interpretability	*"Ability of the user to interpret correctly values from their format"*
	Portability	*"The format can be applied to as a wide set of situations as possible"*
	Format precision	*"Ability to distinguish between elements in the domain that must be distinguished by users"*
	Format flexibility	*"Changes in user needs and recording medium can be easily accommodated"*
	Ability to represent null values	*"Ability to distinguish neatly (without ambiguities) null and default values from applicable values of the domain"*
	Efficient use of memory	*"Efficiency in the physical representation. An icon is less efficient than a code"*
	Representation consistency	*"Coherence of physical instances of data with their formats"*

Table 5: Quality dimensions proposed by REDMAN [41] (cited from [25])

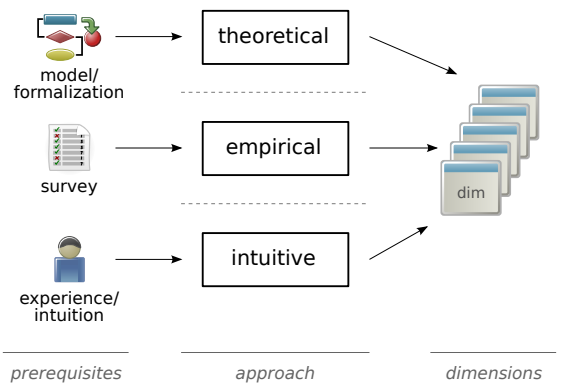

Fig. 7: Approaches to derive quality dimensions to consider for a given domain and their prerequisites

methodologies, proposed in the literature, one popular general data quality assessment methodology was chosen as well as three approaches focusing on Semantic Web data.

The methodology proposed by Lee et al. to assess data quality in a general-purpose manner is called *AIMQ (AIM quality)* [45]. The main components of AIMQ are sketched in Figure 8. The first component defines which quality dimensions to use and groups

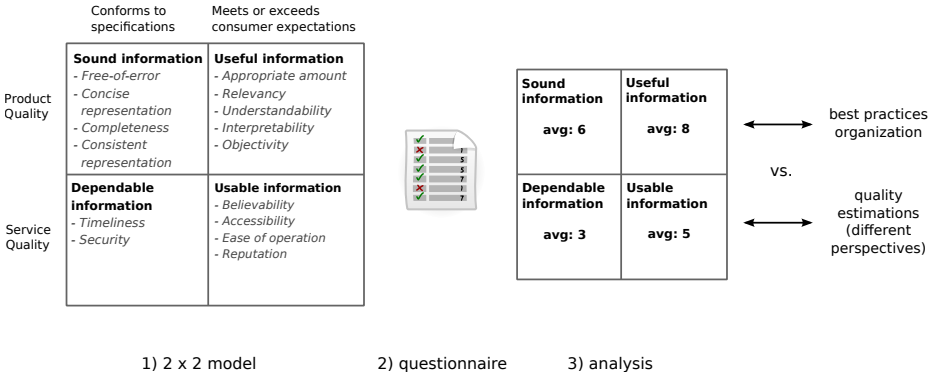

Fig. 8: Schematic representation of the AIMQ components

them into four categories arranged in a 2×2 matrix. The categories are *sound information*, *useful information*, *dependable information* and *usable information*. Based on this 2×2 model a questionnaire is set up covering all dimensions mentioned. After doing the actual measurement based on a survey using the questionnaire, the results are averaged per quadrant of the 2×2 matrix. These results are then compared with a benchmark of a fictional best practice company, as well as with the expectations and estimations of different roles using or producing the data under assessment.

Another methodology considering semantic metadata is introduced by Lei et al. [34]. The main steps here are to first specify and weight the quality issues to assess and to provide a gold standard to compare with. Next, the actual assessment is run, encompassing three tasks. The first task is to detect data problems (e.g. completeness, accuracy) based on a comparison with the provided gold standard. In a second task it is checked if the metadata reflects the real world status, considering other trusted knowledge resources. Finally, the consistency of the involved ontologies is checked. Based on the assessment results and comparisons with best practices the data quality status is calculated in the last step.

A very recent methodology following the crowd-sourcing approach is applied by the *TripleCheckMate* tool [47]. It is tailored for the assessment of RDF data providing a user interface where users can log in and get credit points per detected error. The methodology comprises four steps. In the first one, the resources to assess are chosen based on a manual choice, on the resource's class, or selected randomly. In the following step the evaluation mode is selected, which can be *automatic*, *semi-automatic* or *manual*. Step three performs the resource evaluation. In case the manual mode was chosen, the user then analyzes all resources individually otherwise the evaluation is run (semi-)automatically. The improvement of the assessed data can be performed directly by editing the erroneous resources or by creating a patch using the *Patch Request Ontology* [48].

The last methodology presented is inspired by the ideas of test driven software development, coined *test-driven data quality methodology* [49]. To assess the data quality of a given RDF dataset, *data quality test cases* are used. These are SPARQL queries checking if the dataset contains triples violating certain constraints that must hold. The constraints can be inferred from the vocabularies and ontologies used in the dataset or created manually. After having set up all quality test cases they are run against the considered SPARQL endpoint which returns all constraint violations. These can then be used to correct data or improve the processes that led to the erroneous data.

3 Quality of RDB2RDF Mappings

Since RDB2RDF tools produce Semantic Web data, or even Linked Data, the consideration of its quality is of even more importance because the data is re-used in many different scenarios and the actual context is lost [50]. But even though data quality is well analyzed in the database and information system area [41,51,52,53,25] and there are approaches to tackle data problems [54,55,56,57,58,59,60,61,62] and measure dataset statistics [63] or even quality scores in the Semantic Web [50,64,65,66,67,47,9,49], there were none found that cover RDB2RDF applications in greater depth.

In this section the theoretical framework for a quality assessment of RDB2RDF mappings is considered. After motivating the major design decisions, formal foundations are introduced. These cover a theoretical model and a terminology to describe a quality assessment formally. Afterwards, a methodology is proposed and the derivation of applicable quality dimensions is analyzed. The section is closed with the definition and discussion of concrete metrics to apply in an RDB2RDF quality assessment.

3.1 Design Considerations

Since the RDB2RDF approach is rather generic, covering a variety of application scenarios, in this section the scope considered in this report is narrowed down by introducing concrete design decisions. A classification scheme for the different RDB2RDF techniques and their applications, proposed in [12], is depicted in Figure 9. The main applications considered here are the mass generation of Semantic Web data and the ontology based access to relational data. Accordingly, RDB2RDF techniques to model existing or new, domain-specific ontologies are covered, without applying database reverse engineering. More specifically, the data quality assessment methodology to develop should support the modeling of datasets of a certain domain, backed by existing relational data. Hence, ontology learning approaches like the direct mapping technique are not considered, since *"the resulting ontology looks a lot like a copy of the database relational schema as all relations are translated to RDFS classes, even the ones which are mere artifacts of the logical database design (e.g. relations representing a many-to-many relationship between two entities)"* [12]. Moreover, the direct mapping technique has some difficulties in practice, e.g. with regards to composite (foreign) keys [68,4] or `NULL` values [69]. Thus, a *domain-driven modeling* [14] is preferred to the automatic generation of a schema ontology.

Even though, the quality of the relational data to map has an impact on the resulting RDF data, it is not considered in this report. Data quality aspects of relational data are already discussed in detail [25] and are thus not part of this study.

To evaluate the theoretical findings in terms of the quality of RDB2RDF mappings in practice, any of the tools introduced in Section 2.1 can be used. Although all implementations follow the same principle of a tuple-wise conversion based on mapping definitions, they differ in the number of features, performance and maturity. So, to avoid practical restrictions of the quality assessment and provide a usable and stable evaluation software, the underlying tool should be flexible, feature-rich, performant and mature. Besides this, the mapping language supported by the RDB2RDF tool which will be utilized for the assessment should be expressive and at least compatible to the R2RML

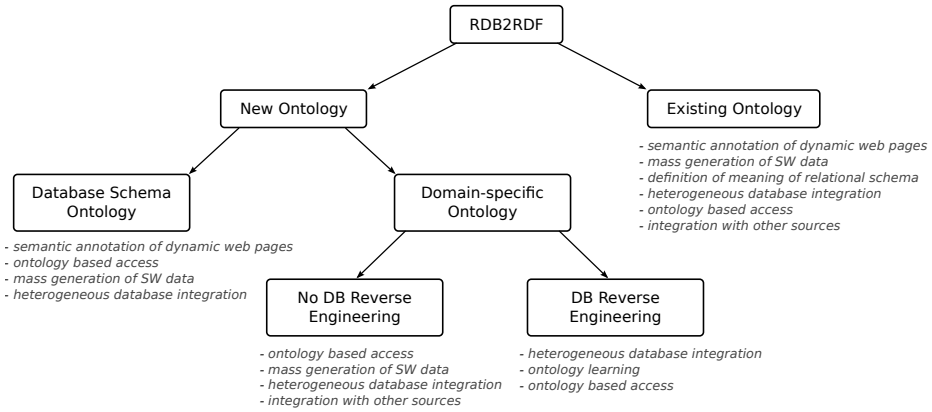

Fig. 9: Classification of RDB2RDF techniques [12]

standard to be as universal as possible. Furthermore the tool should be available under a permissive license with the option to inspect its source code.

One of the more feature-rich, flexible and mature state of the art RDB2RDF tools is the Sparqlify SPARQL to SQL rewriter. Sparqlify is utilized to provide SPARQL access to different important relational datasets like *OpenStreetMap* [70] or *PanLex* [71]. Moreover, Sparqlify proved to be competitive in terms of performance and scalability [15] and turned out to be very efficient serving large datasets. The tool is available as free software[24] and the project team is open for community feedback and issue reports. Besides this promising status quo, there are further development and research efforts to improve and extend Sparqlify. The underlying mappings are defined in the Sparqlification Mapping Language and there are also tools to convert them to R2RML and vice versa[25].

In its entirety the Sparqlify project is considered suitable to be utilized for a practical RDB2RDF quality assessment. Sparqlify and the Sparqlification Mapping Language are thus used as foundations for further quality considerations in this report.

3.2 Formal Foundations

In this section a formal terminology is defined to be able to describe an RDB2RDF quality assessment methodology and actual metrics. Besides the concepts of a *view definition*, a *quad pattern* and a *relation* which were introduced in previous sections, the notion of a *dataset* is used. A dataset is usually defined as a set of graphs, that consist of triples [72]. For the sake of simplicity, the terminology describing the assessment methodology and metrics in this report uses a slightly different definition. Here, the abstraction of a dataset to assess refers to a set of RDF triples, not regarding RDF graphs. This deviation does not restrict the assessment capabilities, but eases later definitions in terms of conciseness and understandability.

[24] `https://github.com/AKSW/Sparqlify`
[25] `https://github.com/AKSW/Sparqlify-Extensions`

Further clarifications have to be made with regards to relations. In the following, only non-metadata relations are considered. Moreover, it is assumed that they are in a consistent state.

The notion of an *RDB2RDF mapping* is fundamental for the RDB2RDF approach and defined as follows:

Definition 1. *Let $\mathcal{H}$ denote the set of valid RDB2RDF mappings, $\mathcal{V}$ the set of valid view definitions, $\mathcal{P}$ the set of valid quads in a pattern, $\mathcal{Q}$ the set of valid quad variables, $\mathcal{D}$ the set of valid RDF datasets and $\mathcal{T}$ the set of valid RDF triples. An **RDB2RDF mapping** $H \in \mathcal{H}$ is a tuple (V, RDB, D) where*

- *$V \subset \mathcal{V}$ is a finite set of SML view definitions, with the option to access the quads in the quad pattern ($quads(v_i) \subset \mathcal{P}$) and relational table ($rel\_table(v_i)$) of each view definition $v_i \in V$. Given a quad variable $q \in \mathcal{Q}$, its term constructor can be retrieved via $term\_constructor(q)$. The term constructor's RDF term type, i.e. whether it generates a URI, blank node, typed or plain literal, is returned by the function $term\_type(term\_constructor(q))$. Moreover, the set of relational columns referenced in the term constructor of q can be retrieved with $cols(term\_constructor(q))$*
- *RDB is the set of relations contained in a considered relational database*
- *$D \in \mathcal{D}$ is the RDF dataset generated when applying all view definitions $v_i \in V$ to the relations in RDB. $D \subset \mathcal{T}$ is a finite set of valid RDF triples.*

Based on this, the conception of a *scope* can be defined:

Definition 2. *Given the sets $\mathcal{N}$, $\mathcal{R}$, $\mathcal{L}$, $\mathcal{Q}$, $\mathcal{T}$, $\mathcal{D}$, $\mathcal{V}$ and the power set function $\mathbb{P}(\ldots)$ with $\mathcal{T}$, $\mathcal{D}$, $\mathcal{V}$, $\mathcal{Q}$ defined as above and*

- *$\mathcal{D} = \mathbb{P}(\mathcal{T})$*
- *$\mathcal{N} = \mathcal{R} \cup \mathcal{L} \cup \mathcal{Q}$ denotes the set of valid nodes, i.e. all valid resources $\mathcal{R}$, literals $\mathcal{L}$ and quad variables $\mathcal{Q}$*

*The **quality assessment scope** of a piece of data x is a function defined as follows*

$$scope(x) = \begin{cases} \text{node scope } S_N & \text{if } x \in \mathcal{N} \\ \text{triple scope } S_T & \text{if } x \in \mathcal{T} \\ \text{dataset scope } S_D & \text{if } x \in \mathcal{D} \\ \text{view scope } S_V & \text{if } x \in \mathbb{P}(\mathcal{V}) \end{cases} \tag{1}$$

Accordingly, the scope is a categorization of the granularity a certain piece of data has. This is useful since different *'amounts'* of context information can be needed for the assessment. These amounts correspond to the introduced scopes, i.e. they can either be the whole dataset, one triple, one node or a set of view definitions. These scopes also correspond to the possible domains of the functions that do the actual computation of a quality score.

Definition 3. *A **mapping quality metric** M is a pair (f, θ) where f is a quality score function and θ is a numerical value representing a threshold. A quality score function f computes a numeric quality score $f(x)$ of a piece of data x. A low quality score reflects*

low quality where the worst possible quality score is 0. Perfect quality is represented by a quality score of 1.

A mapping quality metric $M = (f, \theta)$ can be further classified as follows:

$$M \text{ is called } \begin{cases} \text{node metric} & \text{if } dom(f) = \mathcal{N} \\ \text{triple metric} & \text{if } dom(f) = \mathcal{T} \\ \text{dataset metric} & \text{if } dom(f) = \mathcal{D} \\ \text{view metric} & \text{if } dom(f) = \mathbb{P}(\mathcal{V}) \end{cases}$$

where dom(...) returns the domain of a function.

It has to be noted, that other than proposed in [42,64,66], the concept of a *quality indicator* is not used in the methodology proposed in this report. Another difference is that the quality score function, there called *scoring function*, is not intended to be reusable among different metrics. This is not due to a limitation of the methodology, but simply not regarded necessary as none of the metrics that were implemented in *R2RLint* shared any considerable functionality to reuse.

To initialize an assessment run, a configuration is needed, which is defined as follows:

Definition 4. *A **quality assessment configuration** C is a set of mapping quality metrics $\{M_1, M_2, \ldots, M_n\}$ representing all metrics enabled for an assessment, together with their threshold initializations.*

This conceptualization allows enabling and disabling metrics to fit the given assessment needs as well as defining the per metric thresholds. The threshold concept was introduced to reduce the amount of measurement data and to be able to concentrate on cases that are considered to be critical, as only those quality scores are reported, that are worse than the configured threshold.

Definition 5. *A **quality assessment** (H, C, S) is the process of evaluating the quality score function f_i of every metric $M_i \in C$ on a certain RDB2RDF mapping H with*

- *$D \in \mathcal{D}$ being the RDF dataset generated by H*
- *$V \subset \mathcal{V}$ being the view definitions of H .*

It is further derived, that

$$T = \bigcup_{t \in D} t \tag{2}$$

is the set of triples in D and

$$N = \bigcup_{t \in T} subject(t) \cup predicate(t) \cup object(t) \tag{3}$$

the set of nodes of T, being either a resource or a literal. The functions subject(t), predicate(t) and object(t) of a triple t return its subject, predicate and object, respectively. When applied to sets of triples, these functions will return all subjects, predicates and objects of the corresponding input triples. Additionally μ_{M_i} is defined as a set of metadata of a metric M_i.

Representing the access to a certain tuple position (starting with 0) putting it in subscript brackets, the overall assessment result ρ is defined as

$$\rho = \bigcup_{M_i \in C} \begin{cases} \bigcup_{n \in N} (\mu_{M_i}, f_i(n)) & \text{if } M_{i[0]} = f_i \ \wedge \ dom(f_i) \in \mathcal{N} \\ \bigcup_{t \in T} (\mu_{M_i}, f_i(t)) & \text{if } M_{i[0]} = f_i \ \wedge \ dom(f_i) \in \mathcal{T} \\ (\mu_{M_i}, f_i(D)) & \text{if } M_{i[0]} = f_i \ \wedge \ dom(f_i) \in \mathcal{D} \\ \bigcup_{v \in V} (\mu_{M_i}, f_i(v)) & \text{if } M_{i[0]} = f_i \ \wedge \ dom(f_i) \in \mathbb{P}(\mathcal{V}) \end{cases} \tag{4}$$

ρ is then written to an assessment sink S.

The assessment results comprising pairs $(\mu_{M_i}, f_i(x))$ of a metric's metadata together with its calculated quality score (with respect to the input data x) can then be stored in a configured sink for further inspection. This can be a relational database, a file or other storage mechanisms.

3.3 Methodology

A *quality assessment methodology* as used in this report is a coarse description of how to perform the actual assessment. An assessment methodology should therefore provide a number of steps to be executed in the given order. To find such a general execution plan it is first examined if there are already existing methodologies suiting the needs of an RDB2RDF quality assessment.

The AIMQ methodology [45] uses surveys to get quality scores of considered dimensions. This is not desired, since the assessment tool to develop should run automatically and calculate values that represent the current status of the RDB2RDF mapping quality. Apart from that, AIMQ needs a best practice dataset to compare the assessment results with, which is usually not available in the case of RDB2RDF mappings.

The same problem holds for the methodology of Lei et al. [34] which considers semantic metadata for assessment. Even though its application context might predestine this methodology to be reused in Semantic Web-related quality evaluations, it is also grounded on a gold standard comparison and thus not suitable for an RDB2RDF setup.

The crowd-sourced approach of the TripleCheckMate tool is applicable to assess the generated RDF data but does in no way regard the underlying semantics of an RDB2RDF conversion nor does it allow to explicitly evaluate the mapping definitions. The same holds for the test-driven data quality approach [49].

Thus, ideas of the methodology of Lei et al., the crowd-sourced approach and the test-driven methodology could be used for certain parts of an RDB2RDF quality assessment. But since the underlying structures and focuses differ it would be difficult to integrate them in one methodology. Further methodologies proposed in the literature [29,36,73,27,46,74,75,76,77,33,44,25,78,64,65] were either too generic or tailored for a specific scope and are thus not reusable in the RDB2RDF domain. As a consequence, a methodology specific for the RDB2RDF situation is compiled in the following.

The R2RLint methodology (R2RLM) proposed in this report is based on the RDB2RDF mapping and assessment definition as introduced above. Given a quality assessment $A = (H, C, S)$ its main steps are:

1) Assessment configuration The overall configuration of the assessment comprises three parts: the setup of the database connection to access the set of relations *RDB* of the RDB2RDF mapping H, the selection of metrics to apply together with their thresholds (C) and the configuration of the assessment sink S to write the assessment results to.

2) Automatic assessment run After the configuration, the actual assessment is run, examining the RDB2RDF mapping on the different scopes. Every metric M_i may have access to the underlying relations *RDB* and a service called *pinpointer*. Given a triple $t \in D$ this service can determine all the information of view definitions $V_t \subseteq V$ that most probably generated t. The assessment runner feeds

- all dataset metrics with the generated dataset D
- all triple metrics with all triples $t_j \in D$
- all node metrics with all nodes $n_k \in N$ (with N defined as above)
- all view metrics with the set of definitions V defined in H

When a metric M_i finished the assessment of the given piece of input data, it writes the quality score and a set of metadata μ_{M_i} to the sink S. This metadata may contain pinpointing information, scope information, the actual name of the metric etc. The concrete set of metadata is defined by the metric.

3) Result analysis After the assessment finished, all assessment results ρ are written to the sink S. Depending on the utilized sink, they can now be further aggregated, visualized or stored to document a temporal quality progress. Since the results also contain several metadata to locate the actual error causes, i.e. the view definitions' quads, its term constructors and source relation, a manual repair phase may follow.

3.4 Quality Dimensions for RDB2RDF Mappings

To break down the problem of determining the actual quality of RDB2RDF mappings and the resulting data, different quality dimensions are considered. To compile a set of dimensions that are relevant for the RDB2RDF process the theoretical, empirical or intuitive approach can be followed (cf. Figure 10). Since the application of the intuitive approach would lack scientific soundness and comprehensibility, it is not considered. Even though taking the experiences of a group of RDB2RDF users and experts into account would be valid from a scientific point of view, the RDB2RDF technique seems not widespread enough allowing a survey with a representative amount of questionees. Thus, empirical results were only regarded indirectly in terms of metrics proposed by other literature sources. The method to obtain quality dimensions, which is considered in this report, is the theoretical approach.

To derive quality dimensions of importance for a considered domain on a theoretical basis, a quality model is required. In the case of RDB2RDF mappings not all of the introduced quality models are suitable. First of all, the underlying workflow involves transformations of data that are already given in a relational database, which can be viewed as an information system in the sense of Wand and Wang [32] (cf. Figure 11).

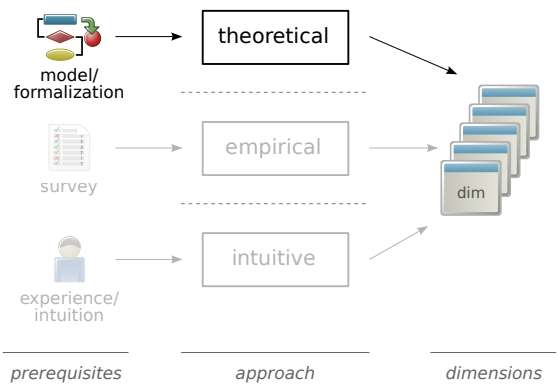

Fig. 10: Approaches to derive quality dimensions to consider for a given domain and their prerequisites (pale approaches are not applied)

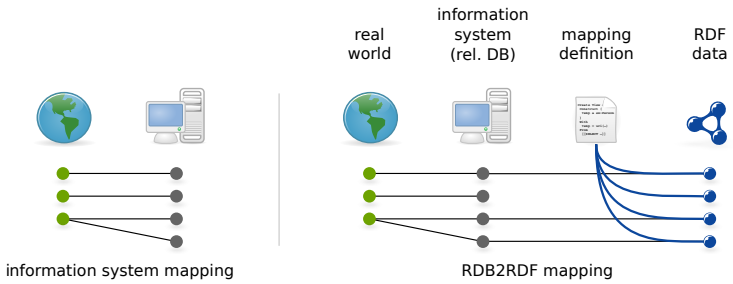

Fig. 11: Comparison of the quality model of WAND and WANG [32] with the RDB2RDF workflow

The data of this information system are then further transformed to be part of another information system, the Resource Description Framework. Accordingly, RDB2RDF could be seen as a transitive mapping problem. But apart from these transformations, the Sparqlification Mapping Language also provides means to add further information not contained in the database. In the models of PARSSIAN et al. or WAND and WANG, this would be considered as introduced inaccuracy. Moreover, since the RDB quality is not evaluated, it is difficult to apply models regarding data quality as an appropriate mapping from a real world to an information system. Even if the relational database would be considered as real world, such a comparison may be unsuitable since the underlying use case does not follow the aim to just convert the hole data in the database to RDF. Thus, a mismatch between database and RDF data may to some extent be intended and should not be an indication of bad quality.

Regarding an RDB2RDF mapping not as a *mapping* but as a *view* brings up the consideration of the *global as view* approach [79,25]. Since such a view based perception is conceptually more free as far as possible restrictions to quality policies are concerned, its quality is also harder to measure. Accordingly there are only few dimensions proposed for this approach [25]. These comprise the considerations whether views

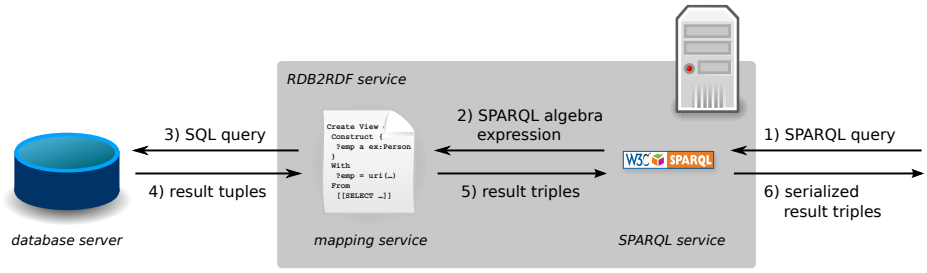

Fig. 12: SML mapping workflow

are *sound*, *complete* and *exact*. In contrast to models assessing if a real world system is *'copied'* accurately to an information system, the view based dimensions are rather abstract, referring to a set oriented assessment. Thus, there are no explicit requirements, that data are not modified, but just that data entities contained in a view must represent entities of the dataset the view is defined on. But since these dimensions are meant to characterize views in general, they are not always suitable to reflect the actual quality of RDB2RDF mappings. Even though they give an impression of how well a mapping definition covers a relational database, this should not be seen as a hard quality criterion. Besides this, these three provided dimensions are not considered sufficient to describe the quality of an RDB2RDF mapping adequately.

Since the RDB2RDF mapping can also be regarded as a process with certain steps (e.g. data retrieval from the relational database, term construction, triple construction, RDF serialization and RDF output), process oriented models are applicable and provide the best abstraction of the given approaches. Thus, in the following this process is analyzed with regards to points where data quality degradations may occur. Along with this, quality aspects are considered that are affected by these possible degradations. To divide the RDB2RDF mapping workflow into single steps, the mapping model of the Sparqlification Mapping Language is regarded. In Figure 12 the case of processing a SPARQL query is depicted. To answer a query, received by the SPARQL service (1), it has to be parsed and transformed to primitives of the SPARQL algebra (2). Afterwards the query is combined with the mapping definitions and translated to an SQL query sent to the relational database management system (3). Hereafter, the answer containing the relational result set is retrieved (4), transformed to RDF (5) and output in a proper serialization format (6).

With regards to possible quality degradations, step 1 is not of interest since the input is uninfluenceable user input that has no effect on the quality of the actual output. Step 2 performs a lossless, deterministic transformation that does not influence the output as well. The first time quality is affected, is when the SQL query is built based on the mapping definitions (3). Since these definitions provide a certain view of the underlying database, this affects quality aspects like how *complete* an SML view definition is (in a global-as-view sense) or if the portion of data, to be generated by the view definition is *relevant* for the modeled domain. Assuming, that its actual execution time is

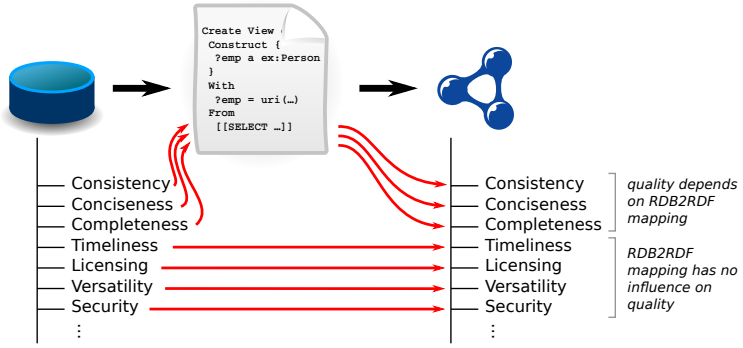

Fig. 13: Quality dimensions depending on RDB2RDF mapping

neglectable[26], running the SQL query in step 4 does not influence the quality. The following transformation to RDF (5) however does have an effect on *representational* and *syntactic* aspects of the generated data, since resource identifiers and literal values are created based on the SQL result set and the mapping configuration. Moreover, the SML quad patterns can be compiled to generate *inconsistent* RDF data. The serialization step (6) is again a lossless and deterministic process that does not harm the quality.

This shows that the main influencing part within the workflow are the mapping definitions. But not all quality dimensions are really affected, as indicated in Figure 13. To gather dimensions relevant for describing quality issues of RDB2RDF mapping, a shortlisting strategy is applied. Starting with data quality dimensions proposed in a recent and comprehensive survey of quality assessment in Linked Data by Zaveri et al. [67] (cf. Table 6), these dimensions are evaluated with regards to their applicability in the RDB2RDF mapping process using the Sparqlification Mapping Language and the introduced formal foundations (cf. Section 3.2). The applicability is determined based

[26] This assumption was made, since the execution time depends on many different factors that are not within the scope of this report. Apart from the fact, that the execution time does not depend on the mere mapping definitions, in theory it can be optimized to be neglectable.

Category	Dimension
Accessibility	Availability
	Licensing
	Interlinking
	Security
	Performance
Contextual	Relevancy
	Trustworthiness
	Understandability
	Timeliness
Intrinsic	Syntactic Validity
	Semantic Accuracy
	Consistency
	Conciseness
	Completeness
Representational	Representational Conciseness
	Interoperability
	Interpretability
	Versatility

Table 6: Overview of the dimensions proposed in [67]

on two issues. First, a dimension is not applicable if it is not relevant for the RDB2RDF process, i.e. the actual quality score does not depend on the RDF transformation. Moreover, a dimension is also considered not applicable if there are no quality indicators [42], i.e. it is not possible to actually measure this dimension due to the lack of information needed to do so. In the following the proposed dimensions are considered in more detail giving an explanation why they are used or why not.

Availability (considered) The availability dimension refers to the extend to which data are *"present, obtainable and ready for use"* [67]. An SML view definition only indirectly influences the availability of data, namely when URIs are generated that are not dereferenceable. All other aspects of availability are not influenced.

Completeness (considered) Viewing the completeness quality dimension as *"the degree to which* all required *information is present"* [67], makes it hard to assess without the provision of the gold standard, containing the *required* information to compare with. Thus, the weaker completeness notion from the global-as-view approach is applied, referring to the portion of data, that is covered by a view. Since an RDB2RDF mapping can be regarded as an RDF view on a relational database, the completeness term as used here, describes how well the underlying database is covered. As there is conceptually no need to map all the data values given in the database to RDF, this completeness aspect is of less importance and should not be seen as a hard quality criterion. Nevertheless, getting feedback of the actual portion of data that is used by a view definition helps finding errors in case the completeness value is much greater or much smaller than expected. Moreover, additional completeness metrics can be introduced, like the interlinking completeness or the completeness with respect to modeled classes or properties of reused vocabularies.

Conciseness (considered) Conciseness as understood here covers the avoidance of any kinds of redundancy, be it on the schema, triple or instance level. Such redundancies can arise from low quality view definitions and are considered in the quality assessment.

Consistency (considered) Consistency, expressing the degree to which a dataset is *"free of (logical/formal) contradictions"* [67], highly depends on the view definitions' term constructors and quad patterns. These can produce datatype inconsistencies or ontology violations and are thus considered in the quality assessment.

Interlinking (considered) The importance of providing interlinks to other datasets is already reflected in the Linked Data guidelines [80]. Interlinking aspects can be influenced by a view definition's quad pattern and term constructors and are thus subject of the assessment.

Interoperability (considered) Interoperability issues are violations of best practices like term or vocabulary reuse. Since the generated RDF, and thus the degree of reuse, depends on the term constructors and quad pattern defined in an SML view definition, this dimension is considered in the quality assessment.

Interpretability (considered) This quality dimension measures *"whether information is represented using an appropriate notation"* [67] and thus depends on generated resource identifiers or literal representations as well as certain quad pattern constructs. Interpretability considerations also cover issues elsewhere referred to as *uniformity* or *readability*, and is evaluated in the assessment.

Licensing (not considered) The licensing quality dimension is defined as *"the granting of permission for a consumer to re-use a dataset under defined conditions"* [67]. Since the terms of usage are already determined by the license used for the relational data, in the most cases RDB2RDF tools are not able to influence whether data are *open* or *restricted*. Only in rare cases where relational data is provided under a very permissive license, it may be published under more restricted terms of usage by RDB2RDF tools. Apart from this, there is no standardized way of retrieving the actual license information from relational databases. Usually, such licensing meta information is part of the actual relational data to be mapped to RDF. Since there is also no way to detect licensing metadata in a relational database automatically, it can neither be measured, whether the data contained is open or restricted, nor can be determined if there is any licensing information that could have been provided as RDF data. Thus, the licensing dimension is not considered in the quality assessment.

Performance (considered) The mapping process as introduced comprises different services influencing the overall performance. Besides the actual query rewriting engine there is also the relational database with its search and indexing strategies, the actual RDF generation and serialization, and network bandwidth and latency when transmitting the query results to the client. The only point where RDB2RDF mapping definitions may influence the performance negatively, is when they contain inefficient SQL queries that define logical tables to map to RDF. This issue is not evaluated since the query optimization topic is already covered widely in the literature. Moreover to optimize a query in an RDB2RDF mapping definition, also database details like existing indexes or the underlying storage architecture have to be taken into account, which may not be accessible to the mapping author.

A further performance aspect, discussed controversially[27] and examined in different sources [64,67], is the usage of hash URIs. In the data quality literature, they are usually considered as bad practice as far as performance is concerned, since in case of accessing a Web resource via a hash URI, the whole document has to be retrieved, even though only a fraction of it was requested. Although the usage of hash URIs has no influence on the performance of the RDB2RDF mapping workflow it is evaluated in the quality assessment to be able to give feedback that a bad practice is applied that may harm the performance *in general*.

Relevancy (considered) Even though there are models to compute the relevancy of a document with regards to a certain topic or keywords [81], it is not trivial to calculate if certain data values are relevant or not. Moreover, since *relevancy* refers to a certain task and user [67], there is no easy way to determine relevant data in general. The only

[27] See `http://www.w3.org/wiki/HashVsSlash` for a further discussion

issues that can be measured are coverage concerns, i.e. how detailed a dataset is or how many resources are described. Relevancy is thus considered in the quality assessment.

Representational Conciseness (considered) Representational conciseness in the Semantic Web context mainly refers to issues of URI design and the usage of certain deprecated features of RDF. These depend on the term construction and quad design of an SML view definition and are thus evaluated in the quality assessment.

Security (not considered) Security as a quality dimension mainly covers access control and features to detect unauthorized alteration of data [67]. Since the Sparqlification Mapping Language and the Sparqlify SPARQL to SQL rewriter (as well as the other query rewriting tools) do not provide any means to tackle access control and data integrity, the security dimension is not regarded in the quality assessment.

Semantic Accuracy (considered) The semantic accuracy of data generated by RDB2-RDF mappings comprises the accurate modeling of the semantics of the relational schema and the relational data. Since there is no explicit semantic description that could be used for a quality assessment, semantically inaccurate *data* can not be detected. Nonetheless, if there are any constraints encoded in the relational *schema*, it can be checked whether these are accurately modeled in the RDF domain. The semantic accuracy dimension is thus evaluated in the quality assessment.

Syntactic Validity (considered) The syntactic validity refers to the correct representation and syntax conformance [67]. Since such syntactical aspects highly depend on the actual usage of the term constructors, the syntactic validity dimension has to be covered in the quality assessment.

Timeliness (not considered) Since the RDB2RDF mapping languages introduced in Section 2.1 provide no means to influence *"how up-to-date data is"* [67], the timeliness quality dimension is not considered. Moreover, SPARQL to SQL rewriters like Sparqlify are capable of transforming relational data to RDF on-the-fly which makes the impact on time dependent aspects neglectable.

Trustworthiness (not considered) Trustworthiness, *"the degree to which the information is accepted to be correct, true, real and credible"* [67], primarily depends on the relation between the data's authors and its users. Since it is not the task of an RDB2RDF mapping tool to keep track of data authors and users, trustworthiness is not included in the quality assessment. Moreover, besides the fact that the authorship of the relational data is usually not evaluated, data from different authors may be mixed up in one single resource or statement, making a trust analysis unfeasible.

Understandability (considered) Understandability refers to the data's ease of use by an information consumer and is also called *usability*, *readability*, or *comprehensibility*. This ease of use is mainly achieved by a user-friendly URI design and supporting metadata. Since these aspects can be modeled with the Sparqlification Mapping Language, this dimension is evaluated in the data quality assessment.

Category	Dimension
Intrinsic	Syntactic Validity
	Semantic Accuracy
	Consistency
	Conciseness
	Completeness
Contextual	Relevancy
	Understandability
Accessibility	Availability
	Interlinking
	Performance
Representational	Representational Conciseness
	Interoperability
	Interpretability

Table 7: Overview of the dimensions considered in the RDB2RDF context

Versatility (not considered) Versatility means the versatility of the supported RDF serializations and versatility with regards to internationalized representations of the data values [67]. The former aspect is usually handled by the RDB2RDF tools, independent of the mapping definitions and the created RDF output and thus does not reflect any quality issues of the actual mapping process. Whether any internationalized versions of the data exist, depends on the relational data to map and is therefore not subject of the RDB2RDF quality assessment.

Besides the dimensions proposed by Zaveri et al. [67], further proposals from different literature sources were surveyed (cf. Section C). Among them no additional dimensions were found, that could contribute to the quality assessment of RDB2RDF mappings. Dimensions not covered by Zaveri et al. explicitly, either describe aspects already concerned under a different name, or express more specific cases of dimensions already chosen for the assessment. An overview of the dimensions to assess is given in Table 7.

3.5 Metrics

In this section all the metrics that are implemented and usable in the context of the R2RLint methodology are introduced and explained. To be consistent, these metrics' names describe what is taken care of and not what the actual violations are.

Even though all implemented metrics are compliant with the specifications made in Section 3.2, some definitions of quality score functions made in this section may differ. In these cases the domain of data serving as input to the function does not match the actual scope of the metric. Hence, e.g. a quality score function belonging to a dataset metric and thus having a dataset scope may be presented, getting single triples as input. This deviation was made to avoid complex tuple notations in the quality score function definitions since, depending on what is assessed, e.g. dataset metrics may return multiple quality score values, mathematically represented as quality score value tuples. These slightly simpler expressions were chosen to increase the readability and brevity. Such pseudo quality score functions are then marked as $\hat{f}$. Nonetheless it should be clear how these pseudo quality score functions relate to the actual quality score functions defined on the metrics' scopes.

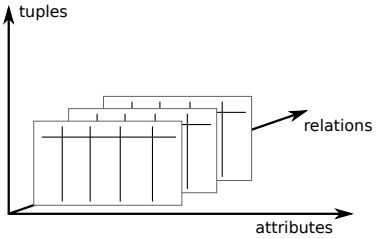

Fig. 14: Completeness dimensions in the relational database context

Availability The only availability concern that is influenceable by the RDB2RDF mapping process, is the dereferenceability of created URIs. To comply with the Linked Data principles [80], and be able to provide further information of a resource via HTTP, the generated URIs should be valid and dereferenceable. This metric was further proposed in different Semantic Web related quality evaluation publications [61,64,62,67]. The assessment of this requirement is covered by the following metric:

Metric 1 (Dereferenceable URIs) *The metric assessing the dereferenceability of a URI resource is a node metric. For an input node $n \in \mathcal{N}$ the dereferenceability quality score function $f_1 : \mathcal{N} \rightarrow \mathbb{R}$ is given as follows:*

$$f_1(n) = \begin{cases} 1 \text{ if } \begin{array}{l} \textit{using n's URI as URL and requesting the corresponding Web resource} \\ \textit{via } \texttt{HTTP GET}\textit{, the returned HTTP response code is 200 after resolving} \\ \textit{any redirects} \end{array} \\ 0 \textit{ otherwise} \end{cases} \tag{5}$$

Completeness To evaluate the completeness in the context of RDB2RDF mappings one can refer to the established *'dimensions'*[28] introduced in the relational database data quality assessment literature [44]. These are shown in Figure 14 and express the completeness with regards to the number of attributes used to represent real world properties, the number of database entries used to represent individuals of the real world and the number of relations describing certain entity types. Adaptions of the proposals in [44] led to the metrics 2 (Schema Completeness), 3 (Population Completeness) and 4 (Property Completeness). So, when mapping a database to RDF, low quality scores of the Schema (i), Population (ii) or Property Completeness Metric (iii) indicate, that

(i) more properties could be created, mapping relational columns to RDF that are not referenced, yet
(ii) more RDF instances could be created utilizing relational tables, not mapped yet, or relaxing the selectivity of the queries that define the logical tables within the view definitions

[28] These dimensions are to be understood as dimensions in a coordinate system, rather than quality dimensions.

(iii) for a considered view definition a greater portion of data and thus a wider range of possible property values could be covered, relaxing the selectivity of the query that defines the view definition's logical table

respectively. Accordingly, these metrics are intended to give hints and should not be understood as *hard* scores, i.e. low completeness values may be intended and thus do not express a bad quality in all cases.

Metric 2 (Schema Completeness) *The metric assessing the ratio between the number of relational columns referenced in the RDB2RDF mapping and the number of columns that could be referenced, is a view metric. To evaluate the schema completeness, for a given relation $\delta \in RDB$ with the attributes $\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$, δ's* column cardinality $|\delta|_{col} = n$ *is defined as the number of columns in δ. Introducing the* referenced column cardinality $|V|_{ref\_col}$ *of a set of view definitions $V \subset \mathcal{V}$ as*

$$|V|_{ref\_col} = \left| \bigcup_{v_i \in V} \left\{ \gamma' \left| \begin{array}{l} q \in \left\{ \begin{array}{l} subject(quads(v_i)) \cup \\ predicate(quads(v_i)) \cup \\ object(quads(v_i)) \end{array} \right\} \cap \mathcal{Q} \wedge \\ \gamma' \in cols(term\_constructor(q)) \end{array} \right. \right\} \right| \tag{6}$$

the Schema Completeness quality score function $f_2 : \mathbb{P}(\mathcal{V}) \to \mathbb{R}$ is computed as follows:

$$f_2(V) = \frac{|V|_{ref\_col}}{\sum\limits_{\delta \in RDB} |\delta|_{col}} \tag{7}$$

Metric 3 (Population Completeness) *The metric measuring the ratio between RDF instances and objects of the relational database is a dataset metric. To get the number of database objects of a relation $\delta \in RDB$ with the attributes $\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$, δ's* relational object cardinality $|\delta|_{rel\_obj}$ *is used:*

$$|\delta|_{rel\_obj} = \left| \pi_{\gamma_{p_1}, \gamma_{p_2}, \ldots, \gamma_{p_m}}(\delta) \right| \tag{8}$$

with $\Gamma_{pk} = \gamma_{p_1}, \gamma_{p_2}, \ldots, \gamma_{p_m}$ representing the (not necessarily compound) primary key of the relation δ and $\pi_{\gamma_j, \gamma_k, \ldots, \gamma_l}(\delta)$ being the projection of δ to its attributes $\gamma_j, \gamma_k, \ldots, \gamma_l$. The cardinality expression of the projection of δ represents the tuple count with duplicate elimination. To avoid counting m:n relations as database objects on its own, a further restriction must hold. Given all referencing foreign key attributes $\Gamma_{fk} = \{\gamma_{f_1}, \gamma_{f_2}, \ldots, \gamma_{f_s}\}$ of δ, the following statement must be true for Equation 8:

$$\Gamma_{fk} \neq \Gamma_{pk}$$

This means that tuples of δ are not counted, if the primary and the foreign key are the same, as in pure m:n relations.

The instance cardinality $|D|_{inst}$ *of a dataset $D \in \mathcal{D}$ is defined as*

$$|D|_{inst} = \left| \begin{array}{l} \left\{ r \left| t \in D \wedge r \in subject(t) \wedge r \not\sqsubseteq (\texttt{rdfs:Class} \sqcup \texttt{owl:Class}) \right. \right\} \cup \\ \left\{ r \left| \begin{array}{l} t \in D \wedge r \in object(t) \wedge r \not\sqsubseteq (\texttt{rdfs:Class} \sqcup \texttt{owl:Class}) \wedge \\ r \notin \mathcal{L} \wedge predicate(t) \neq \texttt{owl:sameAs} \end{array} \right. \right\} \end{array} \right| \tag{9}$$

Thus $|D|_{inst}$ counts all resources not being an `rdfs:Class` *or* `owl:Class`*, whereas objects of* `owl:sameAs` *statements are omitted, to avoid counting resources multiple times that are explicitly stated to be the same. Accordingly the Population Completeness quality score function $f_3 : \mathcal{D} \to \mathbb{R}$ is given as*

$$f_3 = \frac{|D|_{inst}}{\sum_{\delta \in RDB} |\delta|_{rel\_obj}} \tag{10}$$

It has to be noted, that since properties are excluded, the notion of an *instance* here differs from the definition in the RDFS specification [82]. Usually properties are not generated based on database objects, but are rather provided via constant expressions. Hence, to keep the influence of constant and possibly reused external properties low, they are not counted as instances. Moreover, multiple resources that refer to the same instances are only counted once. Besides this, in case static triples are generated, referring to a logical dummy table, e.g. via '`SELECT 1`', the relational object cardinality $|\delta|_{rel\_obj}$ is assumed to be 1.

Metric 4 (Property Completeness) *The metric assessing the completeness with respect to the available values of a property is a view metric. For a given view definition $v_i \in V$ and the tuple cardinality $|\cdot|$ counting all tuples of a given relation, the Property Completeness quality score of v is defined by the function $\hat{f}_4 : \mathcal{V} \to \mathbb{R}$ as follows:*

$$\hat{f}_4(v_i) = \frac{|rel\_table(v_i)|}{|rel\_table(v_i)_{unrestricted}|} \tag{11}$$

where $rel\_table(v_i)_{unrestricted}$ is the logical relational table underlying the view definition v_i with all `WHERE` *clauses removed.*

It has to be noted, that the Property Completeness, as proposed here, might be misleading in cases where certain partitions of a (logical) relational table are covered by dedicated view definitions. Assuming e.g. an employee table with a column that holds the referencing foreign key to the department identifier an employee works in, there might be the aim to create different URI schemes for the different departments. In case the employees are equally distributed to three different departments, each of the corresponding view definitions would have a Property Completeness score of $\frac{1}{3}$, even though the overall completeness is 1. To tackle this issue, the metric above could be modified to consider multiple view definitions if they refer to different partitions of one (logical) table. Nonetheless the metric was introduced in this weaker form to comply with the actual implementation where the partition detection could not be realized due to implementation limitations (cf. Section 4.1).

The introduction of a metric comparing the number of classes with the number of relations, which would refer to the *relations* axis in Figure 14, is omitted here. This is mainly due to the perception that these two quantities are only weakly related. Even though showing how many of the relations of a database are mapped to classes would be beneficial, RDF classes can be derived from many different RDB artifacts, e.g. relational attributes, complex joins etc. which leads to a higher number of classes compared to the number of relations.

Apart from the metrics derived from the completeness dimensions of relational databases, further RDB2RDF related completeness metrics are introduced in the following. One such metric is the assessment of the interlinking completeness, as proposed in the Linked Data quality assessment survey by Zaveri et al. [67]. The interlinking completeness is motivated by the Linked Data principles [80]. Accordingly, the more interlinks exist in a dataset, the better its quality score is with regards to the Interlinking Completeness metric, which is defined as follows:

Metric 5 (Interlinking Completeness) *The metric assessing the ratio of the number of* interlinked external *instances and the total number of instances, is a dataset metric. An instance is considered external, if the string representation of its URI does not start with one of the local URI prefixes, set up for a dataset $D \in \mathcal{D}$. Otherwise the instance is considered local. The set of local resources is thus defined as follows:*

$$R_{local} = \bigcup_{t \in D} \left\{ r \left| \begin{array}{l} r \in \left(\begin{array}{c} subject(t) \cup \\ predicate(t) \cup \\ object(t) \end{array} \right) \cap \mathcal{R} \wedge \\ \\ \text{the string representation of } r\text{'s URI starts with a local prefix} \end{array} \right. \right\} \tag{12}$$

The cardinality function $|D|_{ext\_inst}$, counting the interlinked external resources is given as:

$$|D|_{ext\_inst} = \left| \begin{array}{l} \left\{ subject(t) \left| \begin{array}{l} t \in D \wedge subject(t) \notin R_{local} \wedge \\ object(t) \in R_{local} \wedge \\ subject(t) \not\sqsubseteq (\texttt{rdfs:Class} \sqcup \texttt{owl:Class}) \end{array} \right. \right\} \cup \\ \\ \left\{ object(t) \left| \begin{array}{l} t \in D \wedge subject(t) \in R_{local} \wedge \\ object(t) \notin R_{local} \wedge object(t) \notin \mathcal{L} \wedge \\ object(t) \not\sqsubseteq (\texttt{rdfs:Class} \sqcup \texttt{owl:Class}) \end{array} \right. \right\} \end{array} \right| \tag{13}$$

Given the quality score function $f_5 : \mathcal{D} \rightarrow \mathbb{R}$, the Interlinking Completeness is calculated as follows:

$$f_5(D) = \frac{|D|_{ext\_inst}}{|D|_{inst}} \tag{14}$$

where $|D|_{inst}$ is defined as in Metric 4.

Another best practice concerning the design of RDF data proposes the reuse of established vocabularies. Besides the assessment of how often established vocabularies are reused, which is covered by Metric 21, another aspect of importance is how many of the vocabulary items are actually in use. Considering for example a dataset containing linguistic data, it would be advantageous to reuse a lot of established vocabularies and ontologies to describe the data in a highly interoperable way. But it would not be of much benefit, if e.g. only type assignments for one single class, defined in a reused vocabulary, are provided. Even though this would be vocabulary reuse, it would not allow to query more complex facts using the corresponding vocabulary items, e.g. to retrieve word translations or related part of speech information. This aspect is covered

by the following two metrics, assessing the vocabulary completeness with respect to defined classes and properties. These two metrics were introduced especially for the RDB2RDF case and are not proposed elsewhere, so far.

Metric 6 (Vocabulary Class Completeness) *The metric assessing the ratio of the number of classes* used *and the number of classes* defined*, with respect to a certain vocabulary $D_{voc} \in \mathcal{D}$, is a dataset metric. The classes that are defined in a vocabulary D_{voc} is the set $CLASSES_{voc} \subset \mathcal{R}$:*

$$CLASSES_{voc} = \left\{ r \,\middle|\, r \in \begin{pmatrix} subject(D_{voc}) \cup \\ object(D_{voc}) \end{pmatrix} \wedge\ r \sqsubseteq (\texttt{rdfs:Class} \sqcup \texttt{owl:Class}) \right\} \quad (15)$$

Given the dataset $D \in \mathcal{D}$ and the quality score function $f_6 : \mathcal{D} \rightarrow \mathbb{R}$, the Vocabulary Class Completeness is calculated as follows:

$$f_6(D) = \frac{\left| \Big(subject(D) \cup object(D) \Big) \cap CLASSES_{voc} \right|}{|CLASSES_{voc}|} \quad (16)$$

Metric 7 (Vocabulary Property Completeness) *The metric assessing the ratio of the number of properties* used *and the number of properties* defined*, with respect to a certain vocabulary $D_{voc} \in \mathcal{D}$ is a dataset metric. The properties that are defined in a vocabulary D_{voc} is the set $PROPERTIES_{voc} \subset \mathcal{R}$:*

$$PROPERTIES_{voc} = \left\{ r \,\middle|\, r \in \begin{pmatrix} subject(D_{voc}) \cup \\ predicate(D_{voc}) \cup \\ object(D_{voc}) \end{pmatrix} \wedge\ r \in \texttt{rdf:Property} \right\} \quad (17)$$

Given the dataset $D \in \mathcal{D}$ and the quality score function $f_7 : \mathcal{D} \rightarrow \mathbb{R}$ the Vocabulary Property Completeness is calculated as follows:

$$f_7(D) = \frac{\left| \begin{pmatrix} subject(D) \cup \\ predicate(D) \cup \\ object(D) \end{pmatrix} \cap PROPERTIES_{voc} \right|}{|PROPERTIES_{voc}|} \quad (18)$$

The quality scores of the Vocabulary Class and Vocabulary Property Completeness metrics could also be determined by just taking the view definitions and database values into account. This would introduce further complexity into the metric definitions and implementations but would be more efficient, especially in cases where bigger datasets are assessed.

Conciseness The conciseness dimension refers to the aim of providing data with low redundancy. This is especially important to save network bandwidth, when querying data over a computer network, to save hard disk space when storing the data and most notably to reduce the time to process data.

The aspects of the conciseness dimension, proposed in Zaveri et al. [67] and adapted for the RDB2RDF case, are the concise representation of properties of relational data

objects *(intensional conciseness)* and the non-redundant mapping of relational data objects to RDF resources *(extensional conciseness)*. Furthermore, the conciseness with respect to duplicate statements created by RDB2RDF mappings is regarded. The first two aspects may occur accidentally when copying and pasting lines of the SML view definition, forgetting to update the quad or column variables. Duplicate statements, however, may be introduced e.g. by mapping logical tables that are based on relational joins or in case the database already contains redundant data. The metrics to detect such issues are presented in the following.

Metric 8 (Intensional Conciseness) *The metric assessing how often redundant predicates are used in an RDB2RDF mapping is a view metric. A predicate is considered* redundant *if*

1. *the same subject and object quad pattern variables of a single view definition are used more than once, e.g.:*

```
      ?a ex:worksIn ?b .
      ?a ex:department ?b .   # redundant
      ?a ?p ?b .              # redundant
```

2. *different subject and/or object quad pattern variables of a single view definition are used more than once, with the different pattern variables being created by equal term constructors e.g.:*

```
Create View redundant As
  Construct {
    ?a ex:worksIn ?b .
    ?a ex:department ?c .   # redundant
    ?a ?p ?c .              # redundant
  }
  With
    ?a = uri(ex:person, ?id)
    ?b = uri(ex:dept, '/', ?dept)
    ?c = uri(ex:dept, '/', ?dept)
    ?p = uri(ex:works, ?kind_of_employment, 'In')
    # e.g. ex:worksFullTimeIn
  From
    empl
```

3. *different subject and/or object quad pattern variables of different view definitions are used more than once, with the different view definitions referring to the same (logical) table and the different pattern variables being created by equal term constructors e.g.:*

```
Create View redundant1 As
  Construct {
    ?a ex:worksIn ?b .
  }
  With
    ?a = uri(ex:person, ?id)
    ?b = uri(ex:dept, '/', ?dept)
  From
    empl

Create View redundant2 As
  Construct {
    ?a ex:department ?c .   # redundant
    ?a ?p ?c .              # redundant
  }
```

```
    With
      ?a = uri(ex:person, ?id)
      ?c = uri(ex:dept, '/', ?dept)
      ?p = uri(ex:works, ?kind_of_employment, 'In')
      # e.g. ex:worksFullTimeIn
    From
      empl
```

Since 1. and 2. are special cases of 3., predicate redundancy *can be formulated more generally: A property (variable or constant) of a quad pattern is considered redundant if it appears multiple times in connection with subjects and objects that have equal term constructors using to the same column variables of the same (logical) table, respectively. To track down such duplicates for a given set of view definitions $V \subset \mathcal{V}$ and the multiset union $\biguplus$, a multiset M_{M8} of tuples is generated as follows:*

$$M_{M8} = \biguplus_{v_i \in V} \left\{\!\!\left\{ \left(\begin{array}{l} rel\_table(v_i), \\ term\_constructor(subject(p)), \\ term\_constructor(object(p)) \end{array} \right) \middle| \begin{array}{l} p \in quads(v_i) \wedge \\ subject(p) \in Q \wedge \\ object(p) \in Q \end{array} \right\}\!\!\right\} \tag{19}$$

M_{M8} contains all available combinations of (logical) tables $\times$ subject term constructors $\times$ object term constructors and its counts. The count of a tuple $\tau \in M_{M8}$ can be retrieved via $M_{M8}(\tau)$. Given the (multiset) cardinality $n = \|M_{M8}\|$ and the quality score function $f_8 : \mathbb{P}(\mathcal{V}) \to \mathbb{R}^n$ the Intensional Conciseness is calculated as follows:

$$f_8(V) = \left(\frac{1}{M_{M8}(\tau)} \right)_{\tau \in M_{M8}} \tag{20}$$

f_8 returns a tuple of quality scores containing one score for each of the quads, defined in all $v_i \in V$.

In case of the example under 3., M_{M8} would contain

```
# from '?a ex:worksIn ?b .' in view redundant1:
(empl, uri(ex:person, ?id), uri(ex:dept, '/', ?dept)),
# from '?a ex:department ?c.' in view redundant2:
(empl, uri(ex:person, ?id), uri(ex:dept, '/', ?dept)),
# from '?a ?p ?c .' in view redundant2:
(empl, uri(ex:person, ?id), uri(ex:dept, '/', ?dept))
```

Thus, for each of these tuples $\frac{1}{M_{M8}(\tau)}$ would yield $\frac{1}{3}$.

Metric 9 (Extensional Conciseness) *The metric assessing redundant resources is a view metric. RDF resources are considered redundant if they stem from the same database object or artifact. In SML this is expressed using quad pattern variables that are built applying URI term constructors that refer to the same relational columns and (logical) table. An example that introduces redundant resources in two view definitions is shown below:*

```
  Create View redundant1 As
    Construct {
      ?a a ex:Employee .
    }
    With
      ?a = uri(ex:employee, ?id)
    From
```

```
    empl

Create View redundant2 As
  Construct {
    ?b a ex:Person .
  }
  With
    ?b = uri(ex:person, ?id)  # redundant
  From
    empl
```

A special corner case with regards to the Extensional Conciseness is given, if referencing foreign keys are used, as given in the following example.

```
Create View redundant1 As
  Construct {
    ?a a ex:Department.
  }
  With
    ?a = uri(ex:dept, ?id)
  From
    dept

Create View redundant2 As
  Construct {
    ?b a ex:Person .
    ?b ex:worksIn ?c
  }
  With
    ?b = uri(ex:person, ?id)
    ?c = uri(ex:department, ?dept_id) # redundant
  From
    empl # contains a foreign key empl.dept_id referencing dept.id
```

In the example above, ?c can be considered redundant or inconsistent, with the latter case being covered by Metric 18. To also capture such cases, an extra normalization step has to be performed. Given there are two relational tables δ_j and δ_k with Γ_j being an ordered set of columns of δ_j and Γ_k an ordered set of columns in δ_k. Then $(\delta_k, \Gamma_k) \leftarrow (\delta_j, \Gamma_j)$ denotes the fact, that all columns in Γ_j define a foreign key dependency, referencing the columns in Γ_k . The normalization step is then performed applying the following function:

$$normalize_{M9}(\delta_j, \Gamma_j) = \begin{cases} (\delta_k, \Gamma_k) \text{ if } (\delta_k, \Gamma_k) \leftarrow (\delta_j, \Gamma_j) \\ (\delta_j, \Gamma_j) \text{ otherwise} \end{cases} \tag{21}$$

To track down extensional redundancies for a given set of view definitions $V \subset \mathcal{V}$ and the multiset union $\biguplus$, a multiset of pairs M_{M9} is generated as follows:

$$M_{M9} = \biguplus_{v_i \in V} \left\{\!\!\left\{ (\overleftarrow{\delta_{v_i}}, \overleftarrow{\Gamma_{v_i,q}}) \;\middle|\; \begin{array}{l} p \in quads(v_i) \wedge \\ q \in \left(\begin{array}{l} subject(p) \cup \\ predicate(p) \cup \\ object(p) \end{array} \right) \cap \mathcal{Q} \wedge \\ rdf\_term\_type(term\_constructor(q)) = \texttt{uri} \wedge \\ \Gamma_{v_i,q} = cols(term\_constructor(q)) \wedge \left|\Gamma_{v_i,q}\right| > 0 \wedge \\ (\overleftarrow{\delta_{v_i}}, \overleftarrow{\Gamma_{v_i,q}}) = normalize_{M9}(rel\_table(v_i), \Gamma_{v_i,q}) \end{array} \right\}\!\!\right\} \tag{22}$$

M_{M9} contains all available combinations of (logical) tables and their referenced columns, as well as the corresponding counts. The count of a pair η can be retrieved via $M_{M9}(\eta)$.

Given the (multiset) cardinality $n = \|M_{M9}\|$ and the quality score function $f_9 : \mathbb{P}(\mathcal{V}) \rightarrow \mathbb{R}^n$, the Extensional Conciseness is calculated as follows:

$$f_9(V) = \left(\frac{1}{M_{M9}(\eta)} \right)_{\eta \in M_{M9}} \tag{23}$$

Again, the quality score function f_9 returns a tuple of quality scores with one value for each term constructor that reference at least one relational column. Given the example view definitions above, M_{M9} would look like this:

```
(dept, {?id}), # from redundant1: uri(ex:dept, ?id)
(dept, {?id})  # from redundant2: uri(ex:department, ?dept_id)
```

Thus, for both of these pairs $\frac{1}{M_{M9}(\eta)}$ would yield $\frac{1}{2}$.

For a more practical reporting of the actual error cause, the term constructors, the corresponding quad variables and the view definitions they stem from would have to be stored as well in Metric 9. This is omitted here for brevity.

With regards to the next metric, it has to be noted, that even though the terms *quad pattern* and *quad* are used, as introduced in the description of the Sparqlification Mapping Language in Section 2.1, the definition is formulated for an assessment of duplicate *triples*. This restriction was also made for brevity and Metric 10 can easily be extended to also work on graphs of triples.

Metric 10 (No Duplicate Statements) *The metric assessing statement-level redundancy is a view metric. RDF statements are considered redundant if there are multiple occurrences having the same subject, predicate and object. In SML view definitions this can occur due to multiple SML quads that are equal. The* No Duplicate Statements *metric thus regards the SML quads of all possible combinations of view definitions $\mathbb{P}(V)$ with $V \subset \mathcal{V}$. The power set $\mathbb{P}(V)$ is considered to precisely determine, which SML quad combinations cause statement duplications.*

For every subset $V_{\subseteq} \in \mathbb{P}(V)$ all quads, referencing at least one relational column, are normalized, replacing quad variables with their anonymized term constructors:

$$\widetilde{P} = \bigcup_{v_i \in V_{\subseteq}} \left\{ \begin{pmatrix} normalize_{M10}(subject(p)), \\ normalize_{M10}(predicate(p)), \\ normalize_{M10}(object(p)) \end{pmatrix} \middle| \begin{array}{l} p \in quads(v_i) \wedge \\ q \in \begin{pmatrix} subject(p) \cup \\ predicate(p) \cup \\ object(p) \end{pmatrix} \cap Q \wedge \\ |cols(term\_constructor(q))| > 0 \end{array} \right\} \tag{24}$$

with $normalize_{M10}$ being defined for a node input $n \in \mathcal{N}$ as follows:

$$normalize_{M10}(n) = \begin{cases} anonymize(term\_constructor(n)) & \text{if } n \in Q \\ n & \text{otherwise} \end{cases} \tag{25}$$

The anonymize function, applied to the term constructor, replaces all referenced relational column names with a fixed dummy column, e.g. ?col. Applying this normalization to the quad of the view definition

```
Create View normalization_example As
  Construct {
    ?empl ex:worksAt ?dept .
  }
  With
    ?empl = uri(ex:person, ?id)
    ?dept = uri(ex:dept, '/', ?dept)
  From
    empl
```

will result in a normalized quad

```
(uri(ex:person, ?col), ex:worksAt, uri(ex:dept, '/', ?col))
```

With $\widetilde{P}$ *containing these normalized tuples, one can easily keep track of all available quads and their anonymized term constructors. To also preserve the link to the actual (logical) table of these normalized quads, a mapping has to be established, e.g.*

$$MAP_{V_\subseteq} = \bigcup_{v_i \in V_\subseteq} \left\{ (\widetilde{p}, \delta) \middle| \begin{array}{l} p \in quads(v_i) \wedge \\ \widetilde{p} = \begin{pmatrix} normalize_{M10}(subject(p)), \\ normalize_{M10}(predicate(p)), \\ normalize_{M10}(object(p)) \end{pmatrix} \wedge \\ \delta = rel\_table(v_i) \end{array} \right\} \tag{26}$$

Having the set of normalized quads and the mapping to the underlying relational tables, a generic approach can be followed to detect the introduction of duplicate triples. Below, expressions like $m_{[i]}$ *refer to the ith entry of the tuple m. For quads* $p_i \in (\mathcal{R} \cup \mathcal{Q}) \times (\mathcal{R} \cup \mathcal{Q}) \times (\mathcal{R} \cup \mathcal{L} \cup \mathcal{Q})$ *and their corresponding normalized quads* $\widetilde{p_i} \in \widetilde{P}$ $(0 \leq i < |\widetilde{P}|)$ *the following SQL count queries are generated and executed:*

a) $count_{each}^{(\widetilde{p_i}, m_j)}$*: for every map entry* $m_j \in MAP_{V_\subseteq}$ *with* $m_{j[0]} = \widetilde{p_i}$ *the count of distinct entries of the table* $\delta_j = m_{j[1]}$ *is queried, where* δ_j *is projected to the relational attributes referenced in the original term constructors of* p_i

b) $count_{union}^{(\widetilde{p_i})}$*: for all map entries* $m_k \in M_{\widetilde{p_i}}$ *with* $M_{\widetilde{p_i}} = \left\{ m \,|\, m \in MAP_{V_\subseteq} \wedge m_{[0]} = \widetilde{p_i} \right\}$ *the count of distinct values of* the union *of all tables* $\left\{ \delta \,|\, m \in M_{\widetilde{p_i}} \wedge \delta = m_{[1]} \right\}$ *(each projected to the relational attributes referenced in in the original term constructors of* p_i*) is queried*

Given a considered set of view definitions $V_\subseteq \in \mathbb{P}(V)$*, a quad* $p \in (\mathcal{R} \cup \mathcal{Q}) \times (\mathcal{R} \cup \mathcal{Q}) \times (\mathcal{R} \cup \mathcal{L} \cup \mathcal{Q})$ *and its normalized quad representation* $\widetilde{p} \in \widetilde{P}$*, the quality score with regards to the* No Duplicate Statements *metric is determined by the function* $\hat{f}_{10} : (\mathcal{R} \cup \mathcal{Q}) \times (\mathcal{R} \cup \mathcal{Q}) \times (\mathcal{R} \cup \mathcal{L} \cup \mathcal{Q}) \to \mathbb{R}$*:*

$$\hat{f}_{10}(p) = \frac{count_{union}^{(\widetilde{p})}}{\sum_{m_j \in \left\{ m | m \in MAP_{V_\subseteq} \wedge m_{[0]} = \widetilde{p} \right\}} count_{each}^{(\widetilde{p}, m_j)}} \tag{27}$$

An example, presenting the different steps of Metric 10 is given in Figure 15. The complexity of this metric had to be introduced due to the fact, that in a prac-

relational tables

PERS

pers_id	given_name	surname
1	Klaus	Weichelt
2	Olaf	Schubert
3	Alex	Köhring

EMPL

id	full_name
3	Jochen Barkas
4	Bert Stephan
5	Stephan Gräber

view definitions

```
Create View view_01 As
 Construct {
   ?p a ex:Person .
 }
 With
   ?p = uri(ex:person, ?pers_id)
 From
   PERS
```

```
Create View view_02 As
 Construct {
   ?e a ex:Person .
 }
 With
   ?e = uri(ex:person, ?id)
 From
   EMPL
```

set of normalized quads

```
    ( uri(ex:person, ?col) , rdf:type , ex:Person )
```

$count^{(\widetilde{p})}_{union}$ query

```
# will return 5
SELECT count(*) AS count FROM (
   SELECT DISTINCT col1 FROM (
      (
         # ?pers_id originally referenced in the term
         # constructor of ?pers in view_01
         SELECT pers_id AS col1 FROM PERS
         WHERE pers_id IS NOT NULL
      ) UNION (
         # ?id originally referenced in the term
         # constructor of ?empl in view_02
         SELECT id AS col1 FROM EMPL WHERE id IS NOT NULL
      ) AS unified
   ) AS dstnct
```

$count^{(\widetilde{p},m_j)}_{each}$ query

```
# (~p, m1): will return 3
SELECT count(*) AS count
FROM (
    SELECT DISTINCT pers_id
    FROM PERS
    WHERE pers_id IS NOT NULL
) AS pers_ids
```

```
# (~p, m2): will return 3
SELECT count(*) AS count
FROM (
    SELECT DISTINCT id
    FROM EMPL
    WHERE id IS NOT NULL
) AS empl_ids
```

score

$$\frac{count^{(\widetilde{p})}_{union}}{\sum count^{(\widetilde{p},m_j)}_{each}} = \frac{5}{3+3} = \frac{5}{6} \longrightarrow \text{there are duplicates (<http://ex.org/pers3>)}$$

Fig. 15: Example demonstrating the approach of the *No Duplicate Statements* metric

tical assessment run, depending on which tools or libraries are used to provide access to the dataset, duplicates may already be eliminated when the data are loaded. In such cases, duplicates are only detectable using more complex algorithms like the one proposed above. Nonetheless, the presented approach does not cover all possible ways to introduce duplicate statements via RDB2RDF mappings. Redundant statements might for example be created when different term constructors are involved, that generate the same URIs or literals, e.g. `plainLiteral(?given_name, ' ', ?surname)` and `plainLiteral(?full_name)`. Moreover, it has to be noted, that there might be cases, where the SQL union statement of $count^{(\widetilde{p})}_{union}$ fails due to incompatible datatypes.

Consistency Since an erroneous view definition of an RDB2RDF mapping may affect a lot of generated RDF statements it is of special importance to check whether the created data is *consistent*. An inconsistent dataset may be useless or even harm applications using its data. Besides checking basic consistence aspects covered by the Basic Ontology Conformance metric, other, more subtle issues like the homogeneous usage of datatypes and possible defects of classes and properties are assessed. Moreover, the No Ambiguous Mappings metric checks whether multiple database objects may get mapped to one single RDF resource. Finally, metrics to detect known error patterns and the generation of inconsistent URIs in case of referencing foreign key identifiers, are provided. With these metrics a wide range of errors should be tracked down, increasing the quality of an RDB2RDF mapping with regards to its consistency.

The first metric presented, covers the introduction of inconsistencies with respect to the ontologies that are used. Such errors may occur, when using classes or properties without looking up possible restrictions on them, e.g. when using the property `foaf:interest` with the literal value `"RDB2RDF"`, even though `foaf:interest` is an object property. Parts of this metric were proposed in the survey of Zaveri et al. [67].

Metric 11 (Basic Ontology Conformance) *The metric checking the conformance of different ontology consistency aspects is a dataset metric. Such aspects are*

- *Correct Datatype Property Value: reports property value violations of datatype properties (e.g. when non-literal values are assigned via a datatype property)*
- *Correct Object Property Values: reports property value violations of object properties (e.g. when literal values are assigned via an object property)*
- *Disjoint Classes Conformance: reports violations of class disjointness axioms*
- *Valid Range: reports improper values w.r.t. a defined range*

Accordingly, given a Dataset $D \in \mathcal{D}$ and a set of used vocabularies $\mathcal{D}_{voc} \subset \mathcal{D}$, the actual quality score is assigned per statement $t \in D$ by the function $\hat{f} : \mathcal{T} \rightarrow \mathbb{R}$ as follows:

$$\hat{f}(t) = \begin{cases} 0 \textit{ if a violation was found in } D \cup (\bigcup \mathcal{D}_{voc}) \\ 1 \textit{ otherwise} \end{cases} \tag{28}$$

This metric was introduced rather informal since there are several tools and reasoners that may be utilized without having to know the internals. Apart from this, these generic issues are already discussed in the literature, e.g. [83].

The Basic Ontology Conformance metric here was designed as a dataset metric. This has the advantage of working and reasoning on the real data, but also has the disadvantage that it may not be practically computable at all, if the dataset is too big and the machine running the assessment has not enough memory. A different approach would be to run the metric just considering the view definitions and the underlying database schema. Based on the schema definitions, dummy values could be created for the relational columns referenced in view definitions. This would result in a very small dataset, that only holds surrogate values, which can be assessed much faster and with less memory demand. Compared to the straight forward approach, described in the metric definition above, using dummy data based on the schema might even point to errors that can not be discovered using the real RDF data. This would be the case, if there is a view definition that would generate inconsistent triples, but the underlying logical table is empty. On the other hand, there are also cases, where violations can not be determined only considering the relational schema, e.g. when class disjointness statements were generated using relational data.

Besides this ontological consistence there should also be a consensus on which datatypes to use for property values. This is especially important when processing data in applications, e.g. when relying on a certain type for displaying or further processing steps. A concrete example of such an issue was reported in the paper introducing the *test-driven data quality methodology* [49]. There, data quality test cases yielded false positives because certain dates were in an unexpected format. In the RDB2RDF context inhomogeneous datatypes may occur if different view definitions use the same property but apply conflicting types to the properties' values. The Homogeneous Datatypes metric detecting such issues is defined as follows:

Metric 12 (Homogeneous Datatypes) *The metric assessing the homogeneity of the datatypes of property values, is a dataset metric. Given a dataset $D \in \mathcal{D}$ the following set is created to track occurrences of properties and their value types:*

$$MAP_{M12} = \bigcup_{t \in D} \left\{ (r, type) \middle| \begin{array}{l} r = predicate(t) \wedge object(t) \in \mathcal{L} \wedge \\ object(t) \text{ is of type 'type'} \end{array} \right\} \tag{29}$$

The function $\hat{f}_{12} : \mathcal{R} \rightarrow \mathbb{R}$ determining the quality score of a predicate $r \in \mathcal{R}$ is then defined as follows:

$$\hat{f}_{12}(r) = \begin{cases} 0 \text{ if } \left| \left\{ (r_{MAP}, type) \middle| (r_{MAP}, type) \in MAP_{M12} \wedge r = r_{MAP} \right\} \right| > 1 \\ 1 \text{ otherwise} \end{cases} \tag{30}$$

In the definition above, some details are omitted for brevity. First, it is left open, if plain literals should be considered. If so, their type would be just 'plain', allowing to find inhomogeneities with regards to the usage of plain and typed literals. Metric 12 also does not evaluate type hierarchies, e.g. as defined in the XML Schema specification [84]. Another detail left out in the definition, but implemented in R2RLint is the distinction between *outliers* and *type clashes*. Given the threshold θ, a type inhomogeneity is considered to be an outlier, if just a portion of all occurrences is affected, that is smaller than θ. So, if θ is 0.9 and 9% of the values of the considered property have

one type and 91% of the values are of a different type, the 9% are viewed as outliers. In case the portion is bigger, e.g. 11% a type clash would be reported.

It also has to be noted, that this metric does not evaluate possible `rdfs:range` definitions. Thus, a dataset could be consistent with respect to the datatype homogeneity of property values but inconsistent as far as the compliance with ontological restrictions is concerned, which is covered by Metric 11 (Basic Ontology Conformance). Moreover the comments made for Metric 11 also hold for the Homogeneous Datatypes metric, i.e. that inhomogeneities could also be detected considering the view definitions. Even though this would reduce the amount of data to process, there are again cases where the introduction of homogeneity violations depends on the relational data, e.g. when variable datatypes are used, as in `typedLiteral(?value, ?type)`. Besides this, the distinction between outliers and type clashes could not be made without evaluating the generated RDF data.

In contrast to these literal inconsistencies, the next metric covers a more formal inconsistency, concerning classes and properties. This metric assesses the usage of classes and properties that are declared to be deprecated and thus should not be used because they might be removed from the corresponding vocabulary and thus not be supported in the future:

Metric 13 (No Deprecated Classes or Properties) *The metric detecting if deprecated classes or properties are used, is a dataset metric. Given a dataset $D \in \mathcal{D}$ and a set of vocabularies $\mathcal{D}_{voc} \subset \mathcal{D}$ that are used within the dataset D, the* No Deprecated Classes or Properties *metric looks for explicit statements $t_{dep} \in T_{dep}$ with*

$$T_{dep} = \left\{ t_{dep} \left| \begin{array}{l} t_{dep} \in D \cup (\bigcup \mathcal{D}_{voc}) \ \wedge \ predicate(t_{dep}) = \texttt{rdf:type} \wedge \\ \left(\begin{array}{l} object(t_{dep}) = \texttt{owl:DeprecatedClass} \vee \\ object(t_{dep}) = \texttt{owl:DeprecatedProperty} \end{array} \right) \end{array} \right. \right\} \tag{31}$$

that express deprecation axioms using the classes `owl:DeprecatedClass` *or* `owl:DeprecatedProperty`*.*

Given the sets CLASSES $\subset \mathcal{R}$ containing all defined class resources in D, and PROPERTIES $\subset \mathcal{R}$ containing all defined property resources in D, the quality score of a class $r_c \in$ CLASSES with respect to the No Deprecated Classes or Properties metric is defined by the function $\hat{f}_{13_C} : \mathcal{R} \to \mathbb{R}$ as follows:

$$\hat{f}_{13_C}(r_c) = \begin{cases} 0 \text{ if } r_c \in subject(T_{dep}) \\ 1 \text{ otherwise} \end{cases} \tag{32}$$

Analogously, the quality score of a property $r_p \in$ PROPERTIES is defined by the function $\hat{f}_{13_P} : \mathcal{R} \to \mathbb{R}$ as

$$\hat{f}_{13_P}(r_p) = \begin{cases} 0 \text{ if } r_p \in subject(T_{dep}) \\ 1 \text{ otherwise} \end{cases} \tag{33}$$

The sets *CLASSES* and *PROPERTIES* are only introduced informally, since their actual (sound and complete) acquisition is not within the scope of this report and can be delegated to several reasoner tools or libraries to implement this metric. These sets are also assumed to exist for the introduction of the following metrics.

Besides this, it has to be noted, that such deprecation statements refer to the identifier of a resource and not to the resource itself [85]. Thus, as an example, deprecation can not be inferred for a class that is linked via `owl:equivalentClass` to another class, which has a deprecated identifier.

Even though, the authors that first drew attention to the following issue showed that most of the concrete problems occur rather rarely [61], the corresponding metric was included. Besides the intention of having a means to detect known, but hard to find errors, this blacklist-based metric should also serve as place for further fixed error patterns that may come up in the future.

Metric 14 (No Bogus Inverse-functional Properties) *The metric looking for* bogus inverse-functional properties *as reported in [61], is a triple metric. To detect such deficiencies a black list proposed in [61] is used. This black list contains values of inverse-functional properties that stem from not validated inputs and do in fact not identify a resource uniquely. An example of such a bogus inverse-functional property value is a SHA1 hashed empty e-mail address string 'mailto:', used in connection with the* `foaf:mbox_sha1sum` *property. The actual literal value of the empty, hashed e-mail address is "08445a31a78661b5c746feff39a9db6e4e2cc5cf" which is the same for all empty input data, violating the inverse functional nature of* `foaf:mbox_sha1sum`*. The whole black list $T_{bogus} \subset \mathcal{T}$ is given in Appendix A.1.*

A statement $t \in D$ is considered a violation with respect to the No Bogus Inverse-functional Properties metric if there is a triple $t_b \in T_{bogus}$ with $predicate(t) = predicate(t_b)$ and $object(t) = object(t_b)$. The quality score function $f_{14} : \mathcal{T} \rightarrow \mathbb{R}$ is then given as follows:

$$f_{14}(t) = \begin{cases} 0 \text{ if } \exists t_b \begin{pmatrix} t_b \in T_{bogus} \wedge \\ predicate(t) = predicate(t_b) \wedge \\ object(t) = object(t_b) \end{pmatrix} \\ 1 \text{ otherwise} \end{cases} \tag{34}$$

To be able to utilize datasets participating in the Web of Data in a dependable manner, there must be clear authorities with regards to the definition of established vocabularies and ontologies shared and reused amongst the different data endpoints. Since the definition authorities in the Semantic Web are clearly determined by the domain names used in the corresponding URIs and the domain owners, there should be no vocabulary axioms, set up in datasets not belonging to the URI prefix of the so defined vocabulary. Accordingly, a definition like `rdfs:label rdf:type rdfs:Class .`, in any dataset not provided by the authority of the RDF Schema prefix, is considered as ontology hijacking. Thus, to avoid inconsistencies based on concurrent and conflicting statements published by different datasets, the following metric checks if there are any definitions made for foreign, i.e. non-local, resources. This metric was also part of the collected metrics in Zaveri et al. [67].

Metric 15 (No Ontology Hijacking) *The metric assessing if there are any re-definitions of parts of vocabularies not being under the authority of the owner of a considered dataset $D \in \mathcal{D}$, is a dataset metric. Given a set $\mathcal{D}_{voc} \subset \mathcal{D}$ of known vocabularies, a triple $t \in D$ is a violation with respect to the* No Ontology Hijacking *metric, if for*

any of the vocabularies $D_{voc} \in \mathcal{D}_{voc}$, $subject(t) \in subject(D_{voc})$. *In case the URI of* $subject(t)$ *does not share the local prefix(es) of D, but* $subject(t) \notin subject(D_{voc})$, *t is considered as* 'bad smell'. *With* R_{local} *being defined as in Metric 5, the quality score of a triple* $t \in D$ *is determined by the function* $\hat{f}_{15} : \mathcal{T} \rightarrow \mathbb{R}$*:*

$$\hat{f}_{15}(t) = \begin{cases} 0 & \text{if } \exists D_{voc} \left(D_{voc} \in \mathcal{D}_{voc} \wedge subject(t) \in subject(D_{voc}) \right) \\ 0.5 & \text{if } \begin{array}{l} subject(t) \notin R_{local} \wedge \\ \nexists D_{voc} \left(D_{voc} \in \mathcal{D}_{voc} \wedge subject(t) \in subject(D_{voc}) \right) \end{array} \\ 1 & \text{otherwise} \end{cases} \tag{35}$$

The *No Ontology Hijacking* metric, as defined here, is rather strict since it does not allow any re-definitions, even if they are consistent with the corresponding vocabulary and merely serve as a local cache. Consequently this metric could be weakened, allowing re-definitions of external vocabularies if these are identical copies of the original axioms.

Another issue of RDB2RDF transformations is the silent loss of information due to ambiguous mappings. In this case, multiple different relational database objects are mapped to the same RDF resource. Even though, this metric was proposed by Zaveri et al. [67] as conciseness metric, it is used in the context of the consistency dimension here. This is motivated by the fact, that the transformation process from relational data to RDF data is considered, and normally the *unique name assumption* holds in relational database systems. Accordingly, it would be an inconsistent state, if two relational database objects have the same identifier. Thus, the introduction of ambiguities is viewed as a hard consistency error, rather than a problem of redundant opportunities of an interpretation of a given identifier, as done by Zaveri et al. [67]. In the RDB2RDF context, such errors may occur when two term constructors generate the same URIs based on different database objects. This happens e.g. when existing variable definitions are copied and pasted, without adapting the corresponding term constructors properly.

Metric 16 (No Ambiguous Mappings) *The metric assessing if multiple different database objects were mapped to one single RDF resource, is a view metric. Instead of actually checking the database entries in connection with the views definitions' term constructors, this view metric considers settings in the view definitions that* may *lead to ambiguous RDF resources. Such settings are given, if the same term constructor is used referring to possibly different attributes of different relations, as in the following example:*

```
Create View ambiguous1 As
  Construct {
    ?a a ex:Employee .
  }
  With
    ?a = uri(ex:person, ?pid)  # ambiguous
  From
    pers

Create View ambiguous2 As
  Construct {
```

```
      ?b a ex:Person .
    }
    With
      ?b = uri(ex:person, ?eid)  # ambiguous
    From
      empl
```

An exception of this simple rule is given in cases where the considered attribute of one relation is the parent foreign key attribute of another considered relation (and vice versa). Thus, assuming there is a foreign key dependency between the attribute `empl.dept_id` *and* `dept.id` *(e.g. employee relation pointing to the id of a department an employee works in), the term constructors marked with a comment in the following view definitions are not ambiguous, since both refer to the same database object:*

```
Create View not_ambiguous1 As
  Construct {
    ?e a ex:Employee .
    ?d a ex:Department .
    ?e ex:worksIn ?d .
  }
  With
    ?e = uri(ex:empl, ?id)
    ?d = uri(ex:dept, ?dept_id)  # not ambiguous
  From
    empl

Create View not_ambiguous2 As
  Construct {
    ?d ex:name ?n .
  }
  With
    ?d = uri(ex:dept, ?id)  # not ambiguous
    ?n = plainLiteral(?name)
  From
    dept
```

To find ambiguous mappings, an approach similar to Metric 10 is followed. First of all, the anonymize function, introduced there, is reused. Applying anonymize to any term constructor tc ∈ TC, available in the given set of view definitions V, with

$$TC = \bigcup_{v_i \in V} \left\{ tc \;\middle|\; \begin{array}{l} q \in \left(\begin{array}{c} subject(quads(v_i)) \cup \\ predicate(quads(v_i)) \cup \\ object(quads(v_i)) \end{array} \right) \cap Q \wedge \\ tc = term\_constructor(q) \end{array} \right\} \tag{36}$$

it replaces the relational columns referenced in each term constructor tc with a dummy variable, e.g. "?col". Besides this, the normalization function normalize$_{M9}$ from Metric 9 is utilized. Given the foreign key dependency $(\delta_j, \Gamma_j) \leftarrow (\delta_k, \Gamma_k)$, this normalization step replaces the referencing table and foreign keys with their actual dereferenced target. Accordingly normalize$_{M9}(\delta_k, \Gamma_k)$ would yield (δ_j, Γ_j). To keep track of the (deref-

erenced) relational tables and the (dereferenced) columns the original term constructor referred to, the set MAP_{M16} is used, given as follows:

$$MAP_{M16} = \bigcup_{v_i \in V} \left\{ (\widetilde{tc}, \overleftarrow{\delta}, \overleftarrow{\Gamma}) \;\middle|\; \begin{array}{c} q \in \left(\begin{array}{c} subject(quads(v_i)) \cup \\ predicate(quads(v_i)) \cup \\ object(quads(v_i)) \end{array} \right) \cap Q \wedge \\ tc = term\_constructor(q) \wedge \\ \widetilde{tc} = anonymize(tc) \wedge \\ (\overleftarrow{\delta}, \overleftarrow{\Gamma}) = normalize_{M9}(rel\_table(v_i), cols(tc)) \end{array} \right\} \tag{37}$$

With $\mathcal{TC}$ being the set of all valid term constructors, the function $\hat{f}_{16} : \mathcal{TC} \rightarrow \mathbb{R}$ to determine the quality score of a term constructor $tc \in TC$ is then defined as follows:

$$\hat{f}_{16}(tc) = \begin{cases} 0 \; if \left| \left\{ (\widetilde{tc}, \overleftarrow{\delta}, \overleftarrow{\Gamma}) \;\middle|\; \begin{array}{l} \widetilde{tc} = anonymize(tc) \wedge \\ (\widetilde{tc}, \overleftarrow{\delta}, \overleftarrow{\Gamma}) \in MAP_{M16} \end{array} \right\} \right| > 1 \\ 1 \; otherwise \end{cases} \tag{38}$$

Accordingly, if there are at least two different entries in MAP_{M16} having the same anonymized term constructor, this may result in an ambiguous mapping and is thus considered a violation.

Besides this process-oriented assessment approach to find ambiguities introduced by an RDB2RDF mapping, the problem can also be tackled on an ontological basis. A weakness of the No Ambiguous Mappings metric is that it does not cover ambiguities introduced by different term constructors, i.e. when $\widetilde{tc_i} \neq \widetilde{tc_j}$ for two term constructors tc_i and tc_j holds. This can be the case, if e.g. tc_i builds URIs using complete URL strings retrieved from the underlying relational table and tc_j builds URIs only inserting certain strings gotten from the relational table into a URI template. Considering the two database values *'http://ex.org/ont/ambiguous'* and *'ambiguous'*, they will both result in the URI `<http://ex.org/ont/ambiguous>` when applied to the different term constructors `url(?val)` and `url(ex:ont, ?val)`, respectively. Thus, the following metric is introduced detecting ambiguities from an ontological perspective.

Metric 17 (No Resource Name Clashes) *The metric assessing if resource identifiers are used multiple times by different resources, is a dataset metric. To determine such resource name clashes in a dataset $D \in \mathcal{D}$, the following sets are used: CLASSES, PROPERTIES and INDIVIDUALS, where the first two sets are defined as above and INDIVIDUALS* $\subset$

$\mathcal{R}$ holds all individuals defined in D. The function $\hat{f}_{17} : \mathcal{R} \rightarrow \mathbb{R}$, returning the quality score for a given resource $r \in \mathcal{R}$, is defined as follows:

$$\hat{f}_{17}(r) = \begin{cases} 0 \text{ if } r \in \textit{CLASSES} \ \wedge \ (s,\ r,\ o) \ \in D & (a_1) \\ 0 \text{ if } r \in \textit{CLASSES} \wedge (r,\ p,\ o) \in D & (a_2) \\ \quad \wedge\ p \text{ is a datatype or object property} & \\ 0 \text{ if } r \in \textit{CLASSES} \wedge (s,\ p,\ r) \in D & (a_3) \\ \quad \wedge\ p \text{ is a datatype or object property} & \\ 0 \text{ if } r \in \textit{PROPERTIES} \wedge (r,\ p,\ o) \in D & (b_1) \\ \quad \wedge\ p \text{ is a datatype or object property} & \\ 0 \text{ if } r \in \textit{PROPERTIES} \wedge (s,\ p,\ r) \in D & (b_2) \\ \quad \wedge\ p \text{ is a datatype or object property} & \\ 0 \text{ if } r \in \textit{INDIVIDUALS} \ \wedge \ (s,\ r,\ o) \ \in D & (c_1) \\ 1 \text{ otherwise} & \end{cases} \tag{39}$$

Thus, to detect resource name clashes, this metric looks for invalid combinations of occurrences of resources in triples of a dataset D. As shown e.g. in the cases (a_1) to (a_3), classes should not appear as predicates of triples or in connection with datatype or object properties.

Since this metric assesses a dataset with regards to ontological violations, it could be consolidated with Metric 11 (Basic Ontology Conformance). But due to the concrete error pattern it covers (copy paste errors), Metric 17 was designed as separate metric to provide a higher flexibility with regards to the configuration of an assessment run.

The last metric presented for the consistency dimension is assessing the consistent mapping of relational objects being involved in a foreign key dependency. Since the primary key values of such objects appear in two relational tables – the referencing and the referenced one – it has to be taken care of a consistent mapping of the corresponding foreign and primary key columns.

Metric 18 (Consistent Foreign Key Resource Identifiers) *The metric assessing, if referenced relational foreign key columns are mapped consistently, is a view metric. Given there are two term constructors tc_1 and tc_2, each referring to exactly one relational column γ_1 and γ_2 respectively, where γ_1 is defined in relation δ_1 and γ_2 is defined in relation δ_2. It is further assumed, that there are no other term constructors, referring to the columns γ_1 and γ_2. In addition to this, a foreign key dependency is assumed to exist between both columns $\delta_1.\gamma_1 \leftarrow \delta_2.\gamma_2$ with $\delta_1.\gamma_1$ being the referenced (or parent) part and $\delta_1.\gamma_1$ the referencing (or child) one. The relational foreign key column γ_2 is not mapped consistently if tc_1 and tc_2 construct different URIs. As a consequence such a mapping would create two different resource identifiers for one single database object.*

Let $V \subset \mathcal{V}$ be a set of view definitions, δ_i, δ_j be relational tables, each referenced in one of the view definitions of V, and Γ_i, Γ_j ordered sets of columns defined in δ_i and δ_j, respectively. Let $(\delta_i, \Gamma_i) \leftarrow (\delta_j, \Gamma_j)$ further denote the fact, that all columns in Γ_j define a foreign key dependency, referencing the columns in Γ_i. To find inconsistent foreign key

identifiers, a normalization function for term constructors is defined. Here 'normalized' *means that in case all columns Γ_j, referred to in a term constructor tc, define a referencing foreign key dependency $(\delta_i, \Gamma_i) \leftarrow (\delta_j, \Gamma_j)$ to another set of columns Γ_i, a new term constructor will be created, where all occurrences of referencing columns in Γ_j are replaced with the corresponding referenced column of Γ_i. This replacement is denoted by $tc_{|\Gamma_i \leftarrow \Gamma_j}$. The second part of the normalization is the replacement of the referencing table δ_j with the referenced table δ_i. The function performing these transformations on a relational table δ_j and a term constructor tc is introduced as follows:*

$$normalize_{M18}(\delta_j,\, tc) = \begin{cases} (\delta_i,\, \Gamma_i,\, tc_{|\Gamma_i \leftarrow \Gamma_j}) \; if \; \begin{array}{l} \Gamma_j \,=\, cols(tc) \,\wedge \\ (\delta_i, \Gamma_i) \leftarrow (\delta_j, \Gamma_j) \end{array} \\ \\ (\delta_j, cols(tc),\, tc) \; otherwise \end{cases} \tag{40}$$

To keep track of term constructors that refer to columns that are the target of some referencing foreign key columns, a set PAR is introduced as follows:

$$PAR = \bigcup_{v_i \in V} \left\{ (\delta, \Gamma, tc) \;\middle|\; \exists (tc', v') \begin{pmatrix} q \in \begin{pmatrix} subject(quad(v_i)) \cup \\ predicate(quad(v_i)) \cup \\ object(quad(v_i)) \end{pmatrix} \cap\, Q \,\wedge \\ tc \,=\, term\_constructor(q) \,\wedge \\ \delta \,=\, rel\_table(v_i) \;\wedge\; \Gamma \,=\, cols(tc) \,\wedge \\ \begin{pmatrix} v' \in V \,\wedge \\ q' \in \begin{pmatrix} subject(quad(v')) \cup \\ predicate(quad(v')) \cup \\ object(quad(v')) \end{pmatrix} \cap Q \,\wedge \\ tc' = term\_constructor(q') \,\wedge \\ \delta' = rel\_table(v') \,\wedge \\ normalize_{M18}(\delta', tc') = (\delta, \Gamma, tc) \end{pmatrix} \end{pmatrix} \right\} \tag{41}$$

Given the set $\mathcal{TC}$ of valid term constructors, the set of valid relational tables Δ, the function $\hat{f}_{18} : \mathcal{TC} \times \Delta \rightarrow \mathbb{R}$ assessing, whether the identifier generated by a term constructor tc, defined on a relational table δ, is consistent, is introduced as follows:

$$\hat{f}_{18}(tc, \delta) = \begin{cases} \begin{array}{ll} \quad rdf\_term\_type(tc) = \texttt{uri} \,\wedge & (a) \\ \quad normalize_{M18}(\delta,\, tc) \,=\, (\overleftarrow{\delta},\, \overleftarrow{\Gamma},\, \overleftarrow{tc}) \,\wedge & \\ 0 \; if \; (\delta, cols(tc), tc) \,\neq\, (\overleftarrow{\delta},\, \overleftarrow{\Gamma},\, \overleftarrow{tc}) \,\wedge & (b) \\ \quad \exists e \left(e \,\in\, PAR \,\wedge\, e_{[0]} \,=\, \overleftarrow{\delta} \,\wedge\, e_{[1]} \,=\, \overleftarrow{\Gamma} \right) \,\wedge & (c) \\ \quad \nexists e' \left(e' \,\in\, PAR \,\wedge\, e'_{[0]} \,=\, \overleftarrow{\delta} \,\wedge\, e'_{[1]} \,=\, \overleftarrow{\Gamma} \,\wedge\, e_{[2]} \,=\, \overleftarrow{tc} \right) & (d) \\ 1 \; otherwise & \end{array} \end{cases} \tag{42}$$

Accordingly,

(a) a `uri` *term constructor tc*
(b) with referencing foreign key columns

is considered inconsistent with regards to the Consistent Foreign Key Resource Identifiers metric, if

(c) the referenced *table and columns are used by at least one term constructor,*
(d) but only by term constructors different from the referencing one, i.e. there is no entry in PAR for which also holds that its term constructor $tc' = e_{[2]}$ *equals* $\overleftarrow{tc}$.

Since it is checked, if tc contains referencing foreign key columns, only those 'referencing term constructors' *are assumed to violate the consistency, not the term constructors holding the referenced columns. An exmple showing the main evaluation steps is given in Figure 16.*

Interlinking To build a *web* of data, the linking between different datasets is of crucial importance. This is also reflected in the Linked Data principles [80] and guidelines [86]. Thus, to provide a high quality RDB2RDF mapping, an adequate portion of interlinks should be contained. This is assessed by the following metric.

Metric 19 (External Same-as Links) *The metric assessing the amount of statements expressing that a local and an external identifier refer to the same resource, is a dataset metric. Given a dataset* $D \in \mathcal{D}$*, a triple* $t \in D$ *is considered as* external same-as link *if* $subject(t)$ *is a local resource,* $predicate(t) =$ `owl:sameAs` *and* $object(t)$ *is a non-local (i.e. external) resource. The same holds, if subject and object are are used conversely, i.e. if the subject is an external resource connected to a local resource via* `owl:sameAs`*. The number of external same-as links of D is expressed with* $|D|_{ext\_same\_as}$*. The quality score function* $f_{19} : \mathcal{D} \rightarrow \mathbb{R}$ *is defined as*

$$f_{19}(D) = \frac{|D|_{ext\_same\_as}}{|D|} \tag{43}$$

Interoperability The interoperability dimension refers to the usage of well known formats and structures [67]. This mainly eases further processing as well as a data source independent use and may also strengthen the interlinking between different datasets. In the context of RDB2RDF mappings this can be achieved, if established terms and vocabularies are introduced. Even though, the former is a generalization of the latter, also covering the reuse of established individuals and the expression of relations to them, both aspects are assessed separately, to provide a higher flexibility. Both metrics were also proposed in Zaveri et al. [67] informally and are presented in the following.

Metric 20 (Term Reuse) *The metric assessing to which extend established terms are reused for the RDB2RDF mapping, is a dataset metric. To evaluate if a term is estab-*

SML view definitions

```
Create View referenced As
  Construct {
    ?d a ex:Department .
    ?d rdfs:label ?n .
  }
  With
    ?d = uri(ex:dept, ?id)
    ?n = plainLiteral(?name)
  From
    DEPT

Create View referencing As
  Construct {
    ?e a ex:Employee .
    ?e ex:worksIn ?d .
  }
  With
    ?e = uri(ex:person, ?id)
    ?d = uri(ex:department, ?dept_id)
  From
    EMPL  # has referencing foreign key EMPL.dept_id to DEPT.id
```

set of referenced tables & foreign key columns with corresponding term constructors

```
PAR = { ( DEPT, {id}, uri(ex:dept, ?id) ) }
```

condition (a)

$$rdf\_term\_type(\texttt{uri(ex:department, ?dept\_id)}) = \texttt{uri} \checkmark$$

condition (b)

$$normalize_{M18}(\texttt{EMPL}, \texttt{uri(ex:department, ?dept\_id)}) = (\texttt{DEPT}, \{\texttt{id}\}, \texttt{uri(ex:department, ?id)})$$

$$(\texttt{EMPL}, \{\texttt{dept\_id}\}, \texttt{uri(ex:department, ?dept\_id)}) \neq (\texttt{DEPT}, \{\texttt{id}\}, \texttt{uri(ex:department, ?id)}) \checkmark$$

condition (c)

$$\exists e(e \in PAR \wedge e_{[0]} = \texttt{DEPT} \wedge e_{[1]} = \{\texttt{id}\}) \checkmark$$

condition (d)

$$\nexists e'(e' \in PAR \wedge e'_{[0]} = \texttt{DEPT} \wedge e'_{[1]} = \{\texttt{id}\} \wedge e'_{[2]} = \texttt{uri(ex:department, ?dept\_id)}) \checkmark$$

$\longrightarrow$ *the URI created by* `uri(ex:department, ?dept_id)` *on table* `EMPL` *is inconsistent with respect to the Consistent Foreign Key Resource Identifiers metric*

Fig. 16: Example showing the main evaluation steps of the Consistent Foreign Key Resource Identifiers metric

lished, a set NS is used containing strings of well known and established namespaces. For a dataset $D \in \mathcal{D}$, the set of established resources can be defined as follows:

$$R_{est} = \bigcup_{t \in D} \left\{ r \;\middle|\; \begin{array}{l} r \in \begin{pmatrix} subject(t) \cup \\ predicate(t) \cup \\ object(t) \end{pmatrix} \cap \mathcal{R} \wedge \\ \\ \exists ns \begin{pmatrix} ns \in NS \wedge \\ \text{the string representation of } r \text{ starts with } ns \end{pmatrix} \wedge \\ \\ \text{using } r\text{'s URI as URL and requesting the corresponding} \\ \text{Web resource via } \texttt{HTTP GET}\text{, returns the HTTP response} \\ \text{code 200 after resolving any redirects} \end{array} \right\} \tag{44}$$

Accordingly, a resource is considered established if it shares an established namespace and is dereferenceable. The set of resources that are not established is denoted by $\overline{R_{est}}$ and defined as follows:

$$\overline{R_{est}} = \left(\bigcup_{t \in D} \left\{ r \;\middle|\; r \in \begin{pmatrix} subject(t) \cup \\ predicate(t) \cup \\ object(t) \end{pmatrix} \cap \mathcal{R} \right\} \right) \setminus R_{est} \tag{45}$$

With the cardinality expressions $|R_{est}|$ and $\left|\overline{R_{est}}\right|$, counting the distinct number of established and not established resources, the Term Reuse quality score of a dataset D is given by the quality score function $f_{20} : \mathcal{D} \to \mathbb{R}$:

$$f_{20}(D) = \frac{|R_{est}|}{|R_{est}| + \left|\overline{R_{est}}\right|} \tag{46}$$

To get the set *NS* of established namespaces, Metric 20 could use data of the *LODStats* [63] project or the *prefix.cc*[29] namespace lookup service.

The concept of term reuse can be further modified to only consider the reuse of vocabularies. Assessing the vocabulary reuse is of importance since established vocabularies are a major requirement for an interoperable Web of Data. Instead of loosing semantic relations between *different* vocabularies or having to state them explicitly, reusing *established* vocabularies allows a direct interoperation amongst different datasets. In contrast to term reuse, vocabulary reuse only refers to the usage of classes and properties stemming from established vocabularies or ontologies. The corresponding metric is defined as follows:

Metric 21 (Vocabulary Reuse) *The metric assessing the usage of established vocabularies within a dataset $D \in \mathcal{D}$ is a dataset metric. To evaluate the vocabulary reuse, the set of established vocabularies $\mathcal{D}_{est\_voc} \subset \mathcal{D}$ and the sets CLASSES and PROPERTIES*

[29] `http://prefix.cc/`

containing the classes and properties defined in D, are used. The set of established classes and properties $R_{est\_voc} \subset \mathcal{R}$ of a dataset D can then be defined as follows:

$$R_{est\_voc} = \bigcup_{t \in D} \left\{ r \left| \begin{array}{l} r \in \left(\begin{array}{l} subject(t) \cup \\ predicate(t) \cup \\ object(t) \end{array} \right) \cap (CLASSES \cup PROPERTIES) \wedge \\ r \in subject(\bigcup \mathcal{D}_{est\_voc}) \end{array} \right. \right\} \tag{47}$$

Accordingly, a class or predicate is considered established if it is defined or explained in an established vocabulary. The set $\overline{R_{est\_voc}} \subset \mathcal{R}$ of not established classes and properties is defined analogously:

$$\overline{R_{est\_voc}} = \bigcup_{t \in D} \left\{ r \left| \begin{array}{l} r \in \left(\begin{array}{l} subject(t) \cup \\ predicate(t) \cup \\ object(t) \end{array} \right) \cap (CLASSES \cup PROPERTIES) \wedge \\ r \notin subject(\bigcup \mathcal{D}_{est\_voc}) \end{array} \right. \right\} \tag{48}$$

Having the two cardinality expressions $\left|R_{est\_voc}\right|$ and $\left|\overline{R_{est\_voc}}\right|$ counting the distinct established and not established classes and properties, the Vocabulary Reuse quality score is given by the function $f_{21} : \mathcal{D} \rightarrow \mathbb{R}$:

$$f_{21}(D) = \frac{\left|R_{est\_voc}\right|}{\left|R_{est\_voc}\right| + \left|\overline{R_{est\_voc}}\right|} \tag{49}$$

To retrieve the set $\mathcal{D}_{est\_voc}$ of established vocabularies, the LODStats project or the *Linked Open Vocabularies* dataset[30] could be used.

Interpretability The interpretability dimension refers to the degree to which data are represented using a proper notation to support their machine processability. Accordingly, an RDB2RDF mapping should generate RDF data, that is meaningful with regards to the applied representation capabilities of the Resource Description Framework. This means, on the one hand, that if special notational constructs like RDF containers or RDF collections are used, these must conform with the underlying specifications. On the other hand, there should also be declarations, describing the semantics and ontological context of RDF resources appropriately. Just having a resource, e.g. `<http://ex.org/person23>` appearing in a statement like `<http://ex.org/person23> rdfs:label "Jochen Barkas".`, may be useless for any further inference and may also be hard to search using SPARQL. This is mainly because from a machine processing perspective it is not clear, whether `<http://ex.org/person23>` is a class, a property, or an instance of a certain class. Since datasets generated by RDB2RDF mappings tend to be rather simple as far as ontological structures are concerned [50], giving feedback about a lacking ontological context seems important. One such requirement for interpretable data, as pointed out by [61], is the provision of type information of RDF resources, which is assessed by the following metric.

[30] `http://lov.okfn.org/dataset/lov/`

Metric 22 (Typed Resources) *The metric assessing if local resources of a dataset $D \in \mathcal{D}$ are properly typed, is a dataset metric. With R_{local} being defined as in Metric 5, the quality score of an input resource $r \in R_{local}$ is determined by the function $\hat{f}_{22} : \mathcal{R} \rightarrow \mathbb{R}$:*

$$\hat{f}_{22}(r) = \begin{cases} 1 \text{ if } (r, \texttt{rdf:type}, o) \in D \wedge o \in \mathcal{R} \\ 1 \text{ if } (r, \texttt{rdf:type}, \texttt{rdfs:Class}) \in D \\ 1 \text{ if } (r, \texttt{rdf:type}, \texttt{owl:Class}) \in D \\ 1 \text{ if } (r, \texttt{rdfs:subClassOf}, o) \in D \wedge o \in \mathcal{R} \\ 1 \text{ if } (r, \texttt{rdfs:subPropertyOf}, o) \in D \wedge o \in \mathcal{R} \\ 1 \text{ if } (r, \texttt{owl:equivalentClass}, o) \in D \wedge o \in \mathcal{R} \\ 1 \text{ if } (r, \texttt{owl:equivalentProperty}, o) \in D \wedge o \in \mathcal{R} \\ 0 \text{ otherwise} \end{cases} \tag{50}$$

Consequently, a bad quality score is assigned if there is no explicit type statement for r and r is not a class itself.

Besides the typing of an RDF resource, further information might be desirable, e.g. subclass hierarchies, class equivalences, special characteristics of properties etc., which can be described using the OWL vocabulary. This is checked by the following metric.

Metric 23 (OWL Ontology Declarations) *The metric checking if a local resource is anchored in an* ontological context*, described via the RDFS or OWL vocabularies, is a dataset metric. 'Ontological context' is referred to as certain statements that define the relations of a considered resource to an underlying ontology, e.g. its assigned type, subclass relations, disjointness statements or domain/range restrictions. The proposed properties, applicable for such statements are subsumed in the set ONTDEFPROPERTIES $\subset \mathcal{R}$ and can be looked up in Appendix A.2. The function $\hat{f}_{23} : \mathcal{R} \rightarrow \mathbb{R}$ assigning a quality score to an input resource $r \in R_{local}$ (with R_{local} being defined as in Metric 5) is then given as follows:*

$$\hat{f}_{23}(r) = \begin{cases} 0 \text{ if } \nexists t \begin{pmatrix} t \in D \wedge subject(t) = r \wedge \\ predicate(t) \in ONTDEFPROPERTIES \end{pmatrix} \\ 1 \text{ otherwise} \end{cases} \tag{51}$$

Most of the violation cases could also be detected, if this metric was designed as view metric, looking for the respective predicate constants in the quad patterns. This would have the advantage that an error is only reported once for a violating resource generation rule in the view definition. In contrast to this, the metric as defined above reports each violating resource that was generated by the RDB2RDF mapping rule, which might result in thousands of entries possibly all having the same cause. Nonetheless, certain violations might not be detectable without assessing the actual database entries, e.g. if predicate variables are used in the corresponding quad patterns. Thus, to reduce the complexity of the metric, the simpler approach of assessing the dataset was followed. Since the pinpointing results should also yield the candidates that most probably generated the violating resource, the reported information should suffice to find the actual cause of the violations.

Another characteristic of highly interpretable data is the conformance with current best practices. One such best practice is to avoid blank nodes, since they cannot be

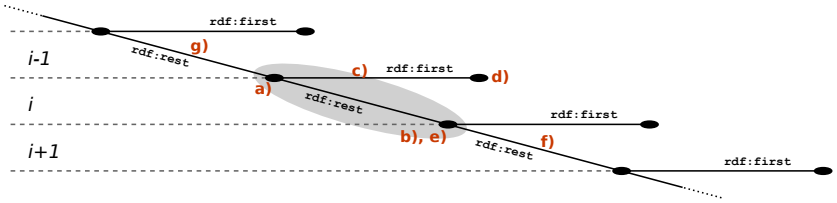

Fig. 17: Schematic depiction of the checks performed in the Correct Collection Use metric on an RDF sub graph expressing an RDF collection

interlinked with other resources and hampers the consolidation or merging of data from different data sources [62]. The metric covering this issue was also proposed by Zaveri et al. [67] and is introduced in the following.

Metric 24 (Avoid Blank Nodes) *The metric assessing if blank nodes are introduced in a dataset $D \in \mathcal{D}$ via RDB2RDF mappings, is a view metric. Given the set of quad variables Q defined in the view definitions of $V \subset \mathcal{V}$*

$$Q = \bigcup_{v_i \in V} \left\{ q \left| q \in \begin{pmatrix} subject(quads(v_i)) \cup \\ predicate(quads(v_i)) \cup \\ object(quads(v_i)) \end{pmatrix} \cap Q \right. \right\}, \tag{52}$$

the function $\hat{f}_{24} : Q \to \mathbb{R}$ assigning a quality score to a quad variable $q \in Q$ is defined as follows:

$$\hat{f}_{24}(q) = \begin{cases} 0 \text{ if } rdf\_term\_type(term\_constructor(q)) = \texttt{bNode} \\ 1 \text{ otherwise} \end{cases} \tag{53}$$

For the creation of Q as well as for the following definitions it is assumed that quad variables are resolved by view name and name (e.g. example_view.?example) to avoid variable name clashes amongst different view definitions.

Further capabilities of the Resource Description Framework to describe more complex structures, are RDF collections, RDF containers and RDF reifications. Since the underlying expression mechanisms also introduce complexity and further constraints with respect to syntactical structures, they are covered by dedicated metrics, described in the following.

Metric 25 (Correct Collection Use) *The metric checking the correct usage of RDF collections in a dataset $D \in \mathcal{D}$, is a dataset metric. For every statement $t_{coll_i^{rest}} \in D$ that has an* `rdf:rest` *predicate, the following checks are performed:*

a) rest statement has `rdf:nil` subject*: check, if $subject(t_{coll_i^{rest}}) =$* `rdf:nil`
b) rest statement has literal object*: check, if $object(t_{coll_i^{rest}})$ is a literal*
c) none or multiple first statements*: check, if there is none or more than one statement $t_{coll_i^{first}}$ with $subject(t_{coll_i^{first}}) = subject(t_{coll_i^{rest}})$ and $predicate(t_{coll_i^{first}}) =$* `rdf:first`

d) first statement has literal object*: if there is a statement $t_{coll_i^{first}}$, check if $object(t_{coll_i^{first}})$ is a literal*

e) collection not terminated with `rdf:nil`*: check, if $object(t_{coll_i^{rest}}) \neq$ `rdf:nil` and there is no statement $t_{coll_{i+1}^{rest}}$ with $subject(t_{coll_{i+1}^{rest}}) = object(t_{coll_i^{rest}})$*

f) multiple successors*: check, if there are multiple statements $t_{coll_{i+1}^{rest}}$ with $subject(t_{coll_{i+1}^{rest}}) = object(t_{coll_i^{rest}})$*

g) multiple predecessors*: check, if there are multiple statements $t_{coll_{i-1}^{rest}}$ with $object(t_{coll_{i-1}^{rest}}) = subject(t_{coll_i^{rest}})$*

A schematic depiction of these checks applied to an example RDF sub graph is given in Figure 17. The function $\hat{f}_{25} : \mathcal{T} \to \mathbb{R}$ determining the quality score of a collection statement $t_{coll_i^{rest}}$ is defined as follows:

$$\hat{f}_{25}(t_{coll_i^{rest}}) = \begin{cases} 0 & \textit{if any of the checks b) and d) is positive} \\ 0.5 & \textit{if any of the checks a), c), e), f) and g) is positive} \\ 1 & \textit{otherwise} \end{cases} \tag{54}$$

Since the RDF specification does not require a collection to be *'well-formed'* [87], only the `rdfs:range` violations in the cases b) and d) are actual errors with regards to the underlying semantics. Accordingly, all the other cases, checked in a), c), e), f) and g) are not violating the RDF specification. Nonetheless these cases are considered violations of a well-formed collection thus yield a score of 0.5.

Metric 26 (Correct Container Use) *The metric checking the correct usage of RDF containers in a dataset $D \in \mathcal{D}$, is a dataset metric. For every statement $t_{cont_i} \in D$ that has a `rdfs:ContainerMembershipProperty` on predicate position, the following checks are performed:*

a) container not typed: *if $predicate(t_{cont_i})$ = `rdf:_1`, check if neither `rdf:Bag`, `rdf:Seq`, nor `rdf:Alt` is assigned to $subject(t_{cont_i})$ via `rdf:type`*

b) literal objects: *check, if $object(t_{cont_i})$ is a literal*

c) multiple entries for one container membership property: *check, if there is a statement $t_{cont_{i'}}$ with $subject(t_{cont_{i'}}) = subject(t_{cont_i})$, $predicate(t_{cont_{i'}}) = predicate(t_{cont_i})$ and $object(t_{cont_{i'}}) \neq object(t_{cont_i})$*

d) numbering gaps: *check, if*
- *there is a statement $t_{cont_{i+2}}$ with $subject(t_{cont_{i+2}}) = subject(t_{cont_i})$ and the predicates of t_{cont_i} and $t_{cont_{i+2}}$ are differing in two steps (with $predicate(t_{cont_{i+2}})$ being the bigger one),*
- *but no statement $t_{cont_{i+1}}$ with $subject(t_{cont_{i+1}}) = subject(t_{cont_i})$ and the predicates of t_{cont_i} and $t_{cont_{i+1}}$ differing in one step (with $predicate(t_{cont_{i+1}})$ being the bigger one)*

e) container starts at `rdf:_0`*: check if there is a statement t_{cont_0} with $predicate(t_{cont_0})$ =* `rdf:_0`

f) container membership properties with leading zeros: *check if there are statements t_{cont_i} with $predicate(t_{cont_i})$ having leading zeros, e.g.* `rdf:_023`

The function $\hat{f}_{26} : \mathcal{T} \rightarrow \mathbb{R}$ *assigning a quality score to a container statement* t_{cont_i} *is defined as follows:*

$$\hat{f}_{26}(t_{cont_i}) = \begin{cases} 0 & \textit{if any of the checks b), e) and f) is positive} \\ 0.5 & \textit{if any of the checks a), c) and d) is positive} \\ 1 & \textit{otherwise} \end{cases} \tag{55}$$

As with collections, the RDF specification does not impose many restrictions on containers, as far as semantics is concerned [87]. Thus, only the range violation of container membership property instances as checked in b), the numbering violation checked in e), and the syntax violation of d) are real errors with respect to the RDF specification. The remaining cases are not excluded explicitly or even explicitly stated to be not a semantic violation. Nonetheless they are considered erroneous with regards to the *'well-formedness'* of an RDF container and thus yield a score of 0.5.

The last metric proposed for the interpretability dimension covers reification statements. Even though, the Correct Reification Use metric only covers RDF reification, it can be easily extended to also support OWL2 annotations, since the classes and properties involved in RDF reifications have their direct counterparts in OWL2.

Metric 27 (Correct Reification Use) *The metric checking the correct usage of reification statements in a dataset* $D \in \mathcal{D}$*, is a dataset metric. For every statement* $t_{reif_i} \in D$ *with either* $predicate(t_{reif_i}) \in \{$ `rdf:subject` , `rdf:predicate` , `rdf:object`$\}$*, or* t_{reif_i} *being typed as* `rdf:Statement`*, the following checks are performed:*

a) reification not typed properly: *check, if* $subject(t_{reif_i})$ *is not typed as* `rdf:Statement`
b) none or multiple `rdf:subject` statements: *check, if there is none or more than one statement* $t_{reif_i^{subj}}$ *with* $subject(t_{reif_i^{subj}}) = subject(t_{reif_i})$ *and* $predicate(t_{reif_i^{subj}}) =$ `rdf:subject`
c) literal value of `rdf:subject` property: *if* $t_{reif_i^{subj}}$ *exists, check if* $object(t_{reif_i^{subj}})$ *is a literal*
d) none or multiple `rdf:predicate` statements: *check, if there is none or more than one statement* $t_{reif_i^{pred}}$ *with* $subject(t_{reif_i^{pred}}) = subject(t_{reif_i})$ *and* $predicate(t_{reif_i^{pred}}) =$ `rdf:predicate`
e) literal or blank node value of `rdf:predicate` property: *if* $t_{reif_i^{pred}}$ *exists, check if* $object(t_{reif_i^{pred}})$ *is a literal or a blank node*
f) none or multiple `rdf:object` statements: *check, if there is none or more than one statement* $t_{reif_i^{obj}}$ *with* $subject(t_{reif_i^{obj}}) = subject(t_{reif_i})$ *and* $predicate(t_{reif_i^{obj}}) =$ `rdf:object`

The function $\hat{f}_{27} : \mathcal{T} \rightarrow \mathbb{R}$ *assigning a quality score to an input statement* t_{reif_i} *involved in a reification, is defined as follows:*

$$\hat{f}_{27}(t_{reif_i}) = \begin{cases} 0 & \textit{if any of the checks a) - f) is positive} \\ 1 & \textit{otherwise} \end{cases} \tag{56}$$

Performance RDB2RDF mappings usually have no influence on the actual performance of the underlying service, which is mainly determined by characteristics of the server machine and the implementation. The only aspect that can be influenced concerns the URI design. Since slash URIs are considered preferable to hash URIs as far as performance is concerned [64], Metric 28 reports hash URI occurrences. This metric was also proposed by Zaveri et al. [67].

Metric 28 (No Hash URIs) *The metric checking if a hash URI is used as identifier of a local resource, is a node metric. Assuming that for every URI the percent-encoding was applied and R_{local} is defined as in Metric 5, the function $\hat{f}_{28} : \mathcal{R} \rightarrow \mathbb{R}$ determining the quality score of an input resource r, is defined as follows:*

$$\hat{f}_{28}(r) = \begin{cases} 0 & r \in R_{local} \wedge \textit{the URI string of r contains the hash character '\#'} \\ 1 & \textit{otherwise} \end{cases} \tag{57}$$

Even though, the usage of hash URIs is mostly considered as bad, as far as performance is concerned, it might also be advantageous in some cases. Given an application will have to access a bigger portion of hash URIs via HTTP, it can be designed to just retrieve the document located at the URL without the fraction part once and save it for caching purposes. Thus, further accesses could be served from the cache without needing any HTTP requests at all, which would be a performance benefit. Accordingly, it also depends on the actual use case and the dataset, if such URIs really harm the overall performance.

Relevancy The relevancy dimension *"refers to the provision of information which is in accordance with the task at hand"* [67]. Since this is usually not assessable without further specifications of the requirements a certain task has, more general aspects are considered here. These comprise the categorization of the created dataset with respect to its triple count and the evaluation of characterizing ratios. Having the triple count, gives at least some rough feedback, whether the dataset can be expected to contain the desired information. For the actual categorization the scale of Flemming [64] is applied. The values used by Flemming reflect the size distribution of the datasets contributing to the LOD Cloud in the year 2011. Based on the current statistics of the LODStats project[31], these values are considered to be still applicable for a categorization and are thus used here. The corresponding metric is defined as follows:

Metric 29 (Amount of Triples) *The metric assessing the size of a dataset $D \in \mathcal{D}$ is a dataset metric. The actual size of D is determined by counting the triples $t \in D$, denoted by the cardinality bars $|D|$. The quality score function $f_{29} : \mathcal{D} \rightarrow \mathbb{R}$ for an input dataset D is defined as follows:*

$$f_{29}(D) = \begin{cases} 1 & \textit{if} \quad |D| \geq 1,000,000,000 \\ 0.75 & \textit{if} \quad 1,000,000,000 > |D| \geq 10,000,000 \\ 0.5 & \textit{if} \quad 10,000,000 > |D| \geq 500,000 \\ 0.25 & \textit{if} \quad 500,000 > |D| \geq 10,000 \\ 0 & \textit{if} \quad 10,000 > |D| \end{cases} \tag{58}$$

[31] `http://stats.lod2.eu/stats`

The ratio characteristics to be evaluated in the assessment are the *coverage* with respect to the level of detail of a dataset and with respect to its scope. As introduced by Flemming [64], these characteristics reflect the aim of providing enough properties to describe resources in detail, and of having enough of these resources to cover the considered domain. Accordingly, if a dataset contains only few distinct RDF properties, its coverage with respect to the level of detail is low. On the other hand, if there are actually only few instances described in the dataset the scope coverage is considered to be bad. Thus, both aspects are contradictory in the sense, that a dataset can not have a perfect scope coverage and detail coverage at the same time. Instead, increasing one of them lowers the other one. The corresponding metrics are defined as follows:

Metric 30 (Coverage (Detail)) *The metric assessing the coverage of a dataset with respect to its level of detail is a dataset metric. For a dataset $D \in \mathcal{D}$ this coverage is given as the ratio of the number of properties $|D|_{prop}$ and the number of triples $|D|$. The quality score function $f_{30} : \mathcal{D} \rightarrow \mathbb{R}$ for an input dataset D is defined as follows:*

$$f_{30}(D) = \frac{|D|_{prop}}{|D|} \tag{59}$$

Metric 31 (Coverage (Scope)) *The metric assessing the coverage of a dataset with regards to its scope is a dataset metric. For a dataset $D \in \mathcal{D}$ this coverage is given as the ratio of the number of instances $|D|_{inst}$ (as introduced in Metric 3), and the number of triples $|D|$. The quality score function $f_{31} : \mathcal{D} \rightarrow \mathbb{R}$ for an input dataset D is defined as follows:*

$$f_{31}(D) = \frac{|D|_{inst}}{|D|} \tag{60}$$

Representational Conciseness The representational conciseness dimension covers best practices that guarantee a terse and clear representation of the RDF data. The main aspects, proposed by Zaveri et al. [67], are short and query parameter free URIs and the avoidance of so called *'prolix features'* [62]. Short URIs can be memorized more easily by users. Moreover, they are better suited for efficient storage concerns, e.g. on-disk indexes, compression techniques or caching [62]. The avoidance of prolix features like RDF collections, RDF containers and reification statements is motivated in [86,62] with the assertion, that they lack a wide tool support and are hard to query via SPARQL.

The actual metrics assessing these issues are introduced in the following.

Metric 32 (Short URIs) *The metric assessing the length of the identifier string of a resource $r \in \mathcal{R}$ is a node metric. Assuming a certain threshold θ was set up, the function $\hat{f}_{32} : \mathcal{R} \rightarrow \mathbb{R}$ assessing the quality score of an input resource r, is defined as follows:*

$$\hat{f}_{32}(r) = \begin{cases} 0 \textit{ the URI string length is greater than } \theta \\ 1 \textit{ otherwise} \end{cases} \tag{61}$$

Metric 33 (No Prolix Features) *The metric checking if* prolix features *are used in an RDF statement is a triple metric. According to [62] such prolix features are* (*a*) *RDF*

reifications, (*b*) *RDF containers and* (*c*) *RDF collections. The function* $f_{33} : \mathcal{T} \rightarrow \mathbb{R}$ *determining the quality score of an input triple* $t \in D$ ($D \in \mathcal{D}$) *is given as:*

$$f_{33}(t) = \begin{cases} 0 \text{ if } predicate(t) = \texttt{rdf:subject} & (a_1) \\ 0 \text{ if } predicate(t) = \texttt{rdf:predicate} & (a_2) \\ 0 \text{ if } predicate(t) = \texttt{rdf:object} & (a_3) \\ 0 \text{ if } subject(t) \in \texttt{rdf:Statement} & (a_4) \\ 0 \text{ if } predicate(t) \in \texttt{rdfs:ContainerMembershipProperty} & (b_1) \\ 0 \text{ if } subject(t) \in \texttt{rdf:Alt} & (b_2) \\ 0 \text{ if } subject(t) \in \texttt{rdf:Bag} & (b_3) \\ 0 \text{ if } subject(t) \in \texttt{rdf:Seq} & (b_4) \\ 0 \text{ if } subject(t) \in \texttt{rdf:Container} & (b_5) \\ 0 \text{ if } predicate(t) = \texttt{rdf:first} & (c_1) \\ 0 \text{ if } predicate(t) = \texttt{rdf:rest} & (c_2) \\ 0 \text{ if } subject(t) \in \texttt{rdf:List} & (c_3) \\ 1 \text{ otherwise} \end{cases} \tag{62}$$

The conditions checked in f_{33} *are grouped as follows:* $(a_1) - (a_4)$ *check for reification use,* $(b_1) - (b_5)$ *asses if* t *is an RDF container statement, and* $(c_1) - (c_3)$ *covers RDF collection expressions.*

Metric 34 (Query Parameter-free URIs) *The metric checking whether the identifier string of a resource* $r \in \mathcal{R}$ *contains query parameters, is a node metric. Assuming that percent-encoding was applied for every URI, the quality score of an input resource* r *is determined by the function* $\hat{f}_{34} : \mathcal{R} \rightarrow \mathbb{R}$*, which is defined as follows:*

$$\hat{f}_{34}(r) = \begin{cases} 0 \text{ if the URI string of } r \text{ contains the question mark character '?'} \\ 1 \text{ otherwise} \end{cases} \tag{63}$$

Semantic Accuracy The semantic accuracy dimension refers to the extent *"to which data values are represented correctly"* [67]. Since the actual values are copied from the database to become (parts of) RDF literals or URIs, an accuracy degradation with regards to the *extensional* data seems implausible. Even the decision, if values should be represented as own resources or whether they should be mapped to literals, highly depends on the actual use case and can not be judged in general. In case inaccuracies *are* introduced, the actual variations have to be stated explicitly using the corresponding modification expressions. Thus, the introduction of extensional inaccuracy based on *erroneous* mappings is unlikely to occur.

Semantic information that should be accurate in both, the RDB and RDF context, and can degrade during the conversion, are *intensional* data stored in the relational schema definitions. Such relational schema information can be divided into *intrarelation* and *interrelation* constraints [25]. Their translation to RDF will result in richer ontologies, which are desired from an accuracy perspective. Intrarelation constraints define restrictions on single attributes as well as constraints that must hold between multiple attributes. One representative of this category is the `NOT NULL` constraint, stating that a certain attribute of a relation must exist. To preserve this in the RDF data,

cardinality constraints have to be introduced for properties using values derived from `NOT NULL`-constrained attributes. To give an example, an employee database can be considered. There, a relational table `EMPL`, holding employee entries, might have a column `birth_date` which is constrained to be not `NULL`. In case the `birth_date` column is used in an RDB2RDF mapping, to accurately translate this to RDF, the modeled vocabulary or ontology should contain a corresponding hint, that every employee resource has a birthday. The metric to assess this is defined as follows:

Metric 35 (Preserved NOT NULL Constraints) *The metric assessing the preservation of relational* `NOT NULL` *constraints is a view metric. Given a set of view definitions $V \subset \mathcal{V}$, the set of quad object variables referencing database values having a* `NOT NULL` *constraint, can be defined as follows:*

$$Q_{NOT\_NULL} = \bigcup_{v_i \in V} \left\{ q \;\middle|\; \begin{array}{l} q \in \left(object(quads(v_i)) \cap Q \right) \wedge \\ \Gamma = cols(term\_constructor(q)) \wedge \\ \exists \gamma \left(\gamma \in \Gamma \;\wedge\; \gamma \text{ has a } \texttt{NOT NULL} \text{ constraint} \right) \end{array} \right\} \tag{64}$$

The function $\hat{f}_{35} : \mathcal{N} \times \mathcal{N} \times \mathcal{N} \to \mathbb{R}$ assigning a quality score to a quad $p \in quads(v_i)$ of a view definition $v_i \in V$ is given as

$$\hat{f}_{35}(p) = \begin{cases} & object(p) \in Q_{NOT\_NULL} \wedge \\ 0 \text{ if } & \nexists p_c \left(\begin{array}{l} p_c \in \bigcup_{v_i \in V} quads(v_i) \wedge \\ subject(p_c) = predicate(p) \wedge \\ \left(\begin{array}{l} predicate(p_c) = \texttt{owl:cardinality} \vee \\ predicate(p_c) = \texttt{owl:minCardinality} \end{array} \right) \end{array} \right) \\ 1 \text{ otherwise} \end{cases} \tag{65}$$

Another intrarelation constraint is the `UNIQUE` restriction, forbidding multiple equivalent values of a certain attribute. Conversely, this means, that a database object is identified by such an attribute *uniquely*. The corresponding OWL class to be assigned to the respective RDF properties used in conjunction with the `UNIQUE` values, is the `owl:InverseFunctionalProperty` class. A metric checking the accurate mapping of `UNIQUE` values could be defined analogous to Metric 35 (Preserved NOT NULL Constraints), which is omitted here for brevity.

A further means to define intrarelation constraints in SQL is the `CHECK` clause. Such an expression can contain arbitrary conditions that must hold and would require a formal introduction of the SQL, and on the implementation side a full fledged SQL parser to read all constraints. Moreover all these constraints that refer to parts of the relational database, that are also mapped to RDF must be translated to suitable OWL constraints. Since this is not an easy task and may not be feasible in a generic and automatic way at all, this is not considered here and also not checked by the R2RLint prototype.

Apart from such restrictions stated explicitly, another kind of constraint arises from the semantics of relational databases. In one single tuple, every attribute can be considered functional for the entry at hand. Regarding the birthday example of the employee database again, this would also impose the explicit declaration of the introduced birthday property to be functional. The preservation of such functional attributes is assessed by the following metric.

Metric 36 (Preserved Functional Attributes) *The metric assessing the preservation of relational attributes' characteristics of being functional is a view metric. Given a set of view definitions $V \subset \mathcal{V}$, the set of quad object variables, whose term constructors refer to functional columns of the underlying relational table, can be defined as follows:*

$$Q_{func} = \bigcup_{v_i \in V} \left\{ n_o \;\middle|\; \begin{array}{c} t \in quads(v_i) \;\wedge\; n_s = subject(t) \;\wedge\; n_o = object(t) \wedge \\ n_s \in \mathcal{Q} \;\wedge n_o \in \mathcal{Q} \wedge \\ \exists \Gamma \left(\begin{array}{l} \Gamma \subseteq cols(term\_constructor(n_s)) \wedge \\ \Gamma \text{ can be considered as primary key of} \\ \text{the underlying (logical) relational table} \end{array} \right) \end{array} \right\} \tag{66}$$

Q_{func} then contains all quad variables on object positions that should be declared functional via the appropriate type of the corresponding property. The function $\hat{f}_{36} : \mathcal{N} \times \mathcal{N} \times \mathcal{N} \to \mathbb{R}$ assigning a quality score to a quad pattern $p \in quads(v_i)$ of a view definition $v_i \in V$ is given as

$$\hat{f}_{36}(p) = \begin{cases} & object(p) \in Q_{func} \wedge \\ 0 \text{ if } & \nexists p_f \left(\begin{array}{l} p_f \in \bigcup_{v_i \in V} quads(v_i) \wedge \\ subject(p_f) = predicate(p) \wedge \\ predicate(p_f) = \texttt{rdf:type} \wedge \\ object(p_f) = \texttt{owl:FunctionalProperty} \end{array} \right) \\ 1 \text{ otherwise} \end{cases} \tag{67}$$

Interrelation constraints, in contrast, are restrictions defined over multiple relations. Besides more complex `CHECK` constraints, foreign key dependencies are the most prominent representatives. One major condition that can be checked with respect to accuracy is whether foreign key relations are preserved by the RDB2RDF mapping. Given a tuple that contains a foreign key reference to another tuple, at least two RDF resources can be introduced: one that represents the considered tuple entity and another one for the referenced tuple. In case, both resources were generated in an RDB2RDF mapping, to preserve the foreign key relation between them, there should also be a statement having the former as triple subject and the latter as triple object (cf. Figure 18). The metric evaluating this aspect is given as follows:

Metric 37 (Preserved Foreign Key Constraints) *The metric assessing the preservation of foreign key relations between relational database entries is a view metric. Given the set Ψ of all foreign key dependencies $(\delta_i, \Gamma_i) \leftarrow (\delta_j, \Gamma_j)$ defined over RDB and the set of view definitions $V \subset \mathcal{V}$, a set PSUB of pairs can be constructed. Each pair in PSUB represents a view definition's quad subject that is constructed using data from columns that are the target of a foreign key dependency. The first entry of each pair contains the corresponding relational table and the second one holds the actual quad variable. PSUB is then defined as follows:*

$$PSUB = \bigcup_{v_i \in V} \left\{ (\delta, q) \;\middle|\; \begin{array}{l} \delta = rel\_table(v_i) \;\wedge\; q \in \Big(subject(quads(v_i)) \cap \mathcal{Q} \Big) \;\wedge \\ \Gamma = cols(term\_constructor(q)) \wedge \\ \exists (\delta', \Gamma') \Big((\delta, \Gamma) \leftarrow (\delta', \Gamma') \in \Psi \Big) \end{array} \right\} \tag{68}$$

With this set, violations with regards to the Preserved Foreign Key Constraints metric can be found. To give an example, the RDB2RDF mappings depicted in Figure 18 are considered. Due to the foreign key dependency (`DEPT`, `id`) ← (`EMPL`, `dept`) *between the two relational tables* `EMPL` *and* `DEPT`*, PSUB would contain* (`DEPT`, `uri(ex:dept, ?id)`)*. This means that there are URIs constructed, using data from the* `id` *column of the* `DEPT` *table. Since the view definition* `empl_view` *also generates statements about resources, that are constructed using the primary key columns of* `EMPL` *(via* `uri(ex:person, ?id)`*), there should be an assertion, expressing the 'foreign key link' that existed in the relational database (cf. Figure 18). Accordingly, the function* $\hat{f}_{37} : \mathcal{V} \rightarrow \mathbb{R}$ *assinging a*

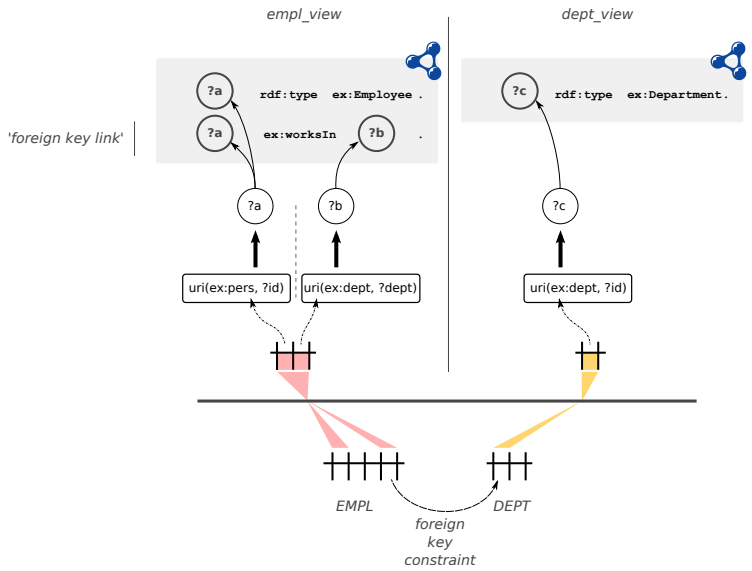

Fig. 18: Example showing the preservation of a relational foreign key constraint via a 'foreign key link' SML quad expression

quality score to a view definition $v_i \in V$*, is defined as follows:*

$$\hat{f}_{37}(v_i) = \begin{cases} 0 \text{ if } & \begin{array}{l} q_s \in \Big(subject(quads(v_i)) \cap Q \Big) \wedge \delta = rel\_table(v_i) \wedge \\ \Gamma_s = cols(term\_constructor(q_s)) \wedge \\ \Gamma_s \text{ are the primary key columns of } \delta \wedge \\ (\delta', \Gamma') \leftarrow (\delta, \Gamma_o) \in \Psi \wedge \end{array} \\ & \exists (\delta', q') \begin{pmatrix} (\delta', q') \in PSUB \wedge \\ cols(term\_constructor(q')) = \Gamma' \end{pmatrix} \wedge \quad \dagger \\ & \nexists p' \begin{pmatrix} p' \in quads(v_i) \wedge subject(p') = q_s \wedge \\ object(p') \in Q \wedge \\ cols(term\_constructor(object(p'))) = \Gamma_o \end{pmatrix} \quad \ddagger \\ 1 \text{ otherwise} \end{cases} \tag{69}$$

In the definition of $\hat{f}$, the term marked with † expresses the requirement, that there is a quad, generating statements about a resource constructed using data from the target columns of a foreign key dependency. Assessing `empl_view` *from the example above, this would correspond to the existence of the quad variable* `?c` *in* `dept_view`*. The term marked with ‡ however, checks if no 'foreign key link' exists. Since the example contains the quad expression* `?a ex:worksIn ?c` *with* `?c` *referring to the referencing foreign key columns Γ_o = {*`dept`*}, it is not violating the preservation of a relational foreign key dependency.*

It has to be noted, that even though it could have been defined somewhere else, in this metric the 'foreign key link' is only searched within one view definition. This was done to reduce complexity and is not a limitation of the overall approach.

Syntactic Validity The syntactic validity dimension refers to the conformance of data with given syntax specifications [67]. Since the specifications of most importance in the RDB2RDF case are those that define the structures of RDF data, the assessment of syntactic validity, as considered here, should ensure that valid RDF data is generated. Due to the fact that the Sparqlification Mapping Language does not allow to create invalid RDF structures in general, the only aspects to cover are the datatype compatibility of typed literals, as proposed by Zaveri et al. [67], and the usage of valid language tags. The introduction of literals with invalid datatypes may occur due to copy and paste errors, when setting up a view definition. Invalid language tags, on the other hand, might be introduced by accident, e.g. because of typing errors, or if database values are used to generate the language tags. The metrics covering these issues, are introduced as follows:

Metric 38 (Datatype-compatible Literals) *The metric assessing the compatibility of literals with respect to their XML Schema datatypes is a node metric. A typed literal's value is compatible with the literal's datatype if its lexical representation is within the datatype's value range as specified in [84]. The quality score function $f_{38} : \mathcal{N} \rightarrow \mathbb{R}$ for an input node $n \in \mathcal{N}$ is given as*

$$f_{38}(n) = \begin{cases} 0 \text{ if } n \in \mathcal{L} \ \wedge \ n\text{'s value is not compatible with } n\text{'s datatype} \\ 1 \text{ otherwise} \end{cases} \tag{70}$$

As a convention, f_{38} returns 1, if the input node is not a literal.

This metric could also be designed as view metric. But, since its formalization would be very complex, especially for the consideration of all the term constructor functions defined in the Sparqlification Mapping Language, this simpler node metric was introduced.

The same holds for the next metric. Here language tags, defined as constant expressions, could also be evaluated in the corresponding view definition and would not require the assessment of all generated plain literals. But since the actual language tag could also be defined referring to database values, and to avoid an overly complex formalization, again the simpler definition of a node metric is introduced as follows:

Metric 39 (Valid Language Tag) *The metric assessing the validity of a literal's language tag is a node metric. A plain literal's language tag is considered valid if it is compliant with the BCP 47 standard [88]. The quality score function* $f_{39} : \mathcal{N} \rightarrow \mathbb{R}$ *for an input node* $n \in \mathcal{N}$ *is given as*

$$f_{39}(n) = \begin{cases} 0 \text{ if } n \in \mathcal{L} \ \wedge \ n \text{ has a language tag} \wedge n\text{'s language tag is not valid} \\ 1 \text{ otherwise} \end{cases} \tag{71}$$

Again, 1 is returned by convention for non-literal nodes or literal nodes not having a language tag.

Understandability The understandability dimension contains metrics assessing whether RDF data, generated by an RDB2RDF mapping can be easily consumed by humans. One first step towards this ease of consumption is the provision of human readable labels for resources, as proposed by Zaveri et al.[67]. The corresponding metric is introduced in the following.

Metric 40 (Labeled Resources) *The metric assessing whether a resource* $r \in \mathcal{R}$ *is properly labeled, is a dataset metric. With* R_{local} *being defined as in Metric 5, the function* $\hat{f}_{40} : \mathcal{R} \rightarrow \mathbb{R}$ *assessing the Labeled Resources quality score of a resource r is given as follows:*

$$\hat{f}_{40}(r) = \begin{cases} 0 \text{ if } r \in R_{local} \ \wedge \ (r \text{ , } \texttt{rdfs:label} \text{ , } l) \notin D \\ 1 \text{ otherwise} \end{cases} \tag{72}$$

This metric could be further extended, to also regard the provision of labels in different languages. Then, the requirements would be to support as much languages as possible and to cover these languages homogeneously. This means, that if e.g. 10 languages are supported, every resource that is labeled, should have a label in each of these languages, which would require 10 labels in the given example. Conversely, if e.g. 50 languages are supported, but every labeled resource just had one label in one single language, this would be regarded as bad quality.

Apart from human readable labels of resources, their actual identifiers, the URIs, should be *sounding* to be easy to remember and easy to type manually. Since it can not be directly derived if a URI is sounding, a dictionary approach could be used. But applying a dictionary comparison introduces further problems of natural language processing, e.g. finding word boundaries in case a URI contains multiple words not separated explicitly or the resolution of abbreviations. To assess if a URI contains parts, that *sound* like words of a given language several trials were made to apply phonotactic [89] techniques [90,91,92], libraries[32] and tools[33]. Unfortunately, none of them were able to provide or persist phonotactic rules, to be reused for the quality assessment, with a reasonable effort. Thus, a probabilistic phonotactics [93] approach based on trigrams was developed. With this approach trigrams that are common in a given language get a higher score than uncommon ones. Accordingly, strings that sound like real words,

[32] e.g. `https://github.com/marytts/marytts/tree/master/marytts-lang-en`
[33] e.g. `http://www.linguistics.ucla.edu/people/hayes/Phonotactics/Manual.pdf`

e.g. ‘uffish’ in the English language are rated higher than words like ‘czvfgw’. The corresponding metric is defined as follows:

Metric 41 (Sounding URIs) *The metric assessing whether a URI is* sounding*, is a node metric. The degree to which a URI is sounding, is determined using trigram statistics of a training set of words stemming from a corpus in a certain language. For every word the occurrences of its contained trigrams are counted and added to a global statistic Φ. Φ then contains the more frequent trigrams of a language (w.r.t. to the underlying corpus) with higher counts and uncommon, less frequent trigrams with lower counts. The global count of a trigram ϕ can be retrieved via $\Phi(\phi)$, returning 0 if ϕ is not contained in Φ. It is further possible to query the maximal count gathered with $max(\Phi)$. The maximal count can be used to set up a normalization factor ν as follows:*

$$\nu = \frac{1}{max(\Phi)} \tag{73}$$

With $\phi \in r$ expressing that a certain trigram ϕ is contained in the identifier of a resource $r \in \mathcal{R}$, and $|r|_{tri}$ denoting the number of trigrams contained in r, the function $\hat{f}_{41} : \mathcal{R} \rightarrow \mathbb{R}$, assessing the quality score of an input resource r is given as:

$$\hat{f}_{41}(r) = \frac{\sum_{\phi \in r} \Phi(\phi)}{|r|_{tri}} \nu \tag{74}$$

One detail, omitted in the metric definition for brevity, is that actually not the whole URI string, but its parts are assessed. The corresponding partial results are then aggregated for the whole URI string. This was mainly done to avoid counting characters like ‘/’ and ‘:’, intended to serve as word separator.

Another important issue concerning the understandability dimension is the provision of further information on the Web. Thus URIs should also be valid HTTP URLs. Since in the RDB2RDF case URIs are mostly generated, it is of importance to verify that the created URIs are valid with respect to the underlying standards. Unfortunately there are different standards that should be taken into account, depending on the considered part of the URI. Even though there are URI and URL schema definitions[34,35] as well as corresponding standards [94,95] provided by the *Internet Engineering Task Force (IETF)* and W3C, these are not throughout consistent with other standards, e.g. for internet host names [96], URIs based on IP addresses [97] or *Internationalized Domain Names (IDN)* [98]. To actually check, if a URI is a valid HTTP URI a regular expression was compiled taking into account most of the involved standards. The regular expression can be looked up in Appendix A.3. This regular expression is used in the following metric, defined to assess if resource identifiers are valid HTTP URIs.

Metric 42 (HTTP URIs) *The metric assessing whether an identifier of a resource $r \in \mathcal{R}$ is a valid HTTP URI, is a node metric. The function $\hat{f}_{42} : \mathcal{R} \rightarrow \mathbb{R}$ determining the quality score of an input resource r is defined as:*

$$\hat{f}_{42}(r) = \begin{cases} 0 \textit{ if r is not a valid HTTP URI} \\ 1 \textit{ otherwise} \end{cases} \tag{75}$$

[34] `http://www.w3.org/Addressing/URL/5_BNF.html`
[35] `http://tools.ietf.org/html/draft-fielding-url-syntax-09#appendix-A`

Another way to support the understandability is the provision of certain metadata [67]. Such metadata may concern the actual data (e.g. a short description, the intended language or the covered topic), their titles or the creators. There are many different vocabularies that might be used to express such information. The vocabularies proposed here are the Vocabulary of Interlinked Datasets (VoID)[36], the Dublin Core Metadata Initiative (DCMI) Metadata Terms vocabularies[37] and the Friend of a Fried (FOAF) vocabulary[38]. Following the proposal of [64] to check, whether metadata is contained in the dataset, the corresponding metric is defined as follows:

Metric 43 (Dataset Metadata) *The metric assessing if certain metadata descriptions are provided, is a dataset metric. Given the three sets of proposed metadata properties $R_{title} \subset \mathcal{R}$, $R_{content} \subset \mathcal{R}$ and $R_{creator} \subset \mathcal{R}$, as defined in Appendix A.4. For a dataset $D \in \mathcal{D}$ the quality score function $f_{43} : \mathcal{D} \rightarrow \mathbb{R}$ is given as follows:*

$$f_{43}(D) = \begin{cases} 1 \quad \textit{if } \exists t_i \, \exists t_j \begin{pmatrix} t_i, t_j \in D \wedge \\ predicate(t_i) = \texttt{rdf:type} \wedge \\ object(t_i) = \texttt{void:Dataset} \wedge \\ subject(t_j) = subject(t_i) \wedge t_j \neq t_i \wedge \\ predicate(t_j) \in (R_{title} \cup R_{content} \cup R_{creator}) \end{pmatrix} \\ 0.5 \quad \textit{if } \exists t_i \, \exists t_j \begin{pmatrix} t_i, t_j \in D \wedge \\ predicate(t_i) = \texttt{rdf:type} \wedge \\ object(t_i) = \texttt{void:Dataset} \wedge \\ subject(t_j) = subject(t_i) \wedge t_j \neq t_i \wedge \\ predicate(t_j) \notin (R_{title} \cup R_{content} \cup R_{creator}) \end{pmatrix} \\ 0 \quad \textit{otherwise} \end{cases} \tag{76}$$

Accordingly, a quality score of 0 is returned in case D does not contain a dataset resource $r_{void:Dataset}$ typed as `void:Dataset` *or D does contain a dataset resource, but no further statements about it are contained. A score of 0.5 is assigned if further statements about $r_{void:Dataset}$ are made, but without using the proposed properties of the sets R_{title}, $R_{content}$ or $R_{creator}$. Only if at least one of these properties is used, the score 1 is returned.*

[36] `http://vocab.deri.ie/void`
[37] `http://dublincore.org/documents/2012/06/14/dcmi-terms`
[38] `http://xmlns.com/foaf/spec/`

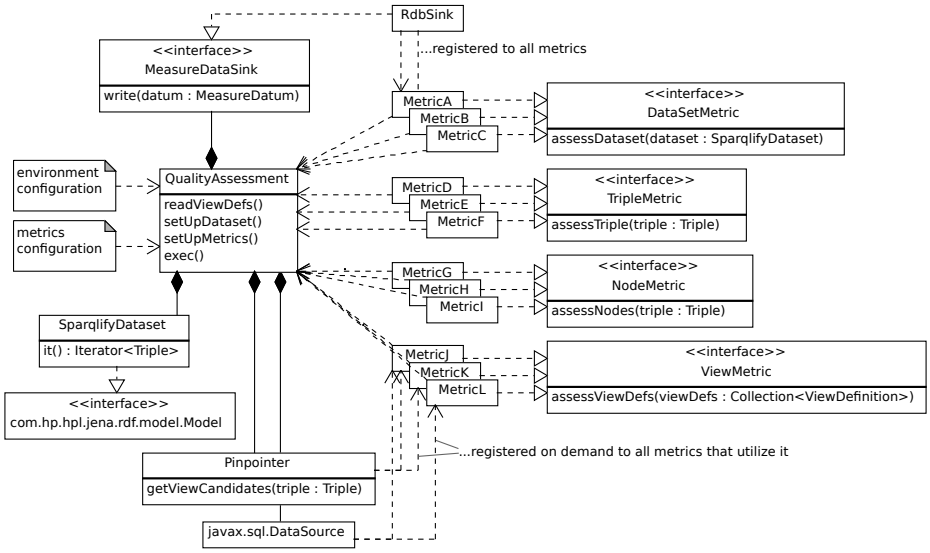

Fig. 19: Class diagram of the R2RLint prototype

4 R2RLint

This section introduces R2RLint, the software prototype implementing the R2RLint methodology. Besides its basic features, implementation limitations are presented to also show the differences to the proposed definitions.

R2RLint is designed as a command line tool, aligned with the requirements for quality evaluation frameworks [34,42]. Due to the decoupling of assessment runner (*QualityAssessment*), configuration (*environment configuration, metrics configuration*), and the actual metrics (example metric classes *MetricA - MetricL*, cf. Figure 19), R2RLint allows to customize the assessment, defining which metrics to apply with which thresholds. Even though R2RLint is equipped with the 43 metrics, introduced in Section 3.5, the R2RLint framework provides an easy way to define own metrics. A simple dummy example is given in Listing 1.3. Due to decoupling mechanisms of the Spring Framework[39] no further wiring or interaction with the assessment framework is needed. This is mainly achieved applying Springs component autowiring mechanisms. The same holds for the actual reporting entity of R2RLint, the measure data sink. Besides the existing RDB sink, writing the assessment results to a configured relational database, and the logging sink, just logging the results to the console, own sinks can be programmed easily, implementing the initialization and write methods of the corresponding interface. This was also used for testing, where special sinks were introduced to easily verify expected results.

R2RLint, comprising the R2RLint framework and 43 implemented metrics backed by 420 software tests, currently contains 16661 lines of code (comments, empty lines

[39] `http://projects.spring.io/spring-framework/`

```java
package org.aksw.sparqlify.qa.metrics.example;

import org.aksw.sparqlify.qa.dataset.SparqlifyDataset;
import org.aksw.sparqlify.qa.metrics.DatasetMetric;
import org.aksw.sparqlify.qa.metrics.MetricImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class Example extends MetricImpl implements DatasetMetric {

    @Autowired
    DataSource rdb;
    @Autowired
    Pinpointer pinpointer;

    @Override
    public void assessDataset(SparqlifyDataset dataset) {
        // custom dataset assessment
    }
}
```

Listing 1.3: Simple example metric defined for the R2RLint framework

and import statements excluded) and is available on GitHub[40] under the Apache License[41]. R2RLint was developed using the state-of-the-art software project management tool Maven[42] and the RDF libraries of the Jena project[43].

Even though R2RLint was implemented as a prototype of the R2RLint methodology and the proposed metrics, there were some practical limitations that forced changes leading to certain deviations. These are covered in the following section.

4.1 Implementation Limitations

Although the R2RLint methodology and the proposed metrics are all implementable in theory, there were some hurdles in practice. The major limitations faced during the development of R2RLint and the practical evaluation were hardware resource shortages when running the assessment on big datasets and the complexity in terms of the programming effort required to calculate intermediate results of certain corner cases. Thus, three metrics could not be run on all assessed datasets, or at least not without modifications.

Besides this, there is one case, also shown in Figure 19, where R2RLint differs due to practical reasons: Since the pinpointing mechanism, yielding the actual RDB2RDF mapping rules that most probably generated a given RDF statement, only works on triples or quads, the node metric method `assessNodes(...)` is defined for triple instead of node input. Nonetheless, internally all node metrics implemented assess the subject,

[40] https://github.com/AKSW/Sparqlify-Extensions/tree/patrick/sparqlify-qa
[41] http://www.apache.org/licenses/LICENSE-2.0.html
[42] http://maven.apache.org
[43] http://jena.apache.org

predicate and object node separately, without the need of a triple scope. The whole triple is just used to provide the triple context information, required by the pinpointer to find view definition candidates that led to the erroneous data.

The more severe difference to the formal metric definitions, is that in most cases logical tables used in view definition, that are based on SQL expressions are not evaluated due to the lack of an applicable SQL parser. Even though there are SQL parsers, they either required an amount of memory typically not available on current desktop or notebook computers or they are embedded in larger software libraries and are hard to reuse or not intended for reuse at all. Thus, the SQL parsing is left out in the prototype which means that the evaluation of logical tables based on SQL expressions is skipped. This affects the metrics 2, 10, 16, 35, 36 and 37.

Another case, where the effort to work with up-to-date external data was not made, concerns the *Vocabulary Reuse* (Metric 21) and *Term Reuse* (Metric 20) metric. To determine a quality score, the top 100 ranking of the namespace lookup service prefix.cc is used. Instead of requesting the current top 100 namespaces before running the assessment, the results retrieved Sep 16, 2013 were hard coded. Apart from the effort to write the source code to fetch the desired entries, in some cases results were discarded because they seemed not to be reasonable – a verification step that can hardly be automatable. One example is the entry `<http://dbpedia.org/property/years/>` for the prefix `dbpprop`.

The last case where things left unimplemented, concerns the regular expression used to detect valid HTTP URIs in Metric 42. Even though a big effort was made to integrate different standards, the specifications for Internationalized Domain Names [98] and IPv6 zone identifiers [97] were left out.

5 Evaluation

To get an impression on actual data quality deficiencies of real RDB2RDF mappings, practical assessment runs were performed on three different datasets. These should also serve to proof the proper functioning of the R2RLint framework and to get feedback with regards to the applicability of the implemented metrics. The assessed RDB2RDF mapping projects are introduced in the following, discussing the assessment results of each quality dimension afterwards.

The first data source under assessment is part of the *LinkedGeoData* [70] project, being the Linked Data mirror of OpenStreetMap. LinkedGeoData provides spatial data stemming from crowd-sourced user input covering the whole globe. Since the amount of data is far to much to be assessed as a whole, only a small portion of LinkedGeoData was chosen for evaluation. This portion was created using the OpenStreetMap database snapshot for the smallest of Germany's federal states, Bremen. After having loaded the snapshot, a full RDF dump was created using the Sparqlify tool and the mapping definitions from the LinkedGeoData GitHub repository[44]. This RDF dump was then loaded into a Virtuoso 7.0.0 triple store[45] which was used for the assessment. Event though the RDF dataset of Bremen is just a very small portion of the whole dataset provided by the project, it is referred to as LinkedGeoData in the following for brevity.

LinkedGeoData was chosen as a medium size dataset with RDB2RDF mapping definitions that are expected to have a high quality. This assumption is backed by the fact that LinkedGeoData is part of *GeoKnow*[46], a comprehensive, EU funded research project aiming at connecting heterogeneous spatial data with Semantic Web technologies. General statistics of the LinkedGeoData dataset are shown in Table 8.

Dataset	Date/Version	Triples	Distinct Resources	Literal Values
LinkedGeoData	bremen-latest.osm.pbf[47]	13,726,852	3,726,142	6,200,583
LCC (Eng)	eng_wikipedia_2010_10K	656,704	128,582	149,788
LinkedBrainz	musicbrainz-server-2013-10-14	197,399,205	1,048,239	92,183,398

Table 8: General statistics of the assessed datasets

[44] `https://github.com/GeoKnow/LinkedGeoData/blob/master/linkedgeodata-core/src/main/resources/org/aksw/linkedgeodata/sml/LinkedGeoData-Triplify-IndividualViews.sml`

[45] `http://sourceforge.net/projects/virtuoso/files/virtuoso/7.0.0/`

[46] `http://geoknow.eu/Project.html`

[47] `http://download.geofabrik.de/europe/germany/bremen-latest.osm.pbf`, retrieved Nov 17, 2013

The second dataset that was evaluated is an RDF version of parts of the *Leipzig Corpora Collection (LCC)* provided by the *Wortschatz* project of the University of Leipzig[48]. The dataset[49] contains per-language statistics about co-occurrences of different words stemming from different corpora, e.g. Wikipedia pages or news sites. It was generated ad-hoc to support the creation of multilingual Linked Open Data applications at the *Multilingual Linked Open Data for Enterprises (MLOD)* conference 2012[50]. Being an ad-hoc attempt, created for a very limited purpose, the mapping is expected to be of poor quality. It does not contain much ontological structures, but merely the core statistics. Since the original RDF data is not available anymore, the dataset was rebuilt using the original SML mapping definitions. Even though the data, as created at the MLODE conference, comprised statistics of different languages, only those of the English language were loaded for this assessment. The RDF data was generated using the 10K version of a tab-separated values (TSV) dump[51] holding statistics of words and stemming from 10,000 sentences of the English Wikipedia. After loading each TSV file, again a full RDF dump was created utilizing the Sparqlify tool with the original SML mapping definitions[52]. This RDF dump was then loaded into a Virtuoso 7.0.0 triple store, used for the assessment run. The dataset is referred to as LCC (Eng) in the following and its general statistics can be looked up in Table 8.

The last RDB2RDF mapping project under assessment is *LinkedBrainz* which provides SPARQL access to an RDF version of the MusicBrainz database. Initially funded by the non-departmental public body Jisc[53], LinkedBrainz later became part of the EUCLID[54] EU project. LinkedBrainz is now maintained at the British Museum[55]. Accordingly, this dataset is also expected to be of high quality.

Since the RDB2RDF mapping definitions for the LinkedBrainz data were only available in R2RML, they were translated to SML in the first step. Afterwards an RDF dump was generated using the MusicBrainz database embedded in the MusicBrainz Server Virtual Machine generated on October 14 2013 and the Sparqlify tool. This dump was loaded into a Virtuoso 7.0.0 triple store for the assessment. General statistics of the LinkedBrainz dataset are given in Table 8.

In the following the assessment results for the introduced RDB2RDF mapping projects are discussed, considering single quality dimensions in separate sections. A more detailed overview, showing the outcome of all metrics is given in Appendix B.

---

[48] http://corpora.uni-leipzig.de
[49] http://datahub.io/dataset/lcc
[50] http://sabre2012.infai.org/mlode
[51] http://corpora.uni-leipzig.de/downloads/eng_wikipedia_2010_10K-text.tar.gz
[52] https://github.com/AKSW/Sparqlify/blob/master/sparqlify-examples/src/main/resources/sparqlify-examples/wortschatz-merged.sparqlify
[53] http://jisc.ac.uk/
[54] http://euclid-project.eu/
[55] http://www.britishmuseum.org/

5.1 Availability

The only metric that was evaluated for the availability dimension, was the dereferenceability of generated URIs. For the assessment run, only external URIs were considered.

With regards to the dereferenceability, LinkedGeoData attained a perfect result without any violations.

The only errors found in the LCC dataset were non-dereferenceable URLs pointing to Wikipedia pages that were the actual corpus sources, but do not exist anymore. Since these were also the only external URIs in this dataset, the assessment result of 117 violations within 9,543 assessed Wikipedia URLs amounts to an overall dereferenceability of about 99%.

The main cause of dereferenceability violations of the LinkedBrainz dataset were Discogs URLs like `http://www.discogs.com/artist/AC%2FDC`. Even though they can all be looked up in a browser, trying to retrieve them via the corresponding Java libraries or curl command line queries results in a response '500 Internal Server Error'. These errors amount to 97% of all dereferenceability problems. An actual mapping error could be found via the Dereferenceable URIs metric, where bare integer values were mapped to URIs. Further dereferenceability issues arose for `owl:sameAs` links to different DBpedia datasets.

5.2 Completeness

For the completeness dimension, the schema, property, interlinking and vocabulary completeness were assessed. First of all, the results of the Schema Completeness metric are highly influenced by the implementation limitation, that SQL parsing is not supported. Since view definitions with query based logical tables could thus not be evaluated, the corresponding results are not meaningful at all. The actual scores are in fact much higher than evaluated in the assessment. In case of the LCC dataset, for example, the correct value would be 0.54 instead of 0.04. This clearly shows the need to extend the R2RLint prototype with SQL parsing support.

With the Property Completeness metric a view definition of the LinkedBrainz RDB2-RDF mappings could be detected, that does not generate any triples. Besides this, it can be observed that the LinkedBrainz and LCC datasets are only poorly interlinked. The different results of the vocabulary completeness metrics show, that only in rare cases higher scores are achieved.

5.3 Conciseness

Regarding the conciseness dimension, it can be said, that even though the assessed datasets are perfectly concise with respect to the intentional conciseness, there are often single view definitions that generate multiple different RDF resources, based on single database objects.

An obvious outlier in the results of the LinkedBrainz dataset could be detected for the No Duplicate Statements metric. The value of 0.04 showed that a lot of duplicate triples were introduced. In fact, this was caused by an erroneous mapping, referring to a

wrong relational column[56]. Further, it has to be noted, that in case of the LinkedGeoData and LinkedBrainz dataset, the No Duplicate Statements metric did not assess the whole power set $\mathbb{P}(V)$ of view definitions V, since the power set generation was refused by the underlying library. The corresponding number of view definitions was too big, so that, as a fallback, only single view definitions were considered.

5.4 Consistency

With regards to the consistency metrics it showed that two of them, the Basic Ontology Conformance and the No Resource Name Clashes metric, were not computable for all datasets due to RAM shortages. In the case of the Basic Ontology Conformance metric the LinkedGeoData dataset could only be assessed using sample triples. Nonetheless, these yielded violations for the object property <`http://linkedgeodata.org/page/ontology/wheelchair`> used with literal values, the datatype property <`http://linkedgeodata.org/page/ontology/agricultural`> used with non-literal values, as well as several properties used with a wrong range datatype. Further, LinkedGeoData made statements about *external* resources from the `geodata` and `dbr` namespaces, which are thus considered to be *bad smells* with regards to the No Ontology Hijacking metric. Actual hijacking *violations* were also found, since the LinkedGeoData dataset contained ontological re-definitions concerning the `foaf:mbox` property. But it has to be noted, that only one of these four statements differs from the original definitions of the FOAF vocabulary.

The reported warnings of the No Ambiguous Mappings metric all stem from cases, where references to tables embedded in logical table definitions could not be resolved due to the lack of SQL parsing capabilities. Thus, after validating most of these violations by hand, it turned out, that these are false negatives and the number of errors should be much smaller.

Besides this, no further violations were found. In some cases, this can be attributed to rather poor ontologies, that do not define many consistency restrictions, as in the case of the LCC dataset. The Consistent Foreign Key Resource Identifiers, on the other hand, could not be violated during the assessment of the LinkedGeoData and LinkedBrainz mappings, since no foreign key dependencies were defined on the underlying databases for performance reasons.

5.5 Interlinking

Even though there are great differences between the LinkedGeoData dataset on the one hand, and LCC (Eng) and LinkedBrainz on the other hand, it has to be noted, that the External Same-as Links metric creates noticeable low scores. Nonetheless it can be seen, that LinkedGeoData is better interlinked than LinkedBrainz, and LCC (Eng) only provides a very small portion of `owl:sameAs` links.

[56] The object map in the mapping `https://github.com/LinkedBrainz/MusicBrainz-R2RML/blob/de10106bde0ae0c14b2a7e51baac49abc7dcd823/mappings/artist.ttl#L212-L224` erroneously refers to `gid` instead of `recording_gid`

5.6 Interoperability

With regards to their interoperability, the LinkedGeoData and LinkedBrainz datasets clearly outperform the LCC (Eng). Whereas LinkedGeoData and LinkedBrainz have a similar score for the Term Reuse metric, LinkedBrainz has the more comprehensive vocabulary reuse.

5.7 Interpretability

The interpretability concerns under assessment comprise the typing of resources, the anchoring of resources in ontological structures, the avoidance of blank nodes and the correct usage of more complex RDF structures, like collections, containers or reification. An obvious quality deficiency with regards to the typing and the provision of an ontological context for classes and properties, can be determined for the LCC (Eng) dataset. Thus, from a formal semantic perspective, for the most of the contained resources it is not clear, if they are instances, classes or properties.

Besides this, it could be detected, that in LinkedBrainz certain resources are not typed. The resources that are explicitly excluded from the type assignments in the RDB2RDF mappings are MusicBrainz release events that are not dated with a year, month *and* day. Nonetheless, since the LinkedBrainz RDB2RDF mappings also generate release event resources, that are just dated with a year or a year and month, this seems to be an error, especially because all other introduced resources are typed.

It further turned out, that the only resources that did not have an ontological context, as measured by the OWL Ontology Declarations metric, were those that were not typed. Thus, the more general OWL Ontology Declarations metric did not find any violations, other than those already reported by the Typed Resources metric.

Another significant error pattern was detected for the LinkedGeoData mappings. There, the first container member is declared using the container membership property `rdf:_0` instead of `rdf:_1`.

5.8 Performance

The only performance aspect, considered relevant for the assessment of RDB2RDF mappings, was the introduction of local hash URIs. With respect to the view that hash URIs should be avoided, the LinkedBrainz dataset is of bad quality, since all local URIs are designed to contain the hash sign. Nonetheless, nearly all of them have the fixed fraction part `#_`. Thus, there are usually no two resources sharing a non-fraction part. Accordingly, the argumentation, that hash URIs would harm the performance does not hold in this case.

5.9 Relevancy

The assessment of the relevancy dimension comprises the classification of the datasets with regards to their triple counts as well as the detection of two different coverage values. With regards to the triple counts, LinkedGeoData and LinkedBrainz are considered of good quality, whereas LCC (Eng) has only medium quality.

The coverage metrics yielded noticeable low scores, which nonetheless do not seem to reflect a bad quality of the assessed datasets, but rather should be normal for datasets of a certain size. The coverage scores of the LinkedGeoData dataset are considerably higher than those of the other datasets, with LinkedBrainz having the lowest quality with regards to the combination of detail and scope. For the Coverage (Scope) metric, evaluated with the LCC (Eng) dataset, the worst possible quality score was assigned. This is also not considered to reveal its actual quality, but has to be attributed to the missing type information. Since the dataset does not contain type statements, there are no explicitly defined instances, which have a direct influence on the calculated score.

5.10 Representational Conciseness

The assessment of the representational conciseness dimension comprises the check, if short and query parameter-free URIs are introduced, and if so called prolix features were avoided. With regards to the URI length, it has to be noted, that only a smaller portion of the very long URIs can be attributed to a bad URI design. Instead many of the violations are URIs that contain a lot of special characters, being percent-encoded. With this respect, resource identifiers based on characters from writing systems not allowed within URIs, have a clear disadvantage. The only exception, where URIs were considerably long by design was given in the RDB2RDF mappings of the LinkedBrainz dataset. There, URIs were generated that hold two UUID[57] strings, each having 36 characters.

The only URIs found in the assessment, that contain query parameters were introduced by the mappings of the LinkedGeoData dataset. Since the corresponding resources were external, built to express interlinks, these are not considered as an indication of bad quality.

The only prolix features were also found in the LinkedGeoData dataset. There containers are used to express node paths. Since such node paths have to be expressed as an ordered list, it is doubtful if the usage of containers has to be considered as an indicator of bad quality.

5.11 Semantic Accuracy

The metrics assessing the semantic accuracy of RDB2RDF mappings all refer to certain characteristics of the relational schema definitions of the underlying database. Since a view definition's logical table was not assessed, if it is based on an SQL query, all these metrics were affected by the missing SQL parsing support of the R2RLint prototype. Nonetheless, with respect to the tables, that could be assessed, a considerable number of inaccuracies could be found. Thus it can be generally stated, that there was certain semantic information contained in the corresponding relational databases, that was not considered in the RDB2RDF mappings under assessment. On the other hand, it has to be noted, that gathering all these (partly implicit) relational constraints is rather cumbersome if it is done by hand. Thus, the R2RLint tool could be extended to propose the corresponding mappings based on an automatic schema evaluation.

[57] `http://www.opengroup.org/dce/info/draft-leach-uuids-guids-01.txt`

5.12 Syntactic Validity

The aspects, assessed with regards to the syntactic validity dimension, were the use of valid datatypes and language tags. The only violation found, was the invalid typing of date information as `xsd:dateTime`. The corresponding RDB2RDF mapping definition stems from the LCC (Eng) project.

Even though, this shows a good quality with regards to the syntactic validity of the assessed datasets, the R2RLint tool could again be used to make guiding suggestions. These concern the datatype to use, in cases where values from relational tables where transformed to typed literals without any modifications. In such cases, the datatype of the underlying schema could be used to propose an XSD datatype.

5.13 Understandability

The understandability dimension was assessed, checking if resources are labeled, whether their URIs are sounding and valid HTTP URLs and if certain metadata are provided. All of the three datasets contained a considerable number of resources that are not labeled and not sounding. With regard to sounding URIs, again a notable portion of the violating URIs contain percent-encoded strings. It further has to be noted, that the training corpus of the Sounding URIs metric stemmed from English Wikipedia sources. Thus, language specific resource names, like for example Czech artists from the LinkedBrainz dataset, might have gotten a lower score than they could have, if they were assessed using a training corpus of their native language. Apart from this, the Sounding URIs showed also some weaknesses. Besides the fact, that some relatively short, but sounding URIs were reported as violations with respect to the configured threshold, there were also URIs that are obviously not sounding, but got sufficiently high scores. These are for example the URIs containing two UUID strings, e.g. `http://musicbrainz.org/release/3b52a520-88b8-4ecb-bbf7-2168ab6c9499#-489ce91b-6658-3307-9877-795b68554c98`. A proposed deviation of the underlying metric would be to just consider the local part of the URI, omitting the URI namespace. Thus, in the example above just the string of hex symbols with dashes would be assessed – a combination that is not likely to appear in natural languages. A further improvement would be to decode percent-encoded URIs before assessing them, to reduce the bias of giving preference to URIs expressible in ASCII characters.

The URIs reported because they are not valid HTTP URLs, mainly contained certain characters that should have been percent-encoded, e.g. the colon character in `http://dbpedia.org/resource/Che:_Chapter_127`.

The metric assessing, whether dataset metadata is contained within the dataset under assessment, yielded a score of 0 for all datasets. One interpretation of these results could be, that it is uncommon to embed such metadata in the actual dataset. In fact, there is a corresponding W3C Interest Group Note, proposing a deployment of VoiD information *"alongside a dataset"* [99]. So even though there are datasets with embedded VoiD metadata, the provision of such information might not be assessable this way in general.

6 Conclusions and Future Work

In this report, a quality assessment methodology as well as aspects to consider for an RDB2RDF quality assessment were developed systematically. After a comprehensive survey of literature sources covering information and data quality, a set of dimensions suitable for the quality assessment of RDB2RDF mappings were compiled. Each quality dimension was substantiated with a set of quality assessment metrics that were introduced formally.

Besides the formal and conceptual considerations, a software prototype was developed, implementing the assessment methodology framework and proposed metrics. In practical assessment runs on three different datasets, generated via RDB2RDF mappings, the software could extract clear characteristics with regards to the considered dataset. The provision of actual quality scores allowed a comparison of the three datasets, general judges on their quality and, most of all, showed actual mapping errors. Making deficiencies measurable and visible further enables the data providers to improve their mappings and fix errors. Thus, the overall goal to provide effective means for a quality assurance of RDB2RDF mappings could be accomplished.

Apart from this, the developed prototype also showed directions for further improvements. The major drawback was, that the computation of some metrics took impractically long time or was not feasible at all due to memory shortages. This scalability problem occurred mainly during the computation of metrics requiring dataset scope. Thus, one future task will be to put effort into the transformation of dataset metrics to view metrics. Some suggestions in this respect were already given in Section 3.5.

Moreover, the practical assessment showed the strong need of an SQL parser to be able to also evaluate logical tables used in the mapping definitions that are expressed as SQL queries. Since the lack of this capability led to a considerable high number of false results, the corresponding extension of the R2RLint prototype is of high importance.

Another future step will be to improve the presentation of the assessment results. Currently the only implemented, practically relevant assessment sink writes the quality scores and the corresponding metadata to a relational database. Since the sink produces a quite complex database schema, the exploration capabilities of the results are weak.

Besides the actual assessment of existing mapping definitions, the prototype could also be extended to make mapping suggestions, which improve the overall quality. Thus, a further vision would be to use the R2RLint tool as back-end for an RDB2RDF editing workbench, which interactively guides RDB2RDF mapping authors to optimize the mapping's quality.

A Auxiliary Definitions

This section contains several auxiliary definitions referred to in the actual metrics definitions in Section 3.5. All symbols and variables used in the auxiliary definitions were introduced and defined in the sections 3.3 and 3.5.

A.1 Metric 14 (No Bogus Inverse-functional Properties)

Blacklist of statements expressing bogus inverse-functional properties ($?s \in Q$):

$$
\begin{aligned}
&T_{bogus} = \\
&\{(?s, \texttt{<http://xmlns.com/foaf/0.1/mbox\_sha1sum>}, \textit{"08445a31a78661b5c746feff39a9db6e4e2cc5cf"}), \\
&\ (?s, \texttt{<http://xmlns.com/foaf/0.1/mbox\_sha1sum>}, \textit{"da39a33ee5e6b4b0d3255bfef95601890afd80709"}), \\
&\ (?s, \texttt{<http://xmlns.com/foaf/0.1/homepage>}, \texttt{<http://>}), \\
&\ (?s, \texttt{<http://xmlns.com/foaf/0.1/mbox\_sha1sum>}, \textit{""}), \\
&\ (?s, \texttt{<http://xmlns.com/foaf/0.1/isPrimaryTopicOf>}, \texttt{<http://>})\}
\end{aligned}
$$

A.2 Metric 23 (OWL Ontology Declarations)

List of proposed properties providing ontological context of an RDF resource ($ONTDEFPROPERTIES \subset \mathcal{R}$):

$$
\begin{aligned}
ONTDEFPROPERTIES = \{&\texttt{rdf:type}, \texttt{rdfs:subClassOf}, \texttt{rdfs:subPropertyOf}, \texttt{rdfs:domain}, \\
&\texttt{rdfs:range}, \texttt{owl:complementOf}, \texttt{owl:disjointWith}, \\
&\texttt{owl:equivalentClass}, \texttt{owl:equivalentProperty}, \\
&\texttt{owl:intersectionOf}, \texttt{owl:inverseOf}, \texttt{owl:oneOf}, \texttt{owl:unionOf}\}
\end{aligned}
$$

A.3 Metric 42 (HTTP URIs)

Definition of the regular expression intended to find valid HTTP URIs, as taken from the Java Source code of R2RLint:

```
public static final String httpUrlPattern = "^" +
       // protocol: http:// or https://
       "(?:(?:https?)://)" +
       // user info, e.g. user@ or user:passwd@
       "(?:\\S+(?::\\S*)?@)?" +
       // host part, e.g. localhost, aksw.org, 127.0.0.1
       "(?:" +
       // IP address based host names, like 193.239.40.138

           // exclude host names based on local IP addresses because they
           // cannot be resolved in the WWW
           // 10.x.x.x
           "(?!10(?:\\.\\d{1,3}){3})" +
           // 127.x.x.x
           "(?!127(?:\\.\\d{1,3}){3})" +
```

```
            // 169.254.x.x
            "(?!169\\.254(?:\\.\\d{1,3}){2})" +
            // 172.16.0.0/12 (172.16.0.0 to 172.31.255.255)
            "(?!172\\.(?:1[6-9]|2\\d|3[0-1])(?:\\d{1,3}){2})" +
            // 192.168.x.x
            "(?!192\\.168(?:\\.\\d{1,3}){2})" +

            // all remaining and valid IP addresses:
                // first octet:
                //  1-99            1xx          2xx up to 223
                "(?:[1-9]\\d?|" + "1\\d\\d|" + "2[01]\\d|22[0-3])" +
                // second and third octet
                //       0-99            1xx          2xx up to 255
                "(?:\\.(?:\\d{1,2}|" + "1\\d\\d|" + "2[0-4]\\d|25[0-5])){2}" +
                // fourth octet
                // omitting network (x.x.x.0) and broadcast (x.x.x.255)
                // addresses
                //       1-99            1xx            2xx up to 254
                "(?:\\.(?:[1-9]\\d?|" + "1\\d\\d|" + "2[0-4]\\d|25[0-4]))" +
        "|" +
        // domain name based host names like aksw.org or
        // mail.informatik.uni-leipzig.de

            // TODO: add support for internationalized domain names

            // domain name
            // restrictions: only one hyphen *between* two chars; a char can
            // be a letter or digit
            "(?:(?:(?:[a-zA-Z0-9]-?)*(?:[a-zA-Z0-9])+\\.)+)" +
            // TLD identifier
            "(?:[a-z]{2,})" +
        ")" +
        // port number
        "(?::\\d{2,5})?" +

        // path
        //
        // according to
        // http://tools.ietf.org/html/draft-fielding-url-syntax-09#appendix-A :
        // path          = [ "/" ] path_segments
        // path_segments = segment *( "/" segment )
        // segment       = *pchar *( ";" param )
        // param         = *pchar
        // pchar         = unreserved | escaped | ":" | "@" | "&" | "=" | "+"
        // unreserved    = alpha | digit | mark
        // escaped       = "%" hex hex
        // alpha         = lowalpha | upalpha
        // mark          = "$" | "-" | "_" | "." | "!" | "~" |
        //                 "*" | "'" | "(" | ")" | ","
        "(?:(?:/([a-zA-Z\\d_~',\\Q$-.!*()\\E]|%[a-fA-F\\d]{2})*)*)" +

        // opaque URLs not considered here


        // query
        //
```

```
72          // http://tools.ietf.org/html/draft-fielding-url-syntax-09#
                appendix-A:
73          //
74          // rel_path      = [ path_segments ] [ "?" query ]
75          // query         = *urlc
76          // urlc          = reserved | unreserved | escaped
77          // reserved      = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+"
78          // unreserved    = alpha | digit | mark
79          // escaped       = "%" hex hex
80
81          // http://tools.ietf.org/html/rfc3986#page-23:
82          //
83          // query      = *( pchar / "/" / "?" )
84          // pchar      = unreserved / pct-encoded / sub-delims / ":" / "@"
85          // unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"
86          // sub-delims = "!" / "$" / "&" / "'" / "(" / ")"
87          //                / "*" / "+" / "," / ";" / "="
88          "(?:\\?" +
89              "(?:" +
90                  // field
91                  "(?:([a-zA-Z\\d;/:_~',\\Q-?@$+*.!()\\E]|%[a-fA-F\\d]{2}))
                        +" +
92                  // =
93                  "(?:=" +
94                  // value &
95                  "(?:([a-zA-Z\\d;/:_~',\\Q-?@$+*.!()\\E]|%[a-fA-F\\d]{2}))
                        +)?[&;]" +
96              ")*" +
97
98              "(?:" +
99                  // field
100                 "(?:([a-zA-Z\\d;/:_~',\\Q-?@$+*.!()\\E]|%[a-fA-F\\d]{2}))
                        +" +
101                 // =
102                 "(?:=" +
103                 // value
104                 "(?:([a-zA-Z\\d/:_~',\\Q;-?@$+*.!()\\E]|%[a-fA-F\\d]{2}))
                        +)?" +
105             ")" +
106         ")?" +
107
108         // fragment
109         "(?:#(?:([a-zA-Z\\d/:_~',=&\\Q;-?@$+*.!()\\E]|%[a-fA-F\\d]{2}))*)
                ?" +
110         "$";
```

Listing 1.4: Regular expression to detect HTTP URIs, defined in Java source code

A.4 Metric 43 (Dataset Metadata)

$$R_{title} = \{\texttt{dcterms:alternative}, \texttt{dcterms:title}, \texttt{dc:title}, \texttt{sioc:name}\}$$

$$R_{content} = \left\{ \begin{array}{l} \texttt{dc:coverage}, \texttt{dc:description}, \texttt{dc:language}, \texttt{dc:source}, \texttt{dc:subject}, \texttt{dc:type}, \\ \texttt{dcterms:abstract}, \texttt{dcterms:accrualMethod}, \texttt{dcterms:accrualPeriodicity}, \\ \texttt{dcterms:accrualPolicy}, \texttt{dcterms:audience}, \texttt{dcterms:available}, \texttt{dcterms:coverage}, \\ \texttt{dcterms:description}, \texttt{dcterms:language}, \texttt{dcterms:provenance}, \texttt{dcterms:source}, \\ \texttt{dcterms:spatial}, \texttt{dcterms:subject}, \texttt{dcterms:tableOfContents}, \texttt{dcterms:type}, \\ \texttt{foaf:primaryTopic}, \texttt{foaf:topic}, \texttt{sioc:about}, \texttt{sioc:has\_space}, \texttt{sioc:topic}, \\ \texttt{void:classPartition}, \texttt{void:classes}, \texttt{void:class}, \texttt{void:dataDump}, \\ \texttt{void:distinctObjects}, \texttt{void:distinctSubjects}, \texttt{void:documents}, \texttt{void:entities}, \\ \texttt{void:exampleResource}, \texttt{void:feature}, \texttt{void:inDataset}, \texttt{void:linkPredicate}, \\ \texttt{void:objectsTarget}, \texttt{void:openSearchDescription}, \texttt{void:properties}, \\ \texttt{void:propertyPartition}, \texttt{void:property}, \texttt{void:rootResource}, \texttt{void:sparqlEndpoint}, \\ \texttt{void:subjectsTarget}, \texttt{void:subset}, \texttt{void:target}, \texttt{void:triples}, \\ \texttt{void:uriLookupEndpoint}, \texttt{void:uriRegexPattern}, \texttt{void:uriSpace}, \texttt{void:vocabulary} \end{array} \right\}$$

$$R_{creator} = \left\{ \begin{array}{l} \texttt{dc:contributor}, \texttt{dc:creator}, \texttt{dc:publisher}, \texttt{dcterms:contributor}, \\ \texttt{dcterms:creator}, \texttt{dcterms:publisher}, \texttt{foaf:maker}, \texttt{sioc:has\_creator} \end{array} \right\}$$

B Evaluation Results

B.1 Availability

Dataset	Metric 1
LinkedGeoData	0
LCC (Eng)	117
LinkedBrainz	239,924‡

Table 9: Assessment results of the availability dimension metric:
Metric 1: Dereferenceable URIs.
The table shows the number of violations with a disabled threshold. Values marked with ‡ are projections based on sample data.

B.2 Completeness

Dataset	Metric 2*	Metric 3	Metric 4*	Metric 5
LinkedGeoData	0.30	2.62	(0.02/0.94/1.00)	0.54
LCC (Eng)	0.04	2.81	(1.00/1.00/1.00)	0.08
LinkedBrainz	0.02	0.69	(0.00/0.88/1.00)	0.03

Table 10: Assessment results of the completeness dimension metrics:
Metric 2: Schema Completeness,
Metric 3: Population Completeness,
Metric 4: Property Completeness,
Metric 5: Interlinking Completeness.
The table shows the quality scores of the corresponding metrics. The scores of metrics marked with † are affected by implementation limitations (cf. Section 4.1) and might thus in fact be higher. The content of columns marked with ‡ represents the (minimum/average/maximum) of the metric's values with respect to the given dataset.

Vocabulary	Metric 6	Metric 7
http://geovocab.org/geometry#	0.11	0.09
http://geovocab.org/spatial#	1.00	0.00
http://purl.org/dc/terms/	0.05	0.07
http://www.opengis.net/ont/geosparql#	0.00	0.03
http://www.w3.org/1999/02/22-rdf-syntax-ns#	0.17	0.14
http://www.w3.org/2000/01/rdf-schema#	0.00	0.44
http://www.w3.org/2002/07/owl#	0.12	0.02
http://www.w3.org/2003/01/geo/wgs84_pos#	0.00	0.40
http://www.w3.org/2004/02/skos/core#	0.00	0.04
http://xmlns.com/foaf/0.1/	0.00	0.03

Table 11: Assessment results of the completeness dimension metrics
Metric 6: Vocabulary Class Completeness,
Metric 7: Vocabulary Property Completeness
applied to the LinkedGeoData dataset

Vocabulary	Metric 6	Metric 7
http://www.w3.org/1999/02/22-rdf-syntax-ns#	0.00	0.14
http://www.w3.org/2000/01/rdf-schema#	0.00	0.22
http://www.w3.org/2002/07/owl#	0.00	0.08

Table 12: Assessment results of the completeness dimension metrics
Metric 6: Vocabulary Class Completeness,
Metric 7: Vocabulary Property Completeness
applied to the LCC (Eng) dataset

Vocabulary	Metric 6	Metric 7
http://www.w3.org/1999/02/22-rdf-syntax-ns#	0.17	0.43
http://www.w3.org/2002/07/owl#	0.65	0.45
http://www.w3.org/2003/01/geo/wgs84_pos#	0.50	0.00
http://www.w3.org/2004/02/skos/core#	0.00	0.04
http://xmlns.com/foaf/0.1/	0.00	0.08
http://purl.org/dc/elements/1.1/	—	0.13
http://open.vocab.org/terms/	0.00	0.01
http://purl.org/ontology/mo/	0.20	0.08
http://purl.org/NET/c4dm/event.owl#	0.00	0.11

Table 13: Assessment results of the completeness dimension metrics
Metric 6: Vocabulary Class Completeness,
Metric 7: Vocabulary Property Completeness
applied to the LinkedBrainz dataset

B.3 Conciseness

Dataset	Metric 8	Metric 9	Metric 10
LinkedGeoData	(1.00/1.00/1.00)	(0.20/0.95/1.00)	(0.94/0.99/1.00)*
LCC (Eng)	(1.00/1.00/1.00)	(0.50/0.97/1.00)	(0.15/0.92/1.00)
LinkedBrainz	(1.00/1.00/1.00)	(0.33/0.93/1)	(0.04/0.99/1.00)*

Table 14: Assessment results of the conciseness dimension metrics:
Metric 8: Intensional Conciseness,
Metric 9: Extensional Conciseness,
Metric 10: No Duplicate Statements.
The table shows the (minimum/average/maximum) of the quality scores of the corresponding metrics. Values marked with * are affected by implementation limitations (cf. Section 4.1).

B.4 Consistency

Metric	Part.	Whole, est.
Correct Datatype Property Value	9	18
Correct Object Property Value	2,942	5,884
Disjoint Classes Conformance	0	0
Correct Datatype Range	47,773	95,546

Table 15: Assessment results of the consistency dimension metric
Metric 11: Basic Ontology Conformance
applied to the LinkedGeoData dataset. Due to memory limitations only sample data of the whole dataset was assessed (cf. Section 4.1). The number of violations w.r.t. a sub-metric and with a disabled threshold are presented in column *Part.*. A simple projection of these numbers to the whole dataset are given in column *Whole, est.*.

Metric	Whole
Correct Datatype Property Value	0
Correct Object Property Value	0
Disjoint Classes Conformance	0
Correct Datatype Range	0

Table 16: Assessment results of the consistency dimension metric
Metric 11: Basic Ontology Conformance
applied to the LCC (Eng) dataset. The number of violations w.r.t. a sub-metric and with a disabled threshold are presented in column *Whole*.

Dataset	Metric 12	Metric 13 C	Metric 13 P	Metric 14	Metric 15	Metric 16	Metric 17	Metric 18
LinkedGeoData	0	0	0	0	(602,151/4)	13*	0	0
LCC (Eng)	0	0	0	0	(0/0)	4*	0	0
LinkedBrainz	3	0	0	0	(0/0)	12*	—	0

Table 17: Assessment results of the consistency dimension metrics:
Metric 12: Homogeneous Datatypes,
Metric 13: No Deprecated Classes or Properties,
Metric 14: No Bogus Inverse-functional Properties,
Metric 15: No Ontology Hijacking,
Metric 16: No Ambiguous Mappings,
Metric 17: No Resource Name Clashes,
Metric 18: Consistent Foreign Key Resource Identifiers.
The table shows the number of violations with a disabled threshold where the columns labeled with *C* refer to the number of violating classes and the number of violating properties are given in the columns labeled with *P*. Values marked with * are affected by implementation limitations (cf. Section 4.1) and may thus in fact be smaller. The values of Metric 15 are given as value pair, where the first entry represents the number of bad smells and the second entry the number of violations.

B.5 Interlinking

Dataset	Metric 19
LinkedGeoData	0.044
LCC (Eng)	~0.000
LinkedBrainz	0.001

Table 18: Assessment results of the interlinking dimension metric
Metric 19: External Same-as Links.

B.6 Interoperability

Dataset	Metric 20	Metric 21
LinkedGeoData	0.41	0.56
LCC (Eng)	0.13	0.25
LinkedBrainz	0.47	0.79

Table 19: Assessment results of the interoperability dimension metrics:
Metric 20: Term Reuse,
Metric 21: Vocabulary Reuse.

B.7 Interpretability

Dataset	Metric 22	Metric 23	Metric 24	Metric 25	Metric 26	Metric 27
LinkedGeoData	546,154	546,154	0	0	140,737	0
LCC (Eng)	75,837	75,837	0	0	0	0
LinkedBrainz	526,529	526,529	0	0	0	0

Table 20: Assessment results of the interpretability dimension metrics:
Metric 22: Typed Resources,
Metric 23: OWL Ontology Declarations,
Metric 24: Avoid Blank Nodes,
Metric 25: Correct Collection Use,
Metric 26: Correct Container Use,
Metric 27: Correct Reification Use.
The table shows the number of violations with a disabled threshold.

B.8 Performance

Dataset	Metric 28
LinkedGeoData	0
LCC (Eng)	2
LinkedBrainz	397,343,729

Table 21: Assessment results of the performance dimension metric
Metric 28: No Hash URIs.
The table shows the number of violations with a disabled threshold.

B.9 Relevancy

Dataset	Metric 29	Metric 30	Metric 31
LinkedGeoData	0.75	0.000351	0.017343
LCC (Eng)	0.50	0.000171	0.000000
LinkedBrainz	0.75	0.000001	~0.000000

Table 22: Assessment results of the relevancy dimension metrics:
Metric 29: Amount of Triples,
Metric 30: Coverage (Detail),
Metric 31: Coverage (Scope).
The table shows the quality scores of the corresponding metrics.

B.10 Representational Conciseness

Dataset	Metric $32_{(75)}$	Metric $32_{(95)}$	Metric 33	Metric 34
LinkedGeoData	747,934	34	(0/1,274,822/0)	747,127
LCC (Eng)	13,807	32	(0/0/0)	0
LinkedBrainz	4,103,053	2,829,088	(0/0/0)	0

Table 23: Assessment results of the representational conciseness metrics:
Metric 32: Short URIs,
Metric 33: No Prolix Features,
Metric 34: Query Parameter-free URIs.
The table shows the number of violations with a disabled threshold, except in case of Metric 32 (Short URIs). For this metric two results are presented: Metric $32_{(75)}$ with a URI length threshold of 75 and Metric $32_{(95)}$ with a URI length threshold of 95 characters. The values shown for Metric 33 refer to the number of the RDF reification, RDF container and RDF collection statements, respectively.

B.11 Semantic Accuracy

Dataset	Metric 35	Metric 36	Metric 37
LinkedGeoData	7^*	15^*	0^*
LCC (Eng)	0^*	2^*	0^*
LinkedBrainz	17^*	19^*	0^*

Table 24: Assessment results of the accuracy dimension metrics:
Metric 35: Preserved NOT NULL Constraints,
Metric 36: Preserved Functional Attributes,
Metric 37: Preserved Foreign Key Constraints.
The table shows the number of violations with a disabled threshold. Values marked with * are affected by implementation limitations (cf. Section 4.1) and may thus in fact be greater.

B.12 Syntactic Validity

B.13 Understandability

Dataset	Metric 38	Metric 39
LinkedGeoData	0	0
LCC (Eng)	9,543	0
LinkedBrainz	0	0

Table 25: Assessment results of the accuracy dimension metrics:
Metric 38: Datatype-Compatible Literals,
Metric 39: Valid Language Tags.
The table shows the number of violations with a disabled threshold.

Dataset	Metric 40	Metric 41	Metric 42	Metric 43
LinkedGeoData	1,007,987	66,778	21,980	0
LCC (Eng)	77,592	2	53,968	0
LinkedBrainz	1,019,219	1,019,001	1,199,716	0

Table 26: Assessment results of the understandability dimension metrics:
Metric 40: Labeled Resources,
Metric 41: Sounding URIs,
Metric 42: HTTP URIs,
Metric 43: Dataset Metadata.
The table shows the number of violations with a disabled threshold, except for Metric 41 (Sounding URIs). This metric's threshold is adjusted to accept the score of rdf:type. In case of Metric 43 the actual score is shown.

C Data Quality Dimensions Overview

This overview shows the data quality dimensions found in the literature sources. The table contains not all dimensions *mentioned* but these that were actually *proposed* by the corresponding publication. This means, that in publications where first dimensions from all the considered literature sources were collected and in a second step shortlisted according to the given use case, only the shortlisted dimensions will appear in the table.

When possible formulas are shown to calculate a score of the given dimension. For this sake identifiers are used, defined below:

β	sensitivity parameter chosen by the user
$attr_{data}$	set of unique attributes of data in a data source
$attr_{dom}$	set of unique attributes of individuals in the considered domain
currency	see the *Currency* dimension in the overview table

$C(t)$	function that estimates the given value completeness at a point in time t ($time_{pub} \leq t \leq time_{max}$)
D_{corr}	set of data values that are correct (or *'accurate'*)
D_{err}	set of data values that are erroneous
D_{ideal}	set of data values reflecting the modeled domain without any errors
$D_{non-null}$	set of data values not being `NULL`
D_{real}	set of all data values as given in the considered data source
$F_{exp}(t)$	probability distribution function of the probability that for a given point in time t holds: $t = time_{exp}$
obj_{data}	set of unique objects (individuals of the modeled domain) stored in a data source
obj_{dom}	set of unique objects (individuals) of the considered domain
$obj_{\|prop}$	set of all unique objects (individuals of the modeled domain) with a property *prop*, stored in the data source
$obj_{\|prop_{uniq}}$	set of all unique objects (individuals of the modeled domain) having a unique value for a property *prop*, stored in the data source
$obj_{\|prop_{cons}}$	set of all unique objects (individuals of the modeled domain) not having any conflicts for a property *prop*, stored in the data source
$time_{curr}$	the current point in time
$time_{exp}$	the point in time when the data expires
$time_{last\_update}$	the point in time when the last update of the data source occurred
$time_{max}$	the latest point in time the whole system is observed
$time_{next\_change}$	the point in time when the next change of the underlying modeled domain (real world) will occur
$time_{next\_update}$	the point in time when the next update of the data source will occur
$time_{pub}$	the point in time when the data was published
$period\_of\_validity$	see the *Period of validity* dimension in the overview table

Apart from these, abbreviations were used in citations quoted literally. These are: *IS* (information system), *RW* (real world) and *NSI* (National Statistical Institute).

Within a dimension entry, the corresponding descriptions are sorted as follows: Entries with no description (marked with '—') are put on top, followed by descriptions sorted by year (ascending). If there are multiple literature sources, they are noted in chronological order as well. If authors referred to dimensions under a different name, this is marked as *here 'Different name'* . In case of specializations or sub-dimensions these are noted un-italicized at the beginning of the description.

Dimension	Definition	Source
Ability to represent null values	*"Ability to distinguish neatly (without ambiguities) null and default values from applicable values of the domain"*	[41]
Accessibility	—	[40,100] [73,45] [39,76]
	Schema: *"Is the schema definition accessible by the users?"*	[101]

Accessibility (cont.)	Type: *"Is the type visible and accessible for users?"*	[101]
	Agent: *"Is the network sufficient for delivered data?"*	[101]
	Data Store: *"Is the data store accessible?"*	[101]
	"extent to which information is available, or easily and quickly retrievable"	[46,81]
	"physical conditions under which users can obtain data: where to go, how to order, delivery time, clear pricing policy, convenient marketing conditions (copyright, etc.), availability of micro or macro data, various formats (paper, files, CD-ROM, Internet etc.) etc."	[102]
	"refers to the proper functioning of all access methods"	[64]
Accuracy	—	[40,100] [73,39] [34]
	"Distance between v and v′, considered as correct"	[41]
	Agent: *"Number of delivered accurate tuples"*	[101]
	Data Store: *"Level of preciseness; Number of accurate tuples"*	[101]
	$\frac{\lvert D_{corr}\rvert}{\lvert D_{real}\rvert}$	[29,103]
	Known: *"True or error-free w/respect to some known value"*	[104]
	Assigned: *"True or error-free w/respect to some designated or assigned value"*	[104]
	Measured: *"True or error-free w/respect to a measured value"*	[104]
	"The extent to which collected data are free of measurement errors."	[37]
	$1 - \frac{\lvert D_{err}\rvert}{\lvert D_{real}\rvert}$	[76]
	"closeness of computations or estimates to the (unknown) exact or true values"	[102]
	"degree of correctness and precision with which information in an information system represents states of the real world"	[81]
	distinction: *syntactic accuracy* vs. *semantic accuracy*	[44,43] [67]
	here: 'Semantic accuracy': "degree to which data values correctly represent the real world facts"	[67]
Aesthetics	—	[105]
Amount of data	—	[100]
	here 'Appropriate amount of data'	[40,45]
	"size of the query result, measured in bytes"	[103]
	here 'Appropriate amount of data': "The extent to which the volume of information is appropriate for a specific theory."	[37]
	"extent to which the volume of data is appropriate for the task at hand"	[46,81] [64]
Applicability	—	[73]
Appropriateness	—	[76]
	"One format is more appropriate than another if it is more suited to the user needs"	[41]
Arrangement	—	[76]
Availability	Schema: *"Frequency of updates"*	[101]
	Type: *"Frequency of updates"*	[101]
	Agent: *"Response time"*	[101]

Availability (cont.)	Data store: *"Uptime of data store, response time"*	[101]
	"probability that a feasible query is correctly answered in a given time range"	[103]
	"extent to which information is physically accessible"	[77]
	"extent to which data (or some portion of it) is present, obtainable and ready for use"	[67]
Believability	—	[40,100] [45,76]
	"degree to which the data is accepted as correct by the user"	[103]
	"extent to which information is regarded as true and credible"	[46,81]
Clarity	—	[73]
	"The extent to which data contain no fuzzy and ambiguous observations."	[37]
	"the data's information environment whether data are accompanied with appropriate documentation and metadata, illustrations such as graphs and maps, whether information on their quality is also available (including limitation in use etc.) and the extent to which additional assistance is provided by the NSI"	[102]
Coherence	*"adequacy to be reliably combined in different ways and for various uses"*	[102]
Comparability	*"impact of differences in applied statistical concepts and measurement tools/procedures when statistics are compared between geographical areas, non-geographical domains, or over time"*	[102]
Completability	$\int_{time_{curr}}^{time_{max}} C(t)\,dt$	[106]
Completeness	—	[107,40] [100,45] [39,34]
	here 'Spurious'	[34]
	"Degree to which values are present in a data collection"	[41]
	here 'Complete': *"Improper representation: missing IS states"*	[32]
	$\frac{\lvert D_{real} \cap D_{ideal} \rvert}{\lvert D_{ideal} \rvert}$	[38]
	Model: *"Level of covering, number of represented business rules"*	[101]
	Concept: *"Number of missing attributes; Are the assertions related to the concept complete?"*	[101]
	Schema: *"Number of missing entities wrt. conceptual model"*	[101]
	Type: *"Number of missing attributes wrt. conceptual model"*	[101]
	Agent: *"Number of tuples delivered wrt. expected number"*	[101]
	Data Store: *"Number of stored null values where there are not expected"*	[101]
	"All required parts present; all attributes needed are present; no missing records; some tolerance for missing values"	[104]
	$\frac{\lvert D_{non-null} \rvert}{\lvert D \rvert}$	[103]
	"All values that are supposed to be collected as per a collection theory are collected."	[37]
	"degree to which information is not missing"	[46,81]
	extensional: *coverage, completeness of entities*	[108]
	intensional: *density, completeness of attributes*	[108]

Completeness (cont.)	value completeness: *"capture the presence of null values for some attributes of tuples"*	[109]
	tuple completeness: *"characterize the completeness of a whole tuple with respect to the values of all attributes"*	[109]
	attribute completeness: *"measure the number of null values of a specific attribute in a relation"*	[109]
	relation completeness: *"captures the presence of null values in the whole relation"*	[109]
	Schema completeness: *"degree to which entities and attributes are not missing in a schema"*	[81,43]
	Column completeness: *"function of the missing values in a column"*	[81,43]
	Population completeness: *"ratio of entities represented in an information system to the complete population"*	[81,43]
	"degree to which a given data collection includes data describing the corresponding set of real-world objects"	[44]
	extensional: $\frac{\lvert obj_{data}\rvert}{\lvert obj_{dom}\rvert}$	[66]
	intensional: $\frac{\lvert attr_{data}\rvert}{\lvert attr_{dom}\rvert}$	[66]
	of property *prop*: $\frac{\lvert obj_{\lvert prop}\rvert}{\lvert obj_{dom}\rvert}$	[66]
	"degree to which all required information is present in a particular dataset"	[67]
	Schema Completeness/Ontology Completeness: *"degree to which the classes and properties of an ontology are represented"*	[67]
	Property completeness: *"measure of the missing values for a specific property"*	[67]
	Population completeness: *"percentage of all real-world objects of a particular type that are represented in the datasets"*	[67]
	Interlinking completeness: *"degree to which instances in the dataset are interlinked"*	[67]
Comprehensiveness	—	[73]
Comprehensibility	*"ease with which human consumers can understand and utilize the data"*	[64]
Concise representation	—	[40,100] [45,76]
	here 'Representational conciseness': *"degree to which the structure of the data matches the data itself"*	[103]
	"extent to which information is compactly represented"	[46,81]
	here 'Representational conciseness': *"refers to the representation of the data which is compact and well formatted on the one hand and clear and complete on the other hand"*	[67]
Conciseness	—	[73]
	here 'Duplicate'	[34]
	here 'Minimality', Model: *"Number of redundant entities/relationships in a model"*	[101]
	here 'Minimality', Concept: *"Equivalence of the description with that of other concepts in the same model"*	[101]
	here 'Minimality', Schema: *"Number of redundant relations"*	[101]
	here 'Minimality', Type: *"Number of redundant attributes"*	[101]

Conciseness (cont.)	*here 'Redundancy': "same real-world entity or relationship is represented more than once"*	[50]
	here 'Uniqueness': "degree to which data is free of redundancies in breadth, depth, and scope"	[43]
	of property *prop*: $\frac{\lvert obj_{\lvert prop_{uniq}} \rvert}{\lvert obj_{\lvert prop} \rvert}$	[66]
	"refers to the minimization of redundancy of entities at the schema and the data level"	[67]
	distinction: *'intensional conciseness' (schema level, redundant classes and properties)* vs. *'extensional conciseness' (data level, redundant instances)*	[67]
Conformance	—	[105]
Consistent representation	—	[100,45] [76]
	here 'Representation consistency': "Coherence of physical instances of data with their formats"	[41]
	here 'Representational consistency'	[40]
	here 'Representational consistency': "degree to which the structure of the data conforms to previously returned data"	[103]
	"extent to which information is represented in the same format"	[46,81]
	here 'Representational inconsistency': "data quality problems that originate from an actual state σ' of an element E to differ from the required state σ for E"	[50]
Consistency	—	[51,73] [34]
	"Coherence of the same datum, represented in multiple copies, or different data to respect integrity constraints and rules"	[41]
	Agent: *"Is the delivered data consistent with other data"*	[101]
	Data Store: *"Number of tuples violating constraints, number of coding differences"*	[101]
	Discrete: *"Same value across all cases"*	[104]
	Continuous$_1$: *"Same value across multiple occurrences"*	[104]
	Continuous$_2$: *"Tightly dispersed values across multiple measures"*	[104]
	"Different data in a database are logically compatible."	[37]
	distinction: *'format level'* vs. *'instance level'*	[39]
	"Consistency implies that two or more values do not conflict with each other"	[81]
	"refers to the violation of semantic rules defined over a set of data items"	[44]
	"degree to which the statements of a source's data are conflict-free and no conflicting statements are inferable"	[64]
	of a property *prop*: $\frac{\lvert obj_{\lvert prop_{cons}} \rvert}{\lvert obj_{\lvert prop} \rvert}$	[66]
	"means that a knowledge base is free of (logical/formal) contradictions with respect to particular knowledge representation and inference mechanisms"	[67]
Convenience	—	[73]
Correctness	—	[73,76]

Correctness (cont.)	*here 'Correct'*: *"Garbling (map to a wrong state)"*	[32]
	Model: *"Number of conflicts to other models/real world"*	[101]
	Concept: *"Correctness of the description wrt. real world entity"*	[101]
	Schema: *"Correctness of mapping of the conceptual model to logical schema"*	[101]
	Type: *"Correctness of the mapping of the concept to a type"*	[101]
Cost	*"How the user measures the cost of retrieving the information."*	[104]
	own cost model	[76]
	"sum of the cost of data quality assessment and improvement activities, also referred to as the cost of the data quality program and the cost associated with poor data quality"	[44]
Credibility	Agent: *"Believability in the process that delivers the values"*	[101]
	Data Store: *"Number of tuples with default values"*	[101]
Currency	—	[73]
	"Degree to which a datum is up to date"	[41]
	"Recentness of collection"	[104]
	$time_{next\_update} - time_{last\_update} \longrightarrow$ *"age [...] from generation to status change"*	[76]
Customer support	*"amount and usefulness of human help via email or telephone"*	[103]
Data deficiency	—	[110]
Data interpretability	Agent: *"Number of tuples with interpretable data, documentation for key values, is the format understandable?"*	[101]
	Data Store: *"Number of tuples with interpretable data, documentation for key values, is the format understandable?"*	[101]
Design deficiencies	—	[110]
Documentation	*"amount and usefulness of documents with metadata"*	[103]
Durability	—	[105]
Ease of manipulation	*"The extent to which data can be processed easily (e.g., indexed and analyzed)."*	[37]
	"the extent to which data is easy to manipulate and apply to different tasks"	[46]
Ease of operation	—	[45]
Ease of querying	—	[101]
Efficiency	Software efficiency: *"Performance, response time, processing time"*	[101]
	Storage efficiency: *"It takes less space to store data."*	[37]
	Retrieval efficiency: *"It is fast to find desired information."*	[37]
Efficient use of memory	*"Efficiency in the physical representation. An icon is less efficient than a code"*	[41]
Faithfulness	*"The extent to which the presented data are identical to the origin in meaning and precision."*	[37]
Features	—	[105]
Formality	*"Data are presented concisely and consistently"*	[37]
Format flexibility	*"Changes in user needs and recording medium can be easily accommodated"*	[41]
Format precision	*"Ability to distinguish between elements in the domain that must be distinguished by users"*	[41]
Free-of-error	—	[45]

Free-of-error (cont.)	*"extent to which data is correct and reliable"*	[46]
Functionality	*"Number of functions not appropriate for specified tasks, number of modules unable to interact with specified systems"*	[101]
Heterogeneity	structural: *"same real-world domain is represented by different schema elements"*	[50]
	semantic: *"difference in the intension of the compared schemata with overlapping elements"*	[50]
Intelligibility	*"Capable of being understood, apprehended or comprehended"*	[104]
Interactivity	—	[73]
Interlinking	—	[65]
	"degree to which entities that represent the same concept are linked to each other, be it within or between two or more linked data sources"	[67]
Interoperability	*"degree to which the format and structure of the information conforms to previously returned information as well as data from other sources"*	[67]
Interpretability	—	[40,100] [45,39] [76]
	"Ability of the user to interpret correctly values from their format"	[41]
	Model: *"Quality of documentation"*	[101]
	Concept: *"Quality of documentation"*	[101]
	Schema: *"Quality of documentation"*, *"Is the schema understandable?"*	[101]
	Type: *"Quality of documentation"*, *"Is the type understandable?"*	[101]
	Agent: *"Is the data delivered understandable?"*	[101]
	Data Store: *"Is the data stored understandable?"*	[101]
	"degree to which the information conforms to the technical ability of the consumer"	[103]
	"Data have clear meaning."	[37]
	"extent to which information is in appropriate languages, symbols, and units, and the definitions are clear"	[46,81]
	"refers to technical aspects of the data, that is, whether information is represented using an appropriate notation and whether the machine is able to process the data"	[67]
Lack of confusion	—	[111]
Latency	*"amount of time in seconds from issuing the query until the first data item reaches the user"*	[103]
Licensing	*here 'License'*: *"degree to which the provided data can be used with own applications"*	[64]
	"granting of permission for a consumer to re-use a dataset under defined conditions"	[67]
Maintainability	—	[73]
	"Man-hours needed for maintaining and testing this software"	[101]
Meaningfulness	—	[76]
	"If intelligible, the information has some minimum level of meaning to the user. The meaning content may be increased by adding structure or organization."	[104]

Meaningfulness (cont.)	*here 'Meaningful': "Meaningless IS state and Garbling (map to a meaningless state)"*	[32]
Metadata evolution	Model: *"Is the evolution of the model documented?"*	[101]
	Concept: *"Is the evolution of the concept documented?"*	[101]
	Schema: *"Is the evolution of the schema documented?"*	[101]
	Type: *"Is the evolution of the type documented?"*	[101]
Navigation	*here 'Navigability': "One can navigate around the related information."*	[37]
	"extent to which data are easily found and linked to"	[77]
Non-fictitiousness	Records: *"No false or redundant records exist"*	[104]
	Attributes: *"No false or redundant attributes exist"*	[104]
	Values: *"No false values exist"*	[104]
Objectivity	—	[40,100] [45,76]
	here 'Freedom from bias'	[112]
	here 'Neutrality': "Data selected for presentation are not in favor of any particular opinion or purpose."	[37]
	"degree to which data is unbiased and impartial"	[103,46] [81]
	"The extent to which the sample selected for observation is representative of a population."	[37]
Offensiveness	*"Information consumers can consider web content offensive for moral, religious, or political reasons."*	[81]
Operation deficiencies	—	[110]
Perceived quality	—	[105]
Performance	*own performance model*	[105]
	"comprises aspects of enhancing the performance of a source as well as measurings of the actual values"	[64]
	"efficiency of a system that binds to a large dataset, that is, the more performant a data source is the more efficiently a system can process data"	[67]
Period of validity	$time_{next\_change} - time_{last\_update} \longrightarrow$ *"how long the item remains valid"*	[76]
Portability	*"Number of cases where the software failed to adopt to new environments; man-hours needed to install software in new environments"*	[101]
	"The format can be applied to as a wide set of situations as possible"	[41]
Precision	—	[76]
Price	*"amount of money a user has to pay for a query"*	[103]
Privacy	*"The extent to which a task has permissions to access the data."*	[37]
Punctuality	*"refers to the time lag between the release date of data and the target date when it should have been delivered"*	[102]
Quality of service	*"measure for transmission and error rates of Web sources"*	[103]
Readability	—	[76]
Reasonability	—	[113]
Relevancy	—	[40,100] [45,76]

Relevancy (cont.)	*"degree to which the provided information satisfies the users need"*	[103]
	"extent to which data are applicable and useful for a specific theory"	[37]
	"extent to which information is applicable and helpful for the task at hand"	[46,81]
	"degree to which statistics meet current and potential user needs"	[102]
	"refers to the provision of information which is in accordance with the task at hand and important to the users' query"	[67]
Reliability	—	[105]
	"Frequency of failures, Fault tolerance"	[101]
	here 'Reliability of Data Clerks': *"The extent to which data entry clerks are able to avoid mistakes."*	[37]
Reputation	—	[40,100] [45,76]
	"degree to which the data or its source is in high standing"	[103]
	"extent to which data is highly regarded in terms of its source or content"	[46]
Response time	*"delay in seconds between submission of a query by the user and reception of the complete response from the data source"*	[103,81]
Security	—	[73,45]
	here 'Access security'	[40,100]
	Schema: *"Level of security (access rights)"*	[101]
	Type: *"Level of security (access rights)"*	[101]
	Agent: *"Are there physical access restrictions?"*	[101]
	Data Store: *"Is the store able to prevent unauthorized access?"*	[101]
	"degree to which data is passed privately from users to the data source and back"	[103]
	"The extent to which a task has secured access to the data."	[37]
	"extent to which access to data is restricted appropriately to maintain its security"	[46]
	"extent to which data is protected against alteration and misuse"	[67]
Semantic stability	*"The same data have same meaning across time and space."*	[37]
Serviceability	—	[105]
Soundness	$\frac{\lvert D_{real} \cap D_{ideal} \rvert}{\lvert D_{real} \rvert}$	[38]
Speed	—	[73]
Syntactic validity	*"degree to which an RDF document conforms to the specification of the serialization format"*	[67]
Time	*"How long it takes to retrieve the information"*	[104]
Time-inaccuracy	*here 'Time-inaccurate'*	[34]
Timeliness	—	[40,100] [101,73] [45,39]
	"average age of the data in a source"	[103]
	"extent to which data are sufficiently up-to-date for a task."	[37,46] [67]
	$max\left(\left(1 - \frac{currency}{period\_of\_validity}\right), 0\right)^{\beta}$	[76]

Timeliness	*"length of time between [an information's] availability and the event or phenomenon it describes"*	[102]
	"degree to which information is up-to-date"	[81]
	conversion scenario: data outdated if modification time of the source more current than modification time of target or if expiry date passed	[43]
	"refers to the currentness of the data provided by a source"	[64]
Traceability	—	[73]
	Model: *"Are the designer's requirements and changes recorded?"*	[101]
	Concept: *"Are the designer's requirements and changes recorded?"*	[101]
	Schema: *"Are the designer's requirements and changes recorded?"*	[101]
	Type: *"Are the designer's requirements and changes recorded?"*	[101]
Trustworthiness	*here 'Trustworthiness of the collector': "The extent to which the collector has integrity of not committing falsification."*	[37]
	"degree to which the information is accepted to be correct, true, real and credible"	[67]
Unambiguity	*here 'Ambiguous'*	[34]
	here 'Unambiguous': "Improper representation: multiple RW states mapped to the same IS state"	[32]
	here 'Ambiguity': "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."	[50]
Understandability	—	[45]
	here 'Ease of understanding'	[40,100]
	here 'Case of understanding' (sic)	[76]
	"degree to which the data can be easily comprehended by the user"	[103,46] [81]
	"refers to the ease with which data can be comprehended without ambiguity and be used by a human information consumer"	[67]
Uniformity	*"refers to the usage of established techniques in order to increase the usability of the data"*	[64]
Usability	*"Acceptance of the users"*	[101]
	"extent to which information is clear and easily used"	[77]
Usefulness	Schema: *"Is the schema used by any users?"*	[101]
	Type: *"Is the type used by any users?"*	[101]
	Agent: *"Is the data delivered by the agent really used in the destination store?"*	[101]
	Data Store: *"Is the data in this store queried by a user?"*	[101]
Vacuity	*"We consider instances or schema elements that have no meaning at all in the presented context as vacuous."*	[50]
Validity	*"consists of two aspects influencing the usability of the documents: the valid usage of the underlying vocabularies and the valid syntax of the documents"*	[64]
Value-added	—	[40,100] [76]
	"amount of monetary benefit the use of the data provides"	[103]

Value-added (cont.)	*"extent to which data is beneficial and provides advantages from its use"*	[46]
Verifiability	*"degree and ease with which the data can be checked for correctness"*	[103,81]
	"refers to the means a consumer is provided with, which can be used to examine the data for correctness"	[64]
Versatility	*"refers to alternative representations of the data and its handling"*	[64]
	"refers to the availability of the data in an internationalized way and alternative representations of data"	[67]
Volatility	*"How long it remains valid"*	[104]
	$\left(time_{exp} - time_{curr}\right) - \int_{time_{curr}}^{time_{exp}} F_{exp}(t)\,dt$	[106]

List of Figures

List of Tables

Listings

References

1. Berners-Lee, Tim: *Information Management: A Proposal*. Available at http://www.w3.org/History/1989/proposal.html, March 1989.
2. He, Bin, Mitesh Patel, Zhen Zhang and Kevin Chen-Chuan Chang: *Accessing the Deep Web: A Survey*. Communications of the ACM, 50(5):94–101, May 2007.
3. Bertails, Alexandre and Eric Gordon Prud'hommeaux: *Interpreting Relational Databases in the RDF Domain*. In Musen, Mark A. and Oscar Corcho (editors): *Proceedings of the 6th International Conference on Knowledge Capture*, K-CAP '11, pages 129–136, New York, NY, USA, 2011. ACM Press.
4. Motik, Boris, Ian Horrocks and Ulrike Sattler: *Bridging the Gap Between OWL and Relational Databases*. In Williamson, Carey, Mary Ellen Zurko, Peter Patel-Schneider and Prashant Shenoy (editors): *Proceedings of the 16th International Conference on World Wide Web*, pages 807–816, New York, NY, USA, May 2007. ACM Press.
5. Berners-Lee, Tim: *Relational Databases on the Semantic Web*. Available at http://www.w3.org/DesignIssues/RDB-RDF.html, 1998.
6. Manola, Frank and Eric Miller (editors): *RDF Primer*. W3C Recommendation. World Wide Web Consortium, February 2004. Available at http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.
7. Das, Souripriya, Seema Sundara and Richard Cyganiak (editors): *R2RML: RDB to RDF Mapping Language*. World Wide Web Consortium, September 2012. http://www.w3.org/TR/2012/REC-r2rml-20120927/.
8. Auer, Sören, Jens Lehmann, Axel-Cyrille Ngonga Ngomo and Amrapali Zaveri: *Introduction to Linked Data and Its Lifecycle on the Web*. In Rudolph, Sebastian, Georg Gottlob, Ian Horrocks and Frank Harmelen (editors): *Reasoning Web. Semantic Technologies for Intelligent Data Access*, volume 8067 of *Lecture Notes in Computer Science*, pages 1–90. Springer-Verlag, Berlin Heidelberg, 2013.
9. Zaveri, Amrapali, Dimitris Kontokostas, Mohamed A. Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer and Jens Lehmann: *User-driven Quality Evaluation of DBpedia*. In *I-SEMANTICS '13: Proceedings of the 9th International Conference on Semantic Systems*, ACM International Conference Proceeding Series, pages 97–104, New York, NY, USA, September 2013. ACM Press.
10. Juran, Joseph Moses and Albert Blanton Godfrey: *Juran's Quality Handbook*. McGraw-Hill, New York City, United States, 5th edition, 1998.
11. Arenas, Marcelo, Alexandre Bertails, Eric Prud'hommeaux and Juan Sequeda (editors): *A Direct Mapping of Relational Data to RDF*. World Wide Web Consortium, September 2012. Available at http://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/.
12. Spanos, Dimitrios-Emmanuel, Periklis Stavrou and Nikolas Mitrou: *Bringing relational databases into the Semantic Web: A survey*. Semantic Web Journal, 3(2):169–209, 2012.
13. Zhao, Shuxin and Elizabeth Chang: *From Database to Semantic Web Ontology: An Overview*. In Meersman, Robert, Zahir Tari and Pilar Herrero (editors): *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops – OTM Confederated International Workshops and Posters – AWeSOMe, CAMS, OTM Academy Doctoral Consortium, MONET, OnToContent, ORM, PerSys, PPN, RDDS, SSWS, and SWWS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part II*, volume 4806 of *Lecture Notes in Computer Science*, pages 1205–1214, Berlin Heidelberg, November 2007. Springer-Verlag.
14. Sahoo, Satya S., Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda and Ahmed Ezzat: *A Survey of Current Approaches for Mapping of Relational Databases to RDF*. Technical Report, W3C RDB2RDF Incubator

Group, January 2009. Available at http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf.

15. Stadler, Claus, Jörg Unbehauen, Jens Lehmann and Sören Auer: *Connecting Crowdsourced Spatial Information to the Data Web with Sparqlify*. Technical Report, University of Leipzig, Leipzig, 2013. Available at http://sparqlify.org/downloads/documents/2013-Sparqlify-Technical-Report.pdf.
16. RDB2RDF Working Group: *Implementations*. Available at http://www.w3.org/2001/sw/rdb2rdf/wiki/Implementations, July 2012. Accessed October 20.
17. Openlink Software: *Mapping Relational Data to RDF with Virtuoso's RDF Views*. Available at http://virtuoso.openlinksw.com/whitepapers/relational%20rdf%20views%20mapping.html. Accessed September 26, 2013.
18. Unbehauen, Jörg, Claus Stadler and Sören Auer: *Accessing Relational Data on the Web with SparqlMap*. In Takeda, Hideaki, Yuzhong Qu, Riichiro Mizoguchi and Yoshinobu Kitamura (editors): *Semantic Technology: Second Joint International Conference, JIST 2012, Nara, Japan, December 2-4, 2012. Proceedings*, volume 7774 of *Lecture Notes in Computer Science*, pages 65–80, Berlin Heidelberg, December 2012. Springer-Verlag.
19. International Organization for Standardization: *ISO 9000 - Quality management*. Available at http://www.iso.org/iso/home/standards/management-standards/iso_9000.htm. Accessed October 23.
20. European Association for Quality Assurance in Higher Education: *Standards and Guidelines for Quality Assurance in the European Higher Education Area*. European Association for Quality Assurance in Higher Education, Helsinki, Finland, 2005.
21. Oelkers, Jürgen and Kurt Reusser: *Qualität entwickeln – Standards sichern – mit Differenz umgehen*. Bundesministerium für Bildung und Forschung, Berlin, Germany, 2008.
22. Bundesministerium für Familie, Senioren, Frauen und Jugend: *Qualitätsstandards für Beteiligung von Kindern und Jugendlichen*. Bundesministerium für Familie, Senioren, Frauen und Jugend, Berlin, Germany, 2nd edition, February 2012.
23. International Monetary Fund: *Data Quality Assessment Framework*. http://dsbb.imf.org/pages/dqrs/DQAF.aspx, 2012. Accessed August 5, 2013.
24. Rowley, Jennifer: *The Wisdom Hierarchy: Representations of the DIKW Hierarchy*. Journal of Information Science, 33(2):163–180, 2007.
25. Batini, Carlo and Monica Scannapieco: *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer-Verlag, Berlin Heidelberg, 2010.
26. Dasu, Tamraparni and Theodore Johnson: *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons, Hoboken, NJ, USA, 2003.
27. Kahn, Beverly K., Diane M. Strong and Richard Y. Wang: *Information Quality Benchmarks: Product and Service Performance*. Communications of the ACM, 45(4):184–192, April 2002.
28. Reeves, Carol A. and David A. Bednar: *Defining Quality: Alternatives and Implications*. Academy of Management Review, 19(3):419–445, 1994.
29. Parssian, Amir, Sumit Sarkar and Varghese S. Jacob: *Assessing Data Quality for Information Products*. In De, Prabuddha and Janice I. DeGross (editors): *Proceedings of the Twentieth International Conference on Information Systems*. Association for Information Systems, December 1999.
30. Parssian, Amir, Sumit Sarkar and Varghese S. Jacob: *Assessing Data Quality for Information Products: Impact of Selection, Projection, and Cartesian Product*. Management Science, 50(7):967–982, July 2004.
31. Shannon, Claude Elwood and Warren Weaver: *A Mathematical Theory of Communication*. University of Illinois Press, Urbana, Illinois, 1949.
32. Wand, Yair and Richard Y. Wang: *Anchoring Data Quality Dimensions in Ontological Foundations*. Communications of the ACM, 39(11):86–95, November 1996.

33. Lei, Yuangui, Andriy Nikolov, Victoria Uren and Enrico Motta: *Detecting Quality Problems in Semantic Metadata without the Presence of a Gold Standard*. In García-Castro, Raúl, Denny Vrandecic, Asunción Gómez-Pérez, York Sure and Zhisheng Huang (editors): *Proceedings of the 5th International Workshop on Evaluation of Ontologies and Ontology-based Tools (EON2007)*, volume 329 of *CEUR Workshop Proceedings*, pages 51–60, Aachen, Germany, November 2007. Redaktion Sun SITE, Informatik V, RWTH Aachen.
34. Lei, Yuangui, Victoria Uren and Enrico Motta: *A Framework for Evaluating Semantic Metadata*. In Sleeman, Derek and Ken Barker (editors): *Proceedings of the 4th International Conference on Knowledge Capture*, pages 135–142, New York, NY, USA, 2007. ACM Press.
35. Orr, Ken: *Data Quality and Systems Theory*. Communications of the ACM, 41(2):66–71, February 1998.
36. Shankaranarayanan, Ganesan, Richard Y. Wang and Mostapha Ziad: *IP-MAP: Representing the Manufacture of an Information Product*. In Klein, Barbara D. and Donald F. Rossin (editors): *Fifth Conference on Information Quality (IQ 2000)*, pages 1–16, Cambridge, MA, USA, 2000. MIT Press.
37. Liu, Liping and Lauren N. Chi: *Evolutional Data Quality: A Theory-specific View*. In Fisher, Craig and Bruce Davidson [114], pages 292–304.
38. Motro, Amihai and Igor Rakov: *Estimating the Quality of Databases*. In Andreasen, Troels, Henning Christiansen and Henrik Legind Larsen (editors): *Flexible Query Answering Systems: Third International Conference, FQAS'98 Roskilde, Denmark, May 13–15, 1998 Proceedings*, volume 1495 of *Lecture Notes in Computer Science*, pages 298–307, Berlin Heidelberg, 1998. Springer-Verlag.
39. Scannapieco, Monica and Tiziana Catarci: *Data Quality under the Computer Science Perspective*. Archivi & Computer, 2:1–15, 2002.
40. Wang, Richard Y. and Diane M. Strong: *Beyond Accuracy: What Data Quality Means to Data Consumers*. Journal of Management Information Systems, 12(4):5–33, March 1996.
41. Redman, Thomas C.: *Data Quality for the Information Age*. Artech House, 1996.
42. Bizer, Christian and Richard Cyganiak: *Quality-driven Information Filtering Using the WIQA Policy Framework*. Web Semantics: Science, Services and Agents on the World Wide Web, 7(1):1–10, January 2009.
43. Fürber, Christian and Martin Hepp: *Swiqa - A Semantic Web Information Quality Assessment Framework*. In Tuunainen, Virpi Kristiina, Matti Rossi and Joe Nandhakumar (editors): *19th European Conference on Information Systems, ECIS 2011, Helsinki, Finland, June 9-11, 2011*, 2011.
44. Batini, Carlo, Cinzia Cappiello, Chiara Francalanci and Andrea Maurino: *Methodologies for Data Quality Assessment and Improvement*. ACM Computing Surveys, 41(3):16:1–16:52, July 2009.
45. Lee, Yang W., Diane Diane M. Strong, Beverly K. Kahn and Richard Y. Wang: *AIMQ: A Methodology for Information Quality Assessment*. Information & Management, 40(2):133–146, December 2002.
46. Pipino, Leo L., Yang W. Lee and Richard Y. Wang: *Data Quality Assessment*. Communications of the ACM, 45(4):211–218, April 2002.
47. Kontokostas, Dimitris, Amrapali Zaveri, Sören Auer and Jens Lehmann: *TripleCheckMate: A Tool for Crowdsourcing the Quality Assessment of Linked Data*. In Klinov, Pavel and Dmitry Mouromtsev (editors): *Knowledge Engineering and the Semantic Web: 4th International Conference, KESW 2013, St. Petersburg, Russia, October 7-9, 2013. Proceedings*, volume 394 of *Communications in Computer and Information Science*, pages 265–272, Berlin Heidelberg, October 2013. Springer-Verlag.

48. Knuth, Magnus, Johannes Hercher and Harald Sack: *Collaboratively Patching Linked Data*. The Computing Research Repository, abs/1204.2715, April 2012.
49. Kontokostas, Dimitris, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann and Roland Cornelissen: *Test-driven Evaluation of Linked Data Quality*. In Chung, Chin-Wan, Andrei Z. Broder, Kyuseok Shim and Torsten Suel (editors): *WWW '14 Proceedings of the 23rd International Conference on World Wide Web*, pages 747–758, New York, NY, USA, 2014. ACM Press.
50. Fürber, Christian and Martin Hepp: *Using SPARQL and SPIN for Data Quality Management on the Semantic Web*. In Abramowicz, Witold and Robert Tolksdorf (editors): *Business Information Systems: 13th International Conference, BIS 2010, Berlin, Germany, May 3-5, 2010. Proceedings*, volume 47 of *Lecture Notes in Business Information Processing*, pages 35–46, Berlin Heidelberg, May 2010. Springer-Verlag.
51. English, Larry P.: *Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits*. John Wiley & Sons, New York, USA, 1999.
52. Redman, Thomas: *Data Quality – The Field Guide*. Digital Press, 2001.
53. Wang, Richard Y., Mostapha Ziad and Yang W. Lee: *Data Quality*. Kluwer Academic Publishers, Dordrecht, Netherlands, 2001.
54. Artz, Donovan and Yolanda Gil: *A Survey of Trust in Computer Science and the Semantic Web*. Web Semantics: Science, Services and Agents on the World Wide Web, 5(2):58–71, June 2007.
55. Hartig, Olaf: *Trustworthiness of Data on the Web*. In *Proceedings of the STI Berlin & CSW PhD Workshop*, 2008.
56. Lausen, Georg, Michael Meier and Michael Schmidt: *SPARQLing Constraints for RDF*. In Mouaddib, Noureddine, Patrick Valduriez, Alfons Kemper, Mokrane Bouzeghoub, Volker Markl, Laurent Amsaleg and Ioana Manolescu (editors): *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '08, pages 499–509, New York, NY, USA, March 2008. ACM Press.
57. Hogan, Aidan and Richard Cyganiak: *Frequently Observed Problems on the Web of Data*. `http://pedantic-web.org/fops.html`, November 2009. Accessed August 1, 2013.
58. Böhm, Christoph, Felix Naumann, Ziawasch Abedjan, Dandy Fenz, Toni Grütze, Daniel Hefenbrock, Matthias Pohl and David Sonnabend: *Profiling linked open data with ProLOD*. In *2010 IEEE 26th International Conference on Data Engineering Workshops*, pages 175–178, Washington, DC, USA, March 2010. Institute of Electrical and Electronics Engineers.
59. Guéret, Christophe, Paul Groth, Frank van Harmelen and Stefan Schlobach: *Finding the Achilles Heel of the Web of Data: Using Network Analysis for Link-Recommendation*. In Patel-Schneider, Peter F. et al. [115], pages 289–304.
60. Halpin, Harry, Patrick J. Hayes, James P. McCusker, Deborah L. McGuinness and Henry S. Thompson: *When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data*. In Patel-Schneider, Peter F. et al. [115], pages 305–320.
61. Hogan, Aidan, Andreas Harth, Alexandre Passant, Stefan Decker and Axel Polleres: *Weaving the Pedantic Web*. In Bizer, Christian, Tom Heath, Tim Berners-Lee and Michael Hausenblas (editors): *Proceedings of the WWW2010 Workshop on Linked Data on the Web*, volume 628 of *CEUR Workshop Proceedings*, Aachen, Germany, April 2010. Redaktion Sun SITE, Informatik V, RWTH Aachen.
62. Hogan, Aidan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres and Stefan Decker: *An Empirical Survey of Linked Data Conformance*. Web Semantics: Science, Services and Agents on the World Wide Web, 14:14–44, July 2012.
63. Demter, Jan, Sören Auer, Michael Martin and Jens Lehmann: *LODStats – An Extensible Framework for High-performance Dataset Analytics*. In ten Teije, Annette, Johanna Völker, Siegfried Handschuh, Heiner Stuckenschmidt, Mathieu d'Acquin, Andriy

Nikolov, Nathalie Aussenac-Gilles and Nathalie Hernandez (editors): *Knowledge Engineering and Knowledge Management: 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings*, volume 7603 of *Lecture Notes in Computer Science*, pages 353–362, Berlin Heidelberg, October 2012. Springer-Verlag.

64. Flemming, Annika: *Qualitätsmerkmale von Linked Data-veröffentlichenden Datenquellen*. Diploma Thesis, Humbold-Universität zu Berlin, March 2011.
65. Guéret, Christophe, Paul Groth, Claus Stadler and Jens Lehmann: *Assessing linked data mappings using network measures*. In Simperl, Elena, Philipp Cimiano, Axel Polleres, Oscar Corcho and Valentina Presutti (editors): *The Semantic Web: Research and Applications*, volume 7295 of *Lecture Notes in Computer Science*, pages 87–102. Springer-Verlag, Berlin Heidelberg, 2012.
66. Mendes, Pablo N., Hannes Mühleisen and Christian Bizer: *Sieve: Linked Data Quality Assessment and Fusion*. In Srivastava, Divesh and Ismail Ari (editors): *EDBT-ICDT '12: Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 116–123, New York, NY, USA, March 2012. ACM.
67. Zaveri, Amrapali, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann and Sören Auer: *Quality Assessment Methodologies for Linked Open Data*. To appear in the Semantic Web Journal, 2014.
68. Lausen, Georg: *Relational Databases in RDF: Keys and Foreign Keys*. In Christophides, Vassilis, Martine Collard and Claudio Gutierrez (editors): *Semantic Web, Ontologies and Databases: VLDB Workshop, SWDB-ODBIS 2007, Vienna, Austria, September 24, 2007, Revised Selected Papers*, volume 5005 of *Lecture Notes in Computer Science*, pages 43–56, Berlin Heidelberg, September 2007. Springer-Verlag.
69. Levshin, Dmitry V.: *Mapping Relational Databases to the Semantic Web with Original Meaning*. In Karagiannis, Dimitris and Zhi Jin (editors): *Knowledge Science, Engineering and Management: Third International Conference, KSEM 2009, Vienna, Austria, November 25-27, 2009. Proceedings*, volume 5914 of *Lecture Notes in Computer Science*, pages 5–16, Berlin Heidelberg, November 2009. Springer-Verlag.
70. Stadler, Claus, Jens Lehmann, Konrad Höffner and Sören Auer: *LinkedGeoData: A Core for a Web of Spatial Open Data*. Semantic Web Journal, 3(4):333–354, 2012.
71. Westphal, Patrick, Claus Stadler and Jonathan Pool: *Countering language attrition with PanLex and the Web of Data*. To appear in the Semantic Web Journal, 2014.
72. Harris, Steve and Andy Seaborne (editors): *SPARQL 1.1 Query Language*. World Wide Web Consortium, March 2013. Available at `http://www.w3.org/TR/2013/REC-sparql11-query-20130321/`.
73. Eppler, Martin J. and Peter Muenzenmayer: *Measuring Information Quality in the Web Context: A Survey of State-of-the-Art Instruments and an Application Methodology*. In Fisher, Craig and Bruce Davidson [114], pages 187–196.
74. Zeng, Liangzhao, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam and Quan Z. Sheng: *Quality Driven Web Services Composition*. In Hencsey, Gusztáv, Bebo White, Yih-Farn Robin Chen, László Kovács and Steve Lawrence (editors): *Proceedings of the 12th International Conference on World Wide Web*, pages 411–421, New York, NY, USA, May 2003. ACM Press.
75. Mostafavi, Mir Abolfazl, Geoffrey Edwards and Robert Jeansoulin: *An Ontology-based Method for Quality Assessment of Spatial Data Bases*. In Frank, Andrew U. and Eva Grum (editors): *Proceedings of the 3rd International Symposium on Spatial Data Quality*, volume 1/28a of *Geoinfo series*, pages 49–66, Vienna, April 2004. Department for Geoinformation and Cartography, Vienna University of Technology.
76. Su, Ying and Zhanming Jin: *A Methodology for Information Quality Assessment in the Designing and Manufacturing Processes of Mechanical Products*. In Chengalur-Smith, InduShobha N. et al. [116], pages 447–465.

77. Knight, Shirlee ann and Janice Burn: *Developing a Framework for Assessing Information Quality on the World Wide Web*. Informing Science Journal, 8:160–172, 2005.
78. Fürber, Christian and Martin Hepp: *Using Semantic Web Resources for Data Quality Management*. In Cimiano, Philipp and H. Sofia Pinto (editors): *Knowledge Engineering and Management by the Masses: 17th International Conference, EKAW 2010, Lisbon, Portugal, October 11-15, 2010. Proceedings*, volume 6317 of *Lecture Notes in Computer Science*, pages 211–225, Berlin Heidelberg, October 2010. Springer-Verlag.
79. Ullman, Jeffrey D.: *Information Integration Using Logical Views*. In Afrati, Foto and Phokion Kolaitis (editors): *Database Theory — ICDT '97: 6th International Conference Delphi, Greece, January 8–10, 1997 Proceedings*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40, Berlin Heidelberg, January 1997. Springer-Verlag.
80. Berners-Lee, Tim: *Linked Data*. Available at `http://www.w3.org/DesignIssues/LinkedData.html`, June 2009. Accessed October 17, 2013.
81. Bizer, Christian: *Quality-Driven Information Filtering in the Context of Web-Based Information Systems*. PhD thesis, Freie Universität Berlin, March 2007.
82. Brickley, Dan and Ramanathan V. Guha (editors): *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. World Wide Web Consortium, February 2004. Available at `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`.
83. Baader, Franz, Diego Calvanese, Deborah L. McGuinnes, Daniele Nardi and Peter F. Partel-Schneider (editors): *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge, Second edition, 2010.
84. Biron, Paul V. and Ashok Malhotra (editors): *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation. World Wide Web Consortium, October 2004. Available at `http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/`.
85. Bechhofer, Sean, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider and Lynn Andrea Stein: *OWL Web Ontology Language Reference*. W3C Recommendation. World Wide Web Consortium, February 2004. Available at `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`.
86. Heath, Tom and Christian Bizer: *Linked Data: Evolving the Web into a Global Data Space*. Morgan and Claypool, 1st edition, 2011.
87. Hayes, Patrick John and Peter F. Patel-Schneider (editors): *RDF 1.1 Semantics*. W3C Recommendation. World Wide Web Consortium, February 2014. Available at `http://www.w3.org/TR/rdf11-mt/`.
88. Phillips, Addison and Mark Davis (editors): *Tags for Identifying Languages*. Number 5646 in *Request for Comments*. Internet Engineering Task Force, September 2009. Available at `http://tools.ietf.org/rfc/bcp/bcp47.txt`.
89. Bailey, Todd M. and Ulrike Hahn: *Determinants of Wordlikeness: Phonotactics or Lexical Neighborhoods?* Journal of Memory and Language, 44(4):568–591, May 2001.
90. Ljolje, Andrej and Stephen E. Levinson: *Development of an Acoustic-phonetic Hidden Markov Model for Continuous Speech Recognition*. IEEE Transactions on Signal Processing, 39(1):29–39, January 1991.
91. Albro, Daniel M.: *Some Learning Algorithms for Phonotactics*. `http://www.linguistics.ucla.edu/people/grads/albro/ucla-learn-talk1.pdf`, June 2000. Accessed December 13, 2013.
92. Lentz, Tomas Ostar: *Phonotactic Illegality and Probability in Speech Perception: Evidence from second language listeners*. PhD thesis, Utrecht Institute of Linguistics OTS, 2011.
93. Vitevitch, Michael S., Paul A. Luce, David B. Pisoni and Edward T. Auer: *Phonotactics, Neighborhood Activation, and Lexical Access for Spoken Words*. Brain and Language, 68(1-2):306–311, June 1999.

94. Berners-Lee, Tim, Roy Fielding and Larry Masinter (editors): *Uniform Resource Identifiers (URI): Generic Syntax*. Number 2396 in *Request for Comments*. Internet Engineering Task Force, August 1998. Available at `http://www.ietf.org/rfc/rfc2396.txt`.
95. Berners-Lee, Tim, Roy Fielding and Larry Masinter (editors): *Uniform Resource Identifier (URI): Generic Syntax*. Number 3986 in *Request for Comments*. Internet Engineering Task Force, January 2005. Available at `http://www.ietf.org/rfc/rfc3986.txt`.
96. Braden, Robert (editor): *Requirements for Internet Hosts – Application and Support*. Number 1123 in *Request for Comments*. Internet Engineering Task Force, October 1989. Available at `http://tools.ietf.org/rfc/rfc1123.txt`.
97. Carpenter, Brian, Stuard Cheshire and Robert Hinden (editors): *Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers*. Number 6874 in *Request for Comments*. Internet Engineering Task Force, February 2013. Available at `http://www.ietf.org/rfc/rfc6874.txt`.
98. Klensin, John (editor): *Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework*. Number 5890 in *Request for Comments*. Internet Engineering Task Force, August 2010. Available at `http://tools.ietf.org/rfc/rfc5890.txt`.
99. Alexander, Keith, Richard Cyganiak, Michael Hausenblas and Jun Zhao: *Describing Linked Datasets with the VoID Vocabulary*. World Wide Web Consortium, March 2011. W3C Interest Group Note, available at `http://www.w3.org/TR/2011/NOTE-void-20110303/`.
100. Strong, Diane M., Yang W. Lee and Richard Y. Wang: *Data Quality in Context*. Communications of the ACM, 40(5):103–110, May 1997.
101. Jarke, Matthias, Manfred A. Jeusfeld, Christoph Quix and Panos Vassiliadis: *Architecture and Quality in Data Warehouses: An Extended Repository Approach*. Information Systems, 24(3):229–253, May 1999.
102. Bergdahl, Mats, Manfred Ehling, Eva Elvers, Erika Földesi, Thomas Körner, Andrea Kron, Peter Lohauss, Kornelia Mag, Vera Morais, Anja Nimmergut, Hans Viggo Sæbø, Ulrike Timm and Maria João Zilhão: *Handbook on Data Quality Assessment Methods and Tools*. European Commission, Eurostat, Wiesbaden, 2007.
103. Naumann, Felix: *Quality-Driven Query Answering for Integrated Information Systems*, volume 2261 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg, 2002.
104. Bovee, Matthew, Rajendra P. Srivastava and Brenda Mak: *A Conceptual Framework and Belief Function Approach to Assessing Overall Information Quality*. In Pierce, Elizabeth M. and Raïssa Katz-Haas (editors): *Sixth Conference on Information Quality (IQ 2001)*, pages 311–328, Cambridge, MA, USA, November 2001. MIT Press.
105. Garvin, David A.: *Managing Quality: The Strategic and Competitive Edge*. Free Press, New York, USA, 1988.
106. Pernici, Barbara and Monica Scannapieco: *Data Quality in Web Information Systems*. Journal on Data Semantics, 1:48–68, 2003.
107. Ballou, Donald P. and Harold L. Pazer: *Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems*. Management Science, 31(2):150–162, February 1985.
108. Naumann, Felix, Johann-Christoph Freytag and Ulf Leser: *Completeness of Integrated Information Sources*. Information Systems, 29(7):583–615, October 2004.
109. Scannapieco, Monica and Carlo Batini: *Completeness in the Relational Model: a Comprehensive Framework*. In Chengalur-Smith, InduShobha N. et al. [116], pages 333–345.
110. Huang, Kuan-Tsea, Yang W. Lee and Richard Y. Wang: *Quality Information and Knowledge*. Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, USA, 1999.

111. Goodhue, Dale L.: *Understanding User Evaluations of Information Systems*. Management Science, 41(12):1827–1844, December 1995.
112. DeLone, William H. and Ephraim R. McLean: *Information Systems Success: The Quest for the Dependent Variable*. Information Systems Research, 3(2):60–95, 1992.
113. Zmud, Robert: *Concepts, Theories and Techniques: an Empirical Investigation of the Dimensionality of the Concept of Information*. Decision Sciences, 9(2):187–195, 1978.
114. Fisher, Craig and Bruce Davidson (editors): *7th International Conference on Information Quality*, Cambridge, MA, USA, November 2002. MIT Press.
115. Patel-Schneider, Peter F., Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks and Birte Glimm (editors): *9th International Semantic Web Conference*, volume 6496 of *Lecture Notes in Computer Science*, Berlin Heidelberg, 2010. Springer-Verlag.
116. Chengalur-Smith, InduShobha N., Louiqa Raschid, Jennifer Long and Craig Seko (editors): *9th International Conference on Information Quality*, Cambridge, MA, USA, November 2004. MIT Press.