Linked Edit Rules: A Web Friendly Way of Checking Quality of RDF Data Cubes

Albert Meroño-Peñuela[1,2], Christophe Guéret[2], and Stefan Schlobach[1]

[1] Department of Computer Science, VU University Amsterdam, NL
albert.merono@vu.nl
[2] Data Archiving and Networked Services, KNAW, NL

Abstract. Statistical data often come with inconsistencies: records of people with negative ages, pregnant males, and underaged car drivers often populate statistical databases. National Statistical Offices (NSO) encode knowledge to detect these inconsistencies in so-called *edit rules*. These days, there is an increasing number of statistical data being published and linked on the Web using the RDF Data Cube vocabulary. However, edit rules are hardly ever published together with data cubes on the Web. This causes two important problems: (a) quality of RDF Data Cube data cannot be assessed; and (b) reusability of edit rules is hampered. In this paper we present *Linked Edit Rules (LER)*, a method that makes edit rules Web friendly and reusable as Linked Data. We show that LER can be easily linked, retrieved, reused, combined and executed to check quality and consistency of RDF Data Cubes, opening up the internal NSO validation processes to the Web.

Keywords: RDF Data Cube, Edit rules, Statistical consistency

1 Introduction

More and more statistical datasets are being published on the Web using the RDF Data Cube vocabulary (QB) [12,5], the W3C recommendation for publishing and linking multidimensional data, such as statistics, on the Web. As many Web data, these data cubes often come with inconsistencies that data consumers need to identify and fix by themselves before running their workflows. In statistics, data mining and data management this process is called *data cleansing*.

Data cleansing is an arduous and expensive task, mainly due to the heterogeneous nature of these errors and inconsistencies. National Statistical Offices (NSOs) set up procedures to *validate* data cubes before releasing them into the public domain. One of such validations consists of automatically identifying so-called *obvious inconsistencies*. An obvious inconsistency occurs when the cube contains a value or combination of values that cannot correspond to a real-world situation. For example, a person's age cannot be negative, a man cannot be pregnant and an underage person cannot possess a driving license. In order to validate data cubes against obvious inconsistencies, statisticians express this knowledge as *rules*, known in the data editing literature as **edit rules** or **edits**.

Edit rules are used to automatically detect inconsistent data points in statistical datasets, and can be divided into *micro-edits* and *macro-edits*. Micro-edits check obvious consistency of a single data record (or individual, row, observation). E.g., in a record of a demography dataset, the field *age group* cannot be *children* if the field *age* is 31. Macro-edits check obvious consistency of an entire field (or variable, column, dimension). E.g., population heights should be normally distributed; therefore, the field *height* is expected to follow a normal distribution. Current implementations only support the specification of micro-edits.

Edit rules currently exist only within NSOs' closed validation systems, hard-coded into source code, or serialized in a variety of models, syntaxes and formats. In addition, edit rules are rarely published on the Web together with the datasets to which they apply. If published at all, edit rules are only available as 1-star Linked Data, meaning that they are merely available on the Web, but with poor structure, in non-standard formats, in a non-uniquely identifiable way, and not linked with any other Web resource. These non-Web friendly practices hamper the use of these rules to validate statistical data. Consequently, edit rules cannot be easily located, retrieved, shared, reused nor combined on the Web to validate RDF Data Cubes. Statisticians need to constantly reimplement them offline.

The contribution of this paper is to overcome these pitfalls by applying the principles of Linked Data to edit rules. Concretely:

- We survey existing work on Semantic Web languages for constraint checking
- We provide a framework, a data model and an implementation to express micro- and macro-edits on the Web as *Linked Edit Rules (LER)*. The resulting LER are linked to RDF Data Cubes via QB dimensions, and check their per- and inter-record consistency.
- We build an automatic consistency checker using LER, *QBsistent*[3], which accurately finds all expected inconsistencies of various RDF Data Cubes on the Web and generates accurate provenance reports using PROV.

The rest of the paper is organised as follows. In Section 2 we define our problem and list our requirements, linking them to existing work in Section 3. In Section 4 we describe our Linked Edit Rules approach, implementing it in Section 5. In Section 6 we evaluate Linked Edit Rules, and we conclude in Section 7.

2 Background and Problem Definition

Micro-edits are constraints that must be met at the record level, i.e., by each individual row independently of the others. For instance, in statistical frameworks like `R` we can write micro-edits (see Listing 1.1) to check obvious consistency of individual records (e.g., like shown in Table 1). Typically, variable names in edits of Listing 1.1, like `age` and `height`, must match column names of the data to be checked (see Table 1). The edit rules may constrain the values of a variable and the dependency between its values and values of other variables. Therefore, in general, micro-edits can decide if an individual record is consistent or not by

[3]See `http://www.linkededitrules.org/`

```
dat1 : ageGroup %in% c('adult', 'child', 'elderly')
dat7 : maritalStatus %in% c('married', 'single', 'widowed')
num1 : 0 <= age
num2 : 0 < height
num3 : age <= 150
num4 : yearsMarried < age
cat5 : if(ageGroup == 'child') maritalStatus != 'married'
mix6 : if(age < yearsMarried + 17) !(maritalStatus %in% c('married','widowed'))
mix7 : if(ageGroup %in% c('adult', 'elderly') age >= 18
mix8 : if(ageGroup %in% c('child', 'elderly') & 18 <= age) age >= 65
mix9 : if(ageGroup %in% c('adult', 'child')) 65 > age
```

Listing 1.1: Examples of micro-edits in the R `editrules` package.

considering just one record at a time. In the examples in Table 1 and Listing 1.1, record `#2` is inconsistent with edits `cat5` and `mix6`; record `#3` with edits `num4` and `mix6`; record `#4` with edit `num3`; and record `#5` with edits `num2`, `cat5` and `mix8`.

	age	**ageGroup**	**height**	**maritalStatus**	**yearsMarried**
#1	21	adult	6.0	single	-1
#2	2	child	3	married	0
#3	18	adult	5.7	married	20
#4	221	elderly	5	widowed	2
#5	34	child	-7	married	3

Table 1: Example dataset to be validated against obvious inconsistencies.

On the other hand, *macro-edits* are rules aimed at discovering inconsistencies through the analysis of multiple records. These rules can have a very different nature. For instance, the *aggregation method* [6] consists of checking whether aggregates on the data (e.g., the total population count of a country) match their logical decomposition into individual records (e.g., adding up the population totals of every individual municipality). The *distribution method* [1] asserts knowledge about the distribution of a certain variable (e.g., "population counts must be log-normal distributed") and identifies all individual records that do not fit that distribution. Macro-edits can only decide if an individual record is consistent or not by performing a double-pass on the dataset. Hence, it is clear that languages tailored to micro-edits (see Listing 1.1) are insufficient to express the semantics of macro-edits. We investigate whether Semantic Web rule languages are adequate for representing micro- and macro-edits.

Edit rules are not currently published on the Web in a structured way. They are neither linked to the cubes and observations they intend to validate, nor to the dimensions they constrain. Users cannot uniquely reference, retrieve or combine them to validate RDF Data Cubes. Validation workflows are therefore kept closed within NSOs systems or private user frameworks, affecting their openness, referenceability, reusability, linkage and exchange. To overcome these pitfalls, we define a set of **requirements**:

– Expressing edit rules in a **Web friendly**, accessible and standard way:
  - **[WebDistrib]** Edit rules should be published *on the Web in a distributed way*, separately from (but linked to) the statistical data they apply to.

  - **[WebStructured]** Edit rules need to be expressed in a *Web standard structured data format* to ensure their machine-readability.
  - **[UniqID]** Edit rules, and their components, need to be *uniquely identified* in order to be unambiguously referenceable.
- Edit rules as **Semantic Web rules enriched with metadata**:
  - **[WebRules]** Edit rules should be expressed in currently available and adequate *formal Semantic Web rule languages.*
  - **[ConstrainedDims]** Edit rules need to be explicit about, and uniquely reference to, the statistical *dimensions they constrain.*
  - **[Scope]** Edit rules need to be explicit about their *scope* and aggregation level (micro/macro).
- **Checking obvious consistency** combining rules and reasoning:
  - **[Reasoning]** Edit rule checking should be transparently integrated into standard *reasoning* and established consistency checking mechanisms in currently available triplestores and inference engines, in order to preserve interoperability of current infrastructure.
  - **[CubeCheck]** Users need to be able to express *which edit rules* they want to be checked against *which data cubes.*
- Creating interoperable **consistency reports** using Web standards:
  - **[Prov]** *Provenance* of the execution of edit rules should be explicit, traceable, and represent accurately the consistency check workflow.
  - **[Annotation]** Edit rules must *annotate* inconsistent data points for posterior expert correction, preserving the raw data. Statisticians are interested in concrete inconsistent data points rather than a consistent/inconsistent response over the entire dataset.

Consequently, our problem definition is *to identify obvious inconsistency of arbitrary RDF Data Cubes by executing (micro and macro) edit rules following Linked Data principles in an open Web environment.*

3 Related Work

Data editing is "the activity aimed at detecting and correcting errors (logical inconsistencies) in data" [18]. Traditionally, edits have been considered at two levels: *micro-edits*, aimed at finding inconsistencies in individual records; and *macro-edits*, "based upon analysis of an aggregate rather than an individual record" [4]. More recently, automatic processing of these edits has gained importance [10]. For example, the `R` packages `editrules` (validation through micro-edits) and `validate` (validation through cross-record and cross-dataset macro-edits) are useful tools to automatically validate locally stored statistical datasets. However, these rule formats are not Web standard (**[WebStructured]**) nor suitable for efficient Web distribution (**[WebDistrib]**). Eurostat proposes the eDAMIS Validation Engine (EVE) to validate statistical datasets defining intra (micro) and inter (macro) record rules, but these are kept internally in the system. Finally, the SDMX initiative proposes the Validation and Transformation Language (VTL) . However, VTL (1) has a strong emphasis on *transformation* instead of *validation*

(**[CubeCheck]**); and (2) defines no mechanism to publish nor link edit rules on the Web (**[WebDistrib], [WebStructured], [UniqID]**). No existing approach tackles explicitly requirement **[Scope]**, nor generates interoperable validation reporting (**[Prov], [Annotation]**).

The related work on the Semantic Web can be divided in (a) rule languages in which edit rules could be expressed (**[WebRules]**); and (b) initiatives and systems for customized validation of linked and semantic data (**[Reasoning]**). On rule languages, OWL 2 [17] allows the definition of Description Logics (DL) *safe* rules. The Semantic Web Rule Language [9] (SWRL) extends DL rules allowing Horn clauses, although limiting the creation of new individuals in the ABox to avoid infinite rule loops. SPARQL [7] can also be used to express rules as `CONSTRUCT` queries, although these are lost after execution (**[UniqID]**). The Rule Interchange Format Basic Logic Dialect [2] (RIF-BLD) partially overcomes this by allowing the expression of rules in RDF, and particularly serializing SPARQL as RDF. The SPARQL Inference Notation [11] (SPIN) also uses SPARQL to express and store a variety of business rules [13]. Shape Expressions [14] associate RDF graphs with labeled patterns called *shapes*, and can be used for validation, documentation and transformation of RDF data. Similarly, Resource Shapes [15] specify the properties that are allowed or required in a resource, their value types, and cardinality. With respect to validation systems, OWLIM Profiles allow customization of rules for RDF data validation, although these custom rules must be hard coded, hampering their reuse (**[WebStructured], [UniqID]**). TopBraid [16] allows customization of business rules via SPIN. Finally, Stardog [3] follows a polyglot approach and users can write SPARQL queries, OWL axioms, or SWRL rules for integrity constraint validation against RDF data. The RDF Data Cube vocabulary [5] (QB) defines 22 integrity constraints that Web-linked cubes must meet, implemented in SPARQL `ASK`, but these are meant to preserve the constraints of QB and are not useful to model domain-dependent rules.

4 Approach

4.1 Linked Edit Rules and RDF Data Cube

Meeting requirements **[WebStructured]** and **[UniqID]** of Section 2 is straightforward when applying Linked Data principles. Concretely, we propose (1) to represent edit rules in RDF, in order to publish and link edit rules using a Web structured-data standard format; and (2) to use URIs to denote edit rules and their components in RDF to uniquely identify and de-reference them.

To meet requirements **[ConstrainedDims]** and **[Scope]**, we analyse the RDF Data Cube (QB) data model and propose the minimal set of *LER extensions* shown in Figure 1. The LER vocabulary can be found at `http://bit.ly/linked-edit-rules#`. The class `ler:EditRule` represents an edit rule and can be subclassed by any rule model. A `ler:EditRule` has three fundamental components: a `ler:body`, a `ler:scope`, and one or many `ler:components`. The `ler:body` represents the rule itself, encoded according to a specific rule language. The `ler:scope` represents the scope of the edit, and we use it to distinguish *micro-edits*

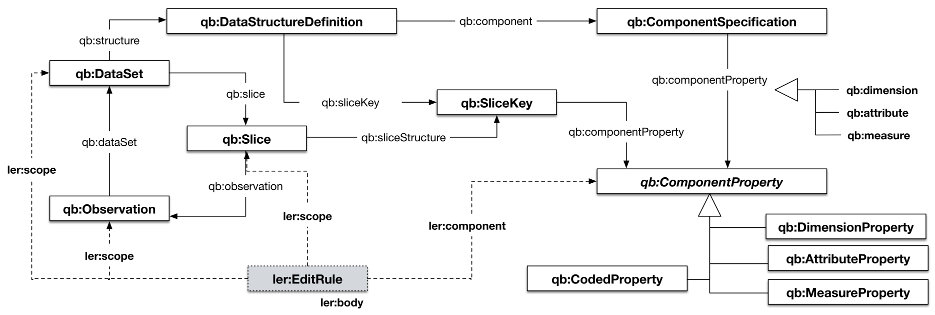


Fig. 1: RDF Data Cube data model (straight lines), and our proposed *LER extensions* (dashed lines) to define Linked Edit Rules.

and *macro-edits*. Since micro-edits are defined at the individual record level, and individual records are represented as `qb:Observations` in QB (see Figure 1), we define micro-edits in our model as any `ler:EditRule` with `qb:Observation` as `ler:scope`. Likewise, QB allows the representation of groups of observations (records) according to some criteria as slices (`qb:Slice`), as well as the complete set of observations of the cube (`qb:DataSet`) (see Figure 1). We define macro-edits as any `ler:EditRule` with `qb:Slice`, `qb:ObservationGroup` or `qb:DataSet` as `ler:scope`. Finally, we link the `ler:EditRule` to all the statistical variables it constrains through `ler:component`. In QB, statistical variables are represented using the subclasses of `qb:ComponentProperty` (typically `qb:DimensionProperty`).

4.2 From edit rules to Linked Edit Rules

In order to meet requirement **[WebRules]** we study the expressiveness of micro-edits and macro-edits.

Body of Linked Micro-Edits. *Definite Horn clauses* are clauses (disjunctions of literals) with exactly one unnegated literal, $\neg p \vee \neg q \vee ... \vee \neg t \vee u$, usually written in implication form as $p \wedge q \wedge ... \wedge t \rightarrow u$. This includes the case of no negative literals, also called *facts* (e.g., u). Definite Horn clauses are known to be tractable in Semantic Web rule languages. Micro-edits (see Listing 1.1) fit definite Horn clauses if we consider each literal $p, q, ..., u$ as an *inequality*. An inequality is an expression that involves variables, numeric constants and algebraic operators, and has exactly one of the symbols $[>, <, \leq, \geq, =, \neq]$. For example, ($\texttt{age} = \texttt{currentYear} - \texttt{bornYear}$) and ($\texttt{height} \geq 11.5$) are inequalities. Micro-edits `num1` to `num4` in Listing 1.1 are *facts*, composed of one inequality (literal). Rules `cat5` and `mix7` to `mix9` do have negative literals and consequently a rule body. We can express the special construct v `%in%` $l_1, ..., l_n$ as the clause $v = l_1 \vee ... \vee v = l_n$. This makes rule `mix6` problematic because of a positive conjunction of literals in the rule's head. Since variables in the rule's body are stable during execution, we solve this using normalization, splitting `mix6` in several rules with the same body as `mix6` and only one of the positive literals in the head.

To express micro-rules as Linked Edit Rules, we convert all variables and values of Listing 1.1 to RDF URIs and literals. We substitute each variable name by URIs of `qb:DimensionProperty` (e.g., replacing `age` by `sdmx-dimension:age`). We substitute string literals by their equivalents in known `skos:ConceptScheme` (e.g., `married, widowed` to `sdmx-code:status-M, sdmx-code:status-W`). Finally, we replace all numeric values by RDF literals.

Extensions for Macro-Edits. Macro-edits need specific statistical terms to be used in the rules. For instance, $X \sim \mathcal{N}(\mu, \sigma^2)$ (where X represents *heights* in the cube of Table 1) is a valid macro-rule that states that this dimension must follow a normal distribution with mean μ and variance σ^2. To encode such constraints, we extend the above rules for micro-edits to macro-edits, by adding the following function as a valid member of literals (i.e., allowing it in inequalities):

$$\textit{statistic}(t, \mathcal{P}, \mathcal{S}, c), \text{ where:}$$

- t is a test statistic to assess the constraint (e.g., the `z.test` normality test)
- $\mathcal{P} = \{p_1, ..., p_n\}$ are the parameters of t (e.g., mean μ and variance σ^2)
- $\mathcal{S} = \{s_1, ..., s_m\}$ are the sets of observations that must adhere to the rule (e.g., all observations of the cube; two particular slices)
- c is the constrained *dimension* of those observations (e.g., `eg:height`)

We use the function *statistic* to describe statistical properties of observation groups in macro-edits, and we embed this function in micro-edit-like clauses (as described above) to express macro-rules as Linked Edit Rules. We use the output of this function (often a p-value) to express the meaning of the macro-edit. For example, we rewrite the macro-edit that checks whether heights of all persons in Table 1 follow a normal distribution as follows, assuming `eg:all` to be the URI of all cube observations, and normal distribution of heights as null hypothesis:

$$\textit{statistic}(\texttt{z.test}, \{\mu, \sigma^2\}, \texttt{eg:all}, \texttt{eg:height}) > 0.05$$

4.3 LER Architecture

Edit rules and *data cubes* may be published in different locations on the Web, but both types of resources need to be combined to validate cubes against obvious inconsistency. To achieve this, we propose a two-component based architecture: *(a)* a node with arbitrary RDF Data Cube; and *(b)* a node with Linked Edit Rules. The workflow is:

1. The user sends query Q to *(a)*, asking which *data cube* entity $e \in E$ of *(a)* (instances of `qb:Observation`, `qb:Slice` or `qb:DataSet`) *is inconsistent with* which rule $r \in R$ (instances of `ler:EditRule`) (see Section 4.1).
2. The cube triplestore *(a)* retrieves rules R from *(b)* as indicated in Q. If rules R are distributed among more nodes, *(a)* proceeds iteratively.
3. The cube triplestore *(a)* updates its *inference engine* with R.
4. The cube triplestore *(a)* performs custom inference as specified by R to decide which $e \in E$ is inconsistent with which $r \in R$.
5. The cube triplestore *(a)* returns to the user: (1) data cube entities $e \in E$ annotated with the $r \in R$ they are inconsistent with; (2) the provenance graph of the execution.

5 Implementation

We implement the Linked Edit Rules (LER) method described in Section 4 using Stardog, using its custom reasoning capabilities to check obvious consistency of Web RDF Data Cubes. Stardog allows this customization supporting rules in multiple formats and models, including SPARQL, OWL axioms, and SWRL.

We implement Linked Edit Rules `ler:EditRule` with *micro-edits* as Stardog Rules `rule:SPARQLRule`. In Stardog, rules are defined using SPARQL Basic Graph Patterns (BGPs), plus the decorators `IF-THEN`, denoting the body and the head of a rule. Listing 1.2 shows how rule `mix6` of Listing 1.1 translates into a Stardog Rule (other rules are published at `http://www.linkededitrules.org/`). To match the data to be validated in RDF Data Cube, the triple pattern in the `IF` clause must select the appropriate rule scope and components. To obtain commonly used URIs for such components we use LSD Dimensions[4] [12]. When the knowledge base is queried in SL reasoning mode and the `IF` clause holds, the `THEN` clause is executed and violation triples are inferred. Finally, we include `ler:scope` and `ler:component` triples to fit the LER extensions (see Section 4.1). Other metadata describe the original rule form, its author and the creation date. We also implement *macro-edits* as Linked Edit Rules, preserving the rule format of micro-edits. Listing 1.2 shows a macro-edit that checks if heights of adults and non-adults in Table 1 follow different statistical distributions. To implement the *statistic* function of Section 4.2 and make statistical tests available in Stardog Rules, we develop a Stardog extension[5] that wraps `R` function calls using SPARQL Extensible Value Testing (EVT). All Linked Edit Rules of this paper (e.g., translations of edit rules in Listing 1.1) are published on the Web as Linked Data at `http://www.linkededitrules.org/`.

To implement the LER Architecture of Section 4.3, we read the data cubes to be validated, we retrieve their associated LER via SPARQL, and we add these LER to Stardog's rule base; these will be triggered whenever Stardog is queried. Stardog query rewriting and rule normalization mechanisms split rules when multiple triples are found in the `THEN` clause. This hampers the generation of provenance and annotation graphs, since new instances (e.g., of class `prov:Activity`) cannot reference the specific data points and rules used for consistency checking. To solve this, we use rules that create only one `ler:inconsistentWith` statement, as shown in Listing 1.2, and we trigger these rules using a SPARQL `INSERT` query, as shown at the bottom of Listing 1.2. Using this query, we materialize provenance and annotation reporting using PROV and the Open Annotation Data Model (OA), avoiding to rewrite the BGP for provenance and annotation generation. Finally, to generate the output we use a SPARQL `CONSTRUCT` query that creates user reports with provenance and annotation graphs describing all inconsistencies found (see Figure 2). We bundle the entire consistency checking procedure in *QBsistent*[6], a customizable LER-QB consistency checking tool.

[4] `http://lsd-dimensions.org/`

[5] `https://github.com/albertmeronyo/stardog-r`

[6] See `http://www.linkededitrules.org/`

```
# Micro-edit
leri:mix6 a rule:SPARQLRule, ler:EditRule;
  rule:content """ # PREFIX definitions
    IF {
      ?obs a qb:Observation.
      ?obs sdmx-dimension:civilStatus ?civilStatus.
      ?obs eg:yearsMarried ?yearsMarried.
      ?obs sdmx-dimension:age ?age.
      FILTER ((?age < ?yearsMarried + 17) && (?civilStatus = sdmx-code:status-M || ?
           civilStatus = sdmx-code:status-W))
    } THEN {
      ?obs ler:inconsistentWith leri:mix6.
    } """;
  ler:scope qb:Observation;
  ler:component sdmx-dimension:age, sdmx-dimension:civilStatus, eg:yearsMarried;
  rdfs:label "if(age < yearsmarried + 17) !(status %in% c('married', 'widowed'))";
  rdfs:comment "An underage can't be married nor widowed";
  dc:creator <http://www.albertmeronyo.org/>;
  dc:date "2015-01-08T16:03:40+01:00"^^xsd:dateTime.

# Macro-edit
leri:macro1 a rule:SPARQLRule, ler:EditRule;
  rule:content """ # PREFIX definitions
    IF {
      ?x a qb:Slice .
      ?x qb:sliceStructure eg:sliceByAdults .
      ?y a qb:Slice .
      ?y qb:sliceStructure eg:sliceByNonAdults .
      FILTER(stardog:R('wilcox.test', ?x, ?y, eg:height) <= 0.05)
    } THEN {
      ?x ler:inconsistentWith leri:macro1 .
      ?y ler:inconsistentWith leri:macro1 .
    } """;
  ler:scope qb:Slice ;
  ler:component eg:height .
  rdfs:label "dist(X) != dist(Y), X heights of adults, Y heights of non-adults";
  rdfs:comment "Heights of adults and non-adults follow different distribs.";
  dc:creator <http://www.albertmeronyo.org/>;
  dc:date "2015-01-08T16:03:40+01:00"^^xsd:dateTime.

# Generating PROV and OA
INSERT { ?act a prov:Activity;
           rdfs:label "Consistency check";
           prov:wasAssociatedWith <http://stardog.com/>;
           prov:startedAtTime ?now;
           prov:used ?dp;
           prov:used ?rule .
         ?ann a oa:Annotation;
           rdfs:label "Inconsistency annotation";
           prov:wasGeneratedBy ?act;
           prov:generatedAtTime ?now;
           oa:hasBody ?body;
           oa:hasTarget ?dp .
         ?body a rdfs:Resource;
           ler:inconsistentWith ?rule.
         BIND (UUID() AS ?act, UUID() AS ?ann, UUID() AS ?body, now() AS ?now)
} WHERE { ?dp ler:inconsistentWith ?rule . }
```

Listing 1.2: Linked Edit Rules as Stardog Rules. The predicate `rule:content` describes the content of the rule, while other triples describe the rule metadata. For macro-edits, statistical tests are accessed via SPARQL custom functions through an `R` wrapper we implement as a Stardog extension. The bottom `INSERT` query triggers Linked Micro- and Macro-Edit Rules in Stardog, generating PROV and OA for each inferred inconsistency.

6 Evaluation

We validate our implementation by studying: (1) the semantic equivalence of edit rules and Linked Edit Rules; (2) their precision and recall at identifying inconsistencies; and (3) insights provided by the implementation of edit rules as Linked Data. We use toy and synthetic datasets, and the QB representations of the Dutch historical censuses[7] and the 2010/2011 French and Australian censuses[8]. Full details on results are available at `http://www.linkededitrules.org/`.

To assess a correct translation between the original edit rules and Linked Edit Rules and show their equivalence (1), we provide translation mappings between edit rules and SPARQL in Table 2a. We stress-test our Linked Edit Rules implementation on correctly identifying inconsistencies (2) in two different datasets: (a) the toy data in Table 1; and (b) a synthetic dataset with artificial data. In both cases we know beforehand where the inconsistencies are: they are trivial in (a) (see Section 2), and we introduce them intentionally in (b). In both cases, our implementation proves to identify correctly all expected inconsistencies, as shown in Table 2b ($p = r = 1.0$). Interestingly, the consistency check process takes only a small fraction (0.5%) of the total runtime, which accounts mostly to overhead due to initialization and data fetching. We also showcase interesting Linked Data features of LER. We run our approach on a real-world dataset, the Dutch historical censuses. Our goal is to identify potential inconsistencies to aid the curation process of the dataset maintainers. We use a set of census domain LER: (i) each population observation must have, at most, one occupation position (occupation positions distinguish, e.g., business managers from ordinary labour) [micro-edit]; (ii) population counts must be integer numbers [micro-edit]; (iii) population counts must be positive [micro-edit]; and (iv) population counts must meet Benford's Law [macro-edit]. We find that, out of 5,429,104 observations, 244,406 (4.50%) are inconsistent with respect to (i), accounting for two or more occupational positions; 30,317 (0.56%) count population with non-integer numbers (ii); and 909 (0.02%) do so with negative numbers (iii). Since these LER apply to any census data, we use their Linked Data features to test (ii), (iii) and (iv) against the French (4,848,096 observations) and Australian (12,650 observations) census editions of 2010-2011 (we skip (i) since these contain no labour position data). In both cases, no observation is inconsistent with (iii), although all of them are inconsistent with (ii), most probably due to data anonymization and normalization. All Dutch, French and Australian datasets fit Benford's Law (iv). Figure 2 shows the PROV graph of a consistency check and the generated inconsistency annotations using the Open Annotation DM (OA).

7 Discussion and Future Work

In this paper we present Linked Edit Rules (LER), a method to express edit rules as Linked Data and use them to validate arbitrary RDF Data Cubes on the Web. Our proposal and implementation of using currently available

[7] `http://lod.cedar-project.nl/cedar/`

[8] `http://www.datalift.org/en/event/semstats2013/challenge`

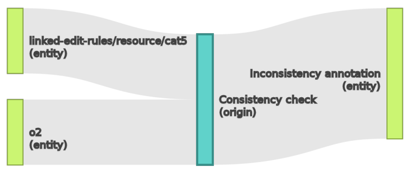
Edit rule	SPARQL
`x > `$t, t \in \mathbb{R}$	`FILTER(?x > `t`)`
`x %in% (`$v_1, ..., v_n$`)`	`FILTER(?x = `v_1` || ... || ?x = `v_n`)`
`if(x) y`	`FILTER(¬?x || ?y )`
$X_s \sim \mathcal{N}(\mu, \sigma^2)$	`FILTER(statistic(`$t, \mathcal{P}, \mathcal{S}, c$`))`

(a) Translating edit rules into SPARQL LER. The ¬ operation in SPARQL means inverting the inequality contained in the literal.

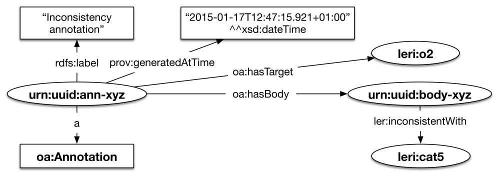
Dataset	Size	Inc.	p	r	Reas.	Overh.
`editrules`	5	8	1.0	1.0	0.005	0.995
Synthetic	70,700	131	1.0	1.0	0.005	0.995

(b) Size of tested datasets (number of observations), number of found inconsistencies, precision, recall, and proportion of execution time devoted to consistency checking (Reas.) and remaining overhead (Overh.) using LER.

Table 2: LER correctness validation: equivalence of edit rules and SPARQL LER, and stress-testing.



(a) PROV-O-Viz [8] diagram showing a consistency check of observation `leri:o2` and LER `leri:cat5` to produce an annotation.



(b) Generated OA annotation of a detected inconsistency in a `qb:Observation`, linking observation `leri:o2` with the LER `leri:cat5`.

Fig. 2: Reporting obvious inconsistency using Web standards for provenance and annotations.

Semantic Web rule standards fulfills the requirements of Section 2. We show that expressing micro-edits using Semantic Web rule languages is possible, by using SPARQL syntax and Stardog's Rule Reasoning. We design a generic *statistic* function that provides a statistical vocabulary of multi-observation tests, enabling the expression of Linked Macro-edits. Expressing edit rules as Linked Data provides: (a) extensive and concise *provenance* (PROV) reports that *annotate* (OA) detected inconsistencies without modifying the source data, enabling interoperable consistency-check reporting; and (b) the ability to link edit rules with constrained Web dimensions and measures, enabling an easy reuse of edit rules, as shown with diverse datasets in Section 6. Arguably, a simpler implementation could be designed using SPARQL only (e.g., by leveraging `CONSTRUCT` and query federation), but important requirements such as **[UniqID]** would break, hampering the reusability of rules. Conversely, we combine Linked Data, rules on the Web, custom reasoning, SPARQL querying, `R` functionality and provenance generation to check quality of data cubes. A limitation of our macro-rule implementation is that observations cannot be arbitrarily selected using the rule's BGP body. To do so it is necessary to implement the function *statistic* (see Section 4.2) as a custom SPARQL *aggregation function*. This is problematic, as (1) such functions are part of SPARQL's grammar, and (2) their customization is currently not supported in Stardog. We solve this by using the links of `qb:Slice` and `qb:DataSet` to the observations they contain, transparently processing these observations without custom aggregation functions. Arbitrary selections of observations must be explicitly asserted in the graph as `qb:Slices`.

We plan to extend LER in several aspects. First, we are interested in checking consistency of rule sets by themselves, before running consistency against data. Second, we intend to automatically retrieve relevant rules given an RDF Data Cube Data Structure Definition (DSD). Third, we plan on publishing our

QBsistent tool as a web service. Fourth, we want to provide a LER editor to facilitate rule editing and publishing. Last, we will study the genericity of our approach by implementing it in other domains.

Acknowledgements This work has been supported by the Computational Humanities Programme (`http://ehumanities.nl`) of the Royal Netherlands Academy of Arts and Sciences and the Dutch national program COMMIT. Special thanks go to Frank van Harmelen, Andrea Scharnhorst, Veruska Zamborlini, Wouter Beek, Steven de Rooij and Rinke Hoekstra.

References

1. Bethlehem, J.: Applied Survey Methods: A Statistical Perspective. Wiley (2009)
2. Boley, H., Kifer, M.: RIF Basic Logic Dialect. Tech. rep., World Wide Web Consortium (2013), `http://www.w3.org/TR/rif-bld/`
3. Complexible, Inc.: Stardog 3.1.4. `http://stardog.com/` (2015)
4. Cox, N., Croot, D.: Data editing in a mixed DBMS environment. Statistical Journal of the United Nations Economic Commission for Europe 8(2), 117–136 (1991)
5. Cyganiak, R., Reynolds, D., Tennison, J.: The RDF Data Cube Vocabulary. Tech. rep., World Wide Web Consortium (2013), `http://www.w3.org/TR/vocab-data-cube/`
6. Granquist, L.: Macro-editing – A review of some methods for rationalizing the editing of survey data. Statistical Journal of the United Nations Economic Commission for Europe 8(2), 137–154 (1991)
7. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. Tech. rep., World Wide Web Consortium (2013), `http://www.w3.org/TR/sparql11-query/`
8. Hoekstra, R., Groth, P.: PROV-O-Viz - Understanding the Role of Activities in Provenance. In: 5th International Provenance and Annotation Workshop (IPAW 2014). LNCS, Springer-Verlag, Berlin, Heidelberg (2014)
9. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Tech. rep., World Wide Web Consortium (2004), `http://www.w3.org/Submission/SWRL/`
10. de Jonge, E., van der Loo, M.: An introduction to data cleaning with R. Tech. rep., Statistics Netherlands (2013), discussion paper
11. Knublauch, H.: SPIN – Modeling Vocabulary. Tech. rep., World Wide Web Consortium (2011), `http://www.w3.org/Submission/spin-modeling/`
12. Meroño-Peñuela, A.: LSD Dimensions: Use and Reuse of Linked Statistical Data. In: Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2014 (2014)
13. O'Riain, S., McCrae, J., Cimiano, P., Spohr, D.: Using SPIN to formalise XBRL Accounting Regulations on the Semantic Web. In: Proceedings of the First International Workshop on Finance and Economics on the Semantic Web (FEOSW 2012). Extended Semantic Web Conference (ESWC) (2012)
14. Prud'hommeaux, E.: Shape Expressions 1.0 Primer. Tech. rep., World Wide Web Consortium (2014), `http://www.w3.org/Submission/2014/SUBM-shex-primer-20140602/`
15. Ryman, A.: Resource Shape 2.0. Tech. rep., World Wide Web Consortium (2014), `http://www.w3.org/Submission/2014/SUBM-shapes-20140211/`
16. TopQuadrant, US: TopBraid Composer. Features and getting Started Guide Version 1.0 (2007), `http://www.topbraidcomposer.com/`
17. W3C OWL Working Group: OWL 2 Web Ontology Language. Tech. rep., World Wide Web Consortium (2012), `http://www.w3.org/TR/owl2-overview/`
18. de Waal, T., Pannekoek, J., Scholtus, S.: Handbook of Statistical Data Editing and Imputation. Wiley (2011)