



UNIVERSITÀ POLITECNICA DELLE MARCHE
Repository ISTITUZIONALE

Analytics for citizens: A linked open data model for statistical data exploration

This is the peer reviewed version of the following article:

Original

Analytics for citizens: A linked open data model for statistical data exploration / Diamantini, Claudia; Potena, Domenico; Storti, Emanuele. - In: CONCURRENCY AND COMPUTATION. - ISSN 1532-0626. - 33:8(2021). [10.1002/cpe.4186]

Availability:

This version is available at: 11566/248520 since: 2024-04-24T16:00:11Z

Publisher:

Published

DOI:10.1002/cpe.4186

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. The use of copyrighted works requires the consent of the rights' holder (author or publisher). Works made available under a Creative Commons license or a Publisher's custom-made license can be used according to the terms and conditions contained therein. See editor's website for further information and terms and conditions.

This item was downloaded from IRIS Università Politecnica delle Marche (<https://iris.univpm.it>). When citing, please refer to the published version.

(Article begins on next page)

Analytics for Citizens: a Linked Open Data Model for Statistical Data Exploration

Claudia Diamantini, Domenico Potena and Emanuele Storti

*Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche,
via Brecce Bianche - 60131 Ancona*

SUMMARY

A growing number of public institutions all over the world has recently started to make government statistical data available in open formats, thus enhancing transparency and accountability, stimulating innovation and promoting civic awareness and engagement. Integration issues related to fragmentation and heterogeneity of these datasets can be partially addressed by referring to the Linked Data approach, which also enables easier access and consumption by users. However, the lack of an explicit representation of how statistical indicators are calculated still hinders their interpretation, and hence the development of applications and services especially useful for citizens, who do not have full knowledge and control over the underlying data and analysis models. In the present work, we discuss an approach to ease the interaction of communities of citizens with statistical Linked Open Data. We define a model and a set of services allowing people to recognize the mathematical structure of statistical indicators, improving in this way user-awareness of the meaning of indicators and their mutual relations. Through such services, it is possible to enable interactive browsing of indicator formulas and novel typologies of data exploration, including dynamic computation of indicators not explicitly stored and comparison of different Linked Data resources. Copyright © 0000 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: Linked Open Data; mathematical formulas; ontology of indicators; data exploration; logic-based reasoning functionalities

1. INTRODUCTION

During the last years, the Open Government Data (OGD) movement has advocated opening up governments' and public authorities' data to allow their free use, reuse and redistribution [1]. Access to open data makes it possible to increase government transparency and accountability by keeping citizens aware about government activities. Releasing data in open formats drives also the birth of novel services and businesses, by stimulating innovation and contributing to economic growth as a by-product. Recently, an increasing number of public institutions all over the world has indeed started to make available their data in this form, e.g. data.gov or Eurostat, just to mention a few, include datasets ranging from public transportation to environmental pollution and energy consumption, to immigration or expenses for health care. As a consequence, this is promoting civic engagement of citizens and communities of interests, with the ultimate goal of improving government processes by collaboratively developing proposals and initiatives. Such communities, whose participatory processes may include both online and offline activities, typically welcome contributions from a plethora of users with very diverse and complementary skills (domain experts, computer scientists, techies, associations, journalists), each with their own interests and specific capabilities (e.g., coding, communication).

*Correspondence to: Emanuele Storti, e.storti@univpm.it

Despite the increasing enthusiasm in this field, the full potential of Open Government Data has not however been realized yet, partially because of the fragmentation and heterogeneity of OGD datasets, and lack of common publications standards [2], with the consequence that a consistent integration effort is required before data can be actually accessed and (re)used. A way to deal with such issues is related to the adoption of Linked Data, an approach to publish data on the web, enabling datasets to be linked together through references to common concepts. This allows to improve the opening, the linking and the reusing of Open Government Data, thus limiting interoperability issues and hence simplifying data visualization and use [3]. In this way, data are represented through the Resource Description Framework (RDF) format, where each datum is identified by a URI and can be linked with other relevant data, thus creating a larger knowledge graph that can be accessed and queried both in a human- and machine-understandable way.

However, analysis of aggregated and statistical data can still remain quite challenging, in particular with distributed data from autonomous sources. Different sources can analyze the same phenomenon in different ways, for instance referring to various representation models, standards, terminologies and units of measure, or different levels of granularity [4]. Other conflicts arise from the multidimensional and aggregate nature of statistical information. In this respect, specific RDF vocabularies have been proposed, like Data Cube [5], that are tailored to represent statistical data on the web with a model that is compatible with the multidimensional cube model. Even in this case, some issues may still arise at measure level because statistical datasets usually lack a precise definition of the mathematical meaning of indicators (or measures), and then the same indicator may be calculated differently by each data source. Without an explicit representation of this information, the risk is to obtain inaccurate outputs from the analysis, compare inconsistent results and take the wrong conclusions from the data. How to compare, for instance, two datasets with data about *total emissions* of pollutants if we do not know which pollutants have been considered, and which is the calculation formula?

To this aim, we take as requirements (1) a machine-readable representation of indicators and their formulas, that can be formally specified and shared within a community of users; on the other side, (2) the capability to automatically manipulate such formulas, in order to enable comparisons among their algebraic structure. This can be used for instance to understand whether a formula can be rewritten as another, and hence to determine their equivalence; finally, (3) services tailored to final users, which are needed to support high-level analysis.

Taking into account these considerations, the high-level goal of this paper (which is an extended version of [6]) is to propose an approach to ease the interaction of citizens with statistical Linked Open Data. To address the mentioned requirements, in this work we define an open and machine-understandable format enabling the definition of a shared, collaboratively built repository of reusable indicator definitions. Secondly, we developed a series of services allowing users to recognize their mathematical structure, improving in this way the awareness of the meaning of indicators and their mutual relations, and perform comparisons among different datasets.

In more details, the main aspects of the approach and its major contributions are the following:

- 1) we formally model mathematical expressions representing formulas for indicators following the Linked Data approach with an open, formal and shareable vocabulary, which is compatible with the RDF Data Cube model and capable of describing their structure in terms of other indicators. To the best of our knowledge, this is the first proposal of an RDF vocabulary for the definition of indicator formulas, specifically in the context of the Data Cube model.
- 2) On the top of this model, we define Prolog-based reasoning capabilities formalizing the basic mathematical axioms for manipulation of formulas. This enables services for consistency checking, formula and dependencies inference, and dynamic calculation of formulas for data retrieval.
- 3) Interactive browsing of indicators formulas and novel typologies of data exploration are enabled by these reasoning services, including dynamic computation of indicators not explicitly stored and comparison of different Linked Data resources. This feature is

innovative with respect to traditional systems, in which the formula is not usually formally given and therefore no manipulation is possible.

Given the diverse skills and competencies usually available in a community, the overall objective of this approach can be achieved only through collaboration among users. Indeed, the provided services are meant to be used by diverse typologies of users: while definition and browsing of indicators and their mathematical formulas may be in charge of users with a mathematical background, domain experts could help in indicator analysis and interpretation.

This paper extends our previous work [6] in many respects. Here, we provide a more extensive description of the proposed methods and models, in particular: we extend the methodology to cover statistical repositories expressed as Linked Open Data and following the Data Cube vocabulary, given the increasing interest in public data repositories represented in this format; we provide a thorough discussion of approaches for representation of mathematical formulas in RDF; we discuss a wider set of services exploiting the model for (i) the definition of customised metrics, which is of help in the setup of analytical tasks, (ii) result explanation, to gain a more in-depth understanding of the output of the analysis and of the structure of indicators, and awareness of the degree of reliability of results, which is of particular interest in open distributed contexts, and (iii) comparison of different alternative data sources.

The rest of the work is organized as follows. In the next Section, we briefly introduce a case study that will be used throughout the paper. In Section 3 we report on the main background knowledge of this work, namely on the multidimensional model, mathematical languages for the web and formal frameworks of indicators and measures. Section 4 is aimed to describe our model for the representation of mathematical formulas of indicators according to the Linked Data approach, and to compare it with other related work. In Section 5 we discuss how the model can be exploited for increasing user awareness on statistical Linked Data through mathematically meaningful data analytics. These services rely on a set of logical reasoning services based on Prolog, discussed in Section 6, that can manipulate mathematical expressions, check consistency and perform structural analysis on formulas. Finally, Section 7 provides some final remarks and outlines future work.

2. CASE STUDY: ANTHROPOGENIC ATMOSPHERIC EMISSIONS

A directive of the European Parliament* sets upper limits for each Member State for the total emissions of the four pollutants responsible for acidification, eutrophication and ground-level ozone pollution, namely sulphur dioxide, nitrogen oxides, volatile organic compounds and ammonia. At European level, Eurostat[†] is in charge of publishing and keeping up-to-date thousands of statistical datasets as open data for institutions of the European Union. Among them, we refer here to datasets about the four pollutants, aggregated by country, year, sector of emissions and expressed in tonnes (the dataset's identifier is in round brackets):

- 1) emissions of sulphur oxides, or SO_x (tsdpc260)
- 2) emissions of nitrogen oxides, or NO_x (tsdpc270)
- 3) emissions of non-methane volatile organic compounds (tsdpc280)
- 4) emissions of ammonia, or NH₃ (tsdpc290)

and also (5) to a dataset about total population per European country and year (tps000001). Hereafter we name such measures respectively by the terms SOXEmissions, NOXEmissions, NonMethaneVolatileOrgEmissions, NH3Emissions and TotalPopulation. Published data refer to a time span of 10 years (from 2004 to 2013), 33 countries and 9 sectors of emissions (e.g., energy production and distribution, energy use in industry, road and non-road transport, industrial processes and others). To provide an example, in Table I we report a small fragment of the dataset tsdpc260

*Directive 2001/81/EC <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2001:309:0022:0030:EN:PDF>

[†]<http://ec.europa.eu/eurostat>

	2009	2010	2011	2012	2013
Spain	181.758	421.112	455.254	402.310	287.128
France	156.429	285.310	249.006	235.467	218.785
Croatia	40.012	34.754	29.010	24.814	16.378
Italy	118.124	214.869	194.193	174.865	145.054
Cyprus	16.074	21.962	20.953	16.249	13.766

Table I. A fragment of dataset tsdpc260 with values (in tonnes) about emissions of sulphur oxides, for years 2009-2013 and all sectors of emissions.

including values about SOXEmissions for 5 countries from 2009 to 2013, aggregated over all sectors of emissions.

In the following, we consider a scenario in which a user wants to monitor two new measures: the total emissions by the four pollutants, i.e. $TotalEmissions = SOXEmissions + NOXEmissions + NonMethaneVolatileOrgEmissions + NH3Emissions$, and its normalization with respect to the population, calculated as $TotalEmissionsPerCapita = \frac{TotalEmissions}{TotalPopulation}$. Given that Eurostat datasets are only available in tabular form, in the following we consider their translation into the Data Cube format[‡], realized by Linked Data Research Center at DERI.

3. BACKGROUND

In this Section we provide the main background knowledge of this work, namely the multidimensional model, RDF vocabularies for its representations according to the Linked Data approach, languages for modeling mathematical knowledge in the web and frameworks for management and monitoring of indicators.

3.1. Multidimensional model and Linked Data

Statistical data are typically conceived as a collection of values observed for a given phenomenon, contextualized according to a set of dimensions. In this context, the notion of *measure* (or indicator) is used as a quantitative metric enabling the monitoring of a fact, while a *dimension* is the coordinate/perspective along which a measure is computed. Following the multidimensional model [7], a dimension is usually structured into a hierarchy of levels $L_1 \preceq \dots \preceq L_n$, where each level L_i represents a different way of grouping elements of the same dimension. For instance, a “place” dimension can be organized in countries, states, provinces and municipalities, while a “time” dimension can be arranged in years, semesters, quarters, months and days. Each level is instantiated in a set of elements known as members of the level, e.g. the country “Italy” or the semester “2016-S1”. The notions of data cube and multidimensional perspective are used for analysis of statistical data, where each point in a data cube represents the measurement of one (or more) measure(s) and its coordinates represent certain aggregation levels for each of the dimensions.

Among the standards for representation and exchange of statistical data on the web, one of the most prominent is the “Statistical data and metadata exchange” (SDMX) [8], by the International Organization for Standardization (ISO). Also the Data Documentation Initiative (DDI)* is aimed at establishing an XML-based standard for describing and documenting statistical and social science data, to improve interpretation, machine-actionability and interoperability. Motivated by the need for general, flexible solutions to modeling and publishing statistical data on the Web, several RDF vocabularies have been proposed in the last years.

[‡]The url of each dataset is given by <http://eurostat.linked-statistics.org/dsd/> followed by the corresponding code, e.g. <http://eurostat.linked-statistics.org/dsd/tsdpc260.ttl> for the dataset about emissions of sulphur oxides

*<http://www.ddialliance.org/>

As a light-weight solution, SCOVO (Statistical Core Vocabulary) [9] addresses the basic use case of expressing statistical data in RDF, by allowing easy adoption by data producers and consumers. Due to its minimalist design, SCOVO cannot however support typical scenarios occurring in statistical publishing, like distinguishing between dimensions, attributes and measures, defining the structure of a dataset independently from concrete data, or grouping together datasets that share the same structure.

To address such limits, SCOVO, that is now deprecated, has been superseded by the Data Cube vocabulary (QB) [5], a proposal by the W3C Government Linked Data Working Group that allows to publish statistical data on the web as RDF following the Linked Data principles. The QB language models the schema of a cube as a set of dimensions, attributes and measures through the corresponding classes. On the other hand, a cube instance including data points is represented in QB as a set of `qb:Observations`, that are typically grouped in subsets named Slices. To make an example about the case study of Section 2, the `qb:DataStructureDefinition` about SOXEmissions contains as `qb:component` the following dimensions:

- `sdmx-dimension:timePeriod`, the time of emission (from 2004 to 2013)
- `property:airsect`, for the specific industrial/business sector producing the emissions (e.g., energy production and distribution, industrial processes, road transport)
- `sdmx-dimension:freq`, the frequency (annual)
- `property:geo`, the source of emissions (e.g., Italy, UK, Spain)
- `property:unit` the unit of measure (tonnes)

and a `qb:attribute` `property:obs_status` to describe the status of the corresponding observation (eg., reliable, estimated, missing). The following is an example of observation in Turtle[†]:

```
<http://eurostat.linked-statistics.org/data/tsdpc260#A,T,SE1_CIH,AT,1990>
  a qb:Observation ;
  sdmx-measure:obsValue 1883900.00 ;
  sdmx-dimension:freq sdmx-code:freq-A ;
  sdmx-dimension:timePeriod ""1990-01-01""^^xsd:date;
  property:airsect dic-airsect:TOT_NAT ;
  property:geo dic-geo:IT ;
  property:unit dic-unit:T;
  qb:dataSet data:tsdpc260.
```

Although QB can define the structure of a cube, it does not provide a mechanism to represent dimension levels and the relationships among them. In fact, although some hierarchical relationships between members of dimensions can be represented, by referring to the SKOS vocabulary, the language does not provide means to model dimension levels and their mutual relation (e.g., for a temporal dimension, the relations among levels Day, Month and Year). As a consequence, roll-up/drill-down operators, which rely on such hierarchical relations among levels, are not supported (see also [10]). This is partially motivated by the fact that QB was originally conceived for analyzing statistical data with a very general model. For this reason the Data Cube vocabulary can be used for various typologies of data sets such as survey data and spreadsheets, but not to support full-fledged Business Intelligence systems operating on Linked Data. Nonetheless, some approaches have been proposed to implement OLAP operations on QB. Among them, solutions that resort to traditional OLAP query technologies by loading of RDF triples into a data warehouse [11] and approaches involving execution of nested sets of OLAP operations as SPARQL queries directly on single data cubes, although taking into account only one level and hierarchy per dimension [12].

With the aim to overcome the modeling issues reported above, the QB4OLAP vocabulary [10] is conceived as an extension to the Data Cube vocabulary allowing to represent OLAP cubes in RDF.

As further development, QB4OLAP associates aggregate functions to measures and, unlike QB, it also allows to implement OLAP operators as SPARQL queries. Although our case study is

[†]<https://www.w3.org/TR/turtle/>

represented in Data Cube, the model and the services described in this work are compliant with both these languages.

3.2. Mathematics in the web: MathML and OpenMath

Many solutions have been proposed to represent mathematical text in a computer system. The most notable ones, among the earliest, include SGML (Standard Generalized Markup Language) and the TeX typesetting system developed by Donald Knuth [13], a de-facto standard within the mathematics community for its high-quality visual rendering of formulas.

The development of a new language was motivated by the need to provide a representation of mathematics on the web as a standard means for information sharing, that could be easily read, processed and generated using commonly available tools.

To this aim, two languages emerged in the last years, namely OpenMath and MathML, whose approaches can be considered as highly complementary more than alternative, in that the former is focused on the semantic meaning of mathematical objects, while the latter is more concerned with their presentation. OpenMath [14] is an extensible XML-based standard for representing the semantics of mathematical objects. OpenMath *basic objects* include integers, floats, strings, byte arrays, variables and symbols, while *compound objects* include, amongst others, applications which are composed of one or more OpenMath objects. Symbols can be linked to an external reference belonging to a Content Dictionary (CD), i.e. a collection of symbols and their definitions expressing their meaning. Different CDs are available in the standard OpenMath for various subsets of mathematics, including linear algebra, polynomials and group theory, transcendental functions, combinatorics and many others, although new CDs can be defined at need. OpenMath can be encoded as an XML document, in order to ease the exchange and storage of mathematical expressions, where <OMA> is the encoding for applications, <OMS> for symbols, <OMI> for integers, <OMF> for floats and <OMV> for variables. In general, <OMA> contains an <OMS> specifying the operation and a set of arguments, among which constants like <OMI> or <OMF>, variables <OMV> or, recursively, an <OMA>. To give an example, the XML representation of the equation $TotalEmissionsPerCapita = \frac{TotalEmissions}{TotalPopulation}$ is the following:

```
<OMOBJ xmlns='http://www.openmath.org/OpenMath'
        cdbase="http://www.openmath.org/cd">
  <OMA>
    <OMS cd='relation1' name='eq' />
    <OMV name='TotalEmissionsPerCapita' />
    <OMA>
      <OMS cd='arith1' name='divide' />
      <OMV name='TotalEmissions' />
      <OMV name='TotalPopulation' />
    </OMA>
  </OMA>
</OMOBJ>
```

In the example, the Content Dictionary “relation1” includes the “eq” symbol for the equality relation, while “arith1” includes arithmetic operators like “divide”.

On the other hand, efforts towards a language capable of capturing both presentation of mathematical formulas and their semantics led to the development of the *Mathematical Markup Language*, or MathML [15], by W3C. Semantics in MathML is described through OpenMath Content Dictionaries, and a subset of MathML3, namely Strict Content, can be directly mapped to OpenMath.

Apart from these languages, repositories of mathematical functions have been developed in the web with the aim to enhance sharing and reuse of formulas. To the best of our knowledge, the most comprehensive repository is maintained by Wolfram Research[‡]. With more than 307 thousand

[‡]<http://functions.wolfram.com/>

mathematical functions, this compendium is the largest encyclopedic collection of information about formulas and graphs, available both in *Mathematica*[§] and in MathML notations.

3.3. Approaches to representation of mathematical expressions in RDF

Motivated by the increasing availability of mathematics-related Linked Data resources, including for instance statistical government data or databases of scientific results, in the last years some research effort has been put into the modeling of mathematical semantics of the expressions. Several approaches were driven by the need to express formulas in RDF as a machine-understandable format that can be possibly automatically manipulated more than just shared. For a comprehensive review on the topic we refer to [16]. Although many steps need to be done in order to properly bridge queries and reasoning in the Semantic Web with formal calculus required for mathematics, hereafter we report on some of the most significant work. They are aimed towards the RDF representation of MathML or OpenMath, ranging from a straight mapping of XML tags to RDF [17], to the extension of existing languages [18, 19] to a translation from XML languages to RDF [20–22].

Indeed, SPARQL 1.1. supports simple mathematical operations like aggregation, and can be used to compute sums or averages of a set of elements represented in RDF. However, to provide more complex computation, some work proposes to extend RDF with custom constructors to express mathematical functions. Following this approach, the author of [18] defines new constructors, e.g. *math:Power*, in Turtle to represent a formula in RDF. In this way, for instance, $\int x^2 + 1 \, dx$ can be represented as “*math:Integral(math:Plus(math:Power(:x,2),1), :x)*”. A corresponding extension is proposed for SPARQL, in order to enable queries to retrieve formulas with certain requirements. In a similar way, authors of [19] extend SWRL[¶] with rules to perform mathematical computations on OpenMath formulas embedded into OWL ontologies.

Following a different approach, Robbins [20] proposes a minimal translation of MathML in RDF. The formal semantics of a Content MathML expression is defined by referring to the equivalent Strict Content MathML expression. In turn, the semantics of this last is expressed, as stated above, in terms of OpenMath Content Dictionaries, from which we define in this work some rules for mapping to RDF. Motivated by the increasing amount of Linked Data resources that could benefit from mathematical support, Wenzel et al. [21] propose an RDF translation of OpenMath. The work is aimed towards the consumption of mathematical Linked Data from computer algebra systems, with extended reasoning systems with inferencing capabilities based on mathematical computations. Finally, [22] shares with our work the same application scenario, namely analysis of government Linked Data. As the authors point out, published data very often do not make semantics of statistical data points explicit. Their approach involves using an ontology to ground statistical data to existing vocabularies and mathematical functions, also to derive additional data points. The latter is achieved by using an ontology, to model how to derive new data items from existing ones relying on OpenMath for the semantics of operators. In order to avoid the extension of standard languages like SPARQL, that would require an extra effort for the adaption of existing tools and reasoners, our approach follows the direction of this work, as explained in Section 4.

3.4. Frameworks of indicators

Indicator management is a hot topic especially in disciplines related to Enterprise Management, where usually the so-called “(Key) Performance Indicators” (PI or KPI) are exploited to monitor the degree of achievement of strategic enterprise goals, with the same meaning of the indicators we consider in this paper. Many reference models (e.g., Supply Chain Operations Reference model (SCOR) [23] or the Value Reference Model (VRM)^{||}) as well as independent dictionaries or libraries were introduced by researchers and international public bodies, witnessing the attention towards a systematization and organization, although informal, of the huge amount of existing PIs. Formal

[§]<http://www.wolfram.com/mathematica/>

[¶]<https://www.w3.org/Submission/SWRL/>

^{||}<http://www.value-chain.org/en/cms/1960>

models of indicators were recently proposed [24, 25] in the context of the performance-oriented view of organizations, even though in these models representation of formulas does not rely on logic-based languages, and hence their manipulation is not possible. Furthermore, the models are conceived for the definition of PIs in a single process-oriented enterprise, and the issue of consistency management is not addressed. In [26] the notion of *composite indicators* and their representation in a tree structure is introduced, together with their calculation with full or partial specification of the formula relating the indicator to its components. In most of these papers, formula representation does not rely on logic-based languages, hence no inference mechanism and formula manipulation is possible.

In the OLAP and Data Warehouse fields, semantic representations were recently proposed, mainly to reduce the gap between the high-level business view of indicators and the technical view of data cubes, supporting the automatic generation of customized Data Marts and OLAP analysis. However, while the ontological representation of the aggregation structure of PIs is largely studied [27–29], the compound nature and consequent dependency among indicators is much less explored. Despite the increasing interest in collaboration and networking there is still a lack of work addressing data cubes in distributed and collaborative contexts. A proposal in this respect is [30], where interoperability of heterogeneous and autonomous Data Warehouses is taken into account. To the best of our knowledge, our approach (firstly published in [31] and used in this paper for Linked Statistical Data) is the first to propose an ontology aimed to support a collaboratively built repository for PIs.

4. MODELING OF FORMULAS

In a previous work of ours [31] we introduced KPIOnto, an ontology to formally describe Performance Indicators. According to the terminology in use within the Performance Monitoring field, we refer to (performance) indicator as the metric (or measure) enabling an analytical task. KPIOnto has been used to support integrated performance monitoring across multiple distributed organizations within a Virtual Enterprise (VE) environment. Performance data, like statistical data, are typically described by resorting to the multidimensional model, as each measured value can be represented as a point in the hypercube defined by a set of dimensions, which in turn include sets of hierarchically organized levels. For this reason the ontology includes the notion of indicator (i.e., a measure), dimension, level and member, that were used to semantically annotate performance values available in enterprise repositories (full specification is available at <http://kdmg.dii.univpm.it/kpionto>). As such, some overlap can be recognized between KPIOnto and QB4OLAP, although they have been developed for different purposes (integration of performance indicators in distributed settings vs. representation of statistical Linked Data) and referring to different scenarios (Virtual Enterprise vs. Linked Open Data produced by public administrations). In KPIOnto an indicator is detailed through a set of properties, which include name, identifier, acronym, definition, compatible dimensions, formula, unit of measurement chosen for the indicator (i.e., both the symbol and the description, by referring to the Measurement Units Ontology*), and aggregation function. A `kpi:Indicator` can be almost completely mapped to the corresponding `qb:MeasureProperty` as defined in QB. Indeed, indicator properties can be represented through `qb:AttributeProperties`, while the aggregation function is explicitly introduced in QB4OLAP through the class `qb4o:AggregationFunction`, that is not available in QB. In the following, we describe the approach we take in this work for the representation of formulas, that involves the extension of QB vocabulary with the fragment of KPIOnto devoted to describe formulas. In such a way, a `qb:MeasureProperty` in a QB/QB4OLAP dataset can be linked to a corresponding `kpi:Indicator` in KPIOnto which will provide a formula for its computation from other indicators, and hence from other `MeasureProperties` in the same or in other datasets.

*Measurement Units Ontology: <http://idi.fundacionctic.org/muo/muo-vocab>

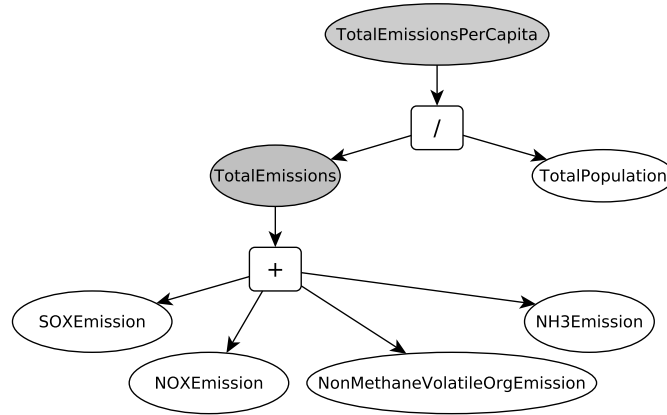


Figure 1. Graph of dependencies for the Eurostat example (compound indicators are in gray).

4.1. Definition of a PI formula

A KPIOnto indicator can be either atomic or compound, built by combining other indicators. Dependencies of a compound indicator *ind* on its building elements are defined through a mathematical expression $f(s_1, \dots, s_n)$, i.e. a *Formula* expressing how the indicator is computed in terms of s_i , which are in turn indicators or constants [31]. In more detail, given the expression $ind = f(s_1, \dots, s_n)$, the right-hand side is a *well-formed formula* for *ind* obtained from the recursive definition given below:

- every numeric constant is a well-formed formula;
- every atomic indicator is a well-formed formula;
- if $\{s_1, \dots, s_k\}$ are well-formed formulas and *op* an operator of arity *k*, then $op(s_1, \dots, s_k)$ is a well-formed formula.

Since we do not deal with generic mathematical expressions but instead with formulas of Performance Indicators, well-formed indicator formulas can be represented by closed-form analytical expressions.

An indicator can be assigned at most one formula, in order to avoid multiple definitions. The set of formulas can be graphically represented by a *graph of dependencies* among indicators, i.e. a forest of disjoint lattices where each node is an indicator and its children are the operands of its formula, while leaves are atomic indicators. Figure 1 shows the graph of dependencies for the Eurostat example discussed in Section 2, where gray nodes represent compound indicators, while white nodes are atomic indicators.

4.2. RDF representation of PI formulas

In KPIOnto the class `kpi:Indicator` is in relation to a class `kpi:Formula` through the property `kpi:hasFormula`, with maximum cardinality of 1 (see Figure 2). In order to express the specific mathematical expression for a `kpi:Formula`, in this work we follow the approach proposed by [22], in which generic OpenMath expressions are encoded in RDF by referring to a specific vocabulary capable representing relevant OpenMath elements. In that work, such a vocabulary is the SCOVOLink ontology[†], proposed for the representation of mathematical formulas in statistical databases expressed in SCOVO, in order to improve their discoverability, reusability and semantic integrability. SCOVOLink allows to associate a statistical value `scovo:Item` to a generic `sl:Operation` (corresponding to `<OMA>`) through property `sl:computedFrom`. A

[†]<http://vocab.deri.ie/scovolink>

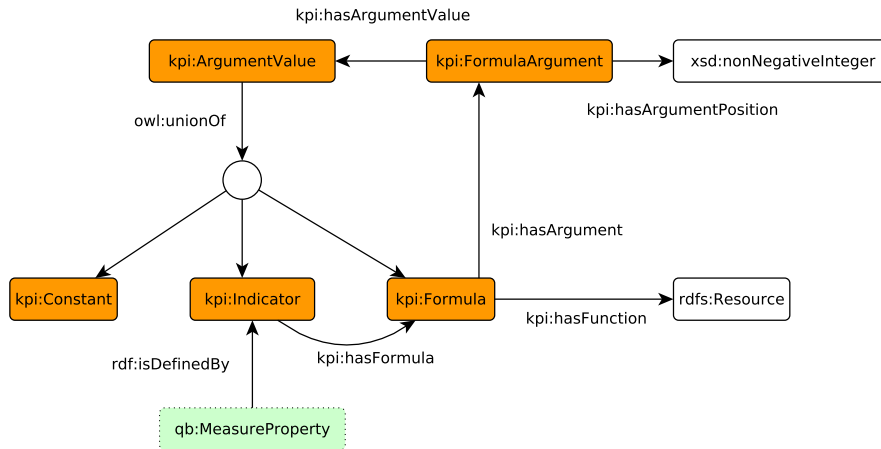


Figure 2. Classes and properties for RDF representation of an OpenMath formula (solid lined boxes stand for KPIOnto classes, the dotted lined box for a QB class, while white boxes represent XSD datatypes).

`sl:Operation` has a function (i.e. an OpenMath symbol represented by an `rdfs:Resource`) and has one or more `sl:Argument`. This last can specify an argument position (to declare its position in the formula) and an argument value, which is again an `rdfs:Resource`.

In our work we however deal with a different degree of details, i.e. we refer to formulas for indicators, valid for more than a single dataset, and not for specific data points in a dataset. Hence, given that our model cannot directly extend/reuse SCOVOLink, and that SCOVO has been superseded (see Subsection 3.1), we define a set of customized terms to represent elements of a formula according to the OpenMath model. These terms do not arise from a straightforward one-to-one mapping between OpenMath and RDF, which would be too language-oriented and less focused on the conceptual model of a formula. In the following, we separately discuss the translation of operands and operators:

- **Operands.** An operand is the object for a mathematical operation. From the definition of well-formed formula given in the previous subsection, here an operand can be an `kpi:Indicator` (i.e., a variable <OMV>), a `kpi:Constant` (either an integer <OMI> or a float <OMF>) or another `kpi:Formula` (i.e., a <OMA>), so that each formula can contain reference to one or more indicators. In turn, class `kpi:Constant` is defined as the union of integer and float datatypes (e.g., `owl:equivalentClass [owl:unionOf (xsd:float xsd:nonNegativeInteger)]`). In our model, we arrange all possible operands under the class `kpi:ArgumentValue`.
- **Operators.** To facilitate the reuse of OpenMath operators (i.e., <OMS>) within a URI-based framework like RDF, the OpenMath documentation [14] provides the following scheme for constructing a canonical URI for an OpenMath Symbol: `cdbase-value + '/' + cd-value + '#' + name-value`, where `cdbase-value` is the url (<http://www.openmath.org/cd> for the standard OpenMath Content Dictionaries), `cd-value` is the content group that includes the specific symbol (e.g., `arith1` for arithmetic operators or `set1` for set operations), while `name-value` is the symbol name (e.g., `plus`, `divide`, `power`, `subset`). For instance, <http://www.openmath.org/cd/arith1#divide> is the URI for the division operator.

The following properties have also been defined, as shown in Figure 2:

- `kpi:hasFunction` (object property), to specify an operator for a `kpi:Formula`. The operator is expressed as an `rdfs:Resource` pointing to its definition in a specific Content Dictionary.

- `kpi:hasArgument` (object property), to declare a `kpi:FormulaArgument` for a `kpi:Formula`. Its properties are:
 - `kpi:hasArgumentPosition` (datatype property), the position of the `kpi:FormulaArgument`, to determine its order;
 - `kpi:hasArgumentValue` (object property) the value of an argument, which can be either an indicator, a constant (integer or float) or another formula (e.g., `rdfs:range [a owl:Class ; owl:unionOf (kpi:Indicator kpi:Constant kpi:Formula)]`).

Finally, a `qb:MeasureProperty` can be linked to the corresponding indicator definition through `rdfs:isDefinedBy` property. An example showing the RDF representation of the Eurostat formulas of the case study is available in Appendix A.1.

4.3. Evaluation

The expressiveness of the RDF model proposed in this Section depends on which types of mathematical expressions can be represented and manipulated. As for the representation, we refer to OpenMath expressivity as a formula can be decomposed in an application of an OpenMath operator to a set of arguments. In our model we consider unary, binary and n-ary operators and we include the order of the operator in the formula (which is needed in mathematical functions, whenever the role of the operand is qualified, i.e. divisor versus dividend). With respect to the previous work by Lange et al. [16], upon which the proposed RDF modeling of formulas has been elaborated, we provide a more general, abstract representation which is self-contained in the model, while they represent a custom formula as a new function with indicators as arguments. By following recursive decomposition of a formula, in our model there is no need to resort to externally defined functions. Moreover, as a further distinction, our aim is related on the definition of general-purpose, reusable dictionaries of formulas. Indeed, formulas are not meant to be defined inside the QB/QB4OLAP datasets, and, as such, their definition can be adopted by multiple datasets by just adding a triple to link the measure property to the corresponding indicator. Finally, simpler representations of formulas are obviously possible, for instance the direct encoding of OpenMath/MathML expressions as `rdf:XMLLiteral`s, as with an earlier version of KPIOnto [31]. However, the current model enables a more uniform and explicit representation of formulas, in which all the dependencies of an indicator are linked through properties. As a major benefit, this enables reasoning on the formula structure, hence making dependency relations browsable and discoverable through standard SPARQL queries, with no need for specific OpenMath parsers.

5. SERVICES TO SUPPORT CITIZEN AWARENESS AND DATA ANALYTICS

Awareness of how indicators are calculated is a precondition to make meaningful considerations about data. Without an explicit representation of indicators' formulas the risk is to produce inconsistent results, derive wrong consequences and take ineffective decisions. This is critical for enterprise scenarios, where decision making must rely on accurate and reliable data, but also for governments and public administrations, where statistics and data nowadays play the role of guidance for political initiatives, and finally for civic activism, where public campaigns aiming at political objectives start from evidences in data.

In this Section we discuss how the model can be exploited to provide support for mathematically meaningful data analytics distributed across several data sources and ultimately for increasing user awareness. The services hereby described include (1) more technical functionalities mainly aimed to support mathematically skilled users to define new indicators and manipulate their formulas and (2) more high-level services for comparison and analysis, targeted to domain experts. Both rely on a set of logical reasoning functionalities that will be discussed in Section 6.

5.1. Definition of custom indicators

For a new indicator, the following steps are to be performed to create a new indicator and its formula in a local repository, given a dataset represented as Linked Data according to the Data Cube vocabulary:

- 1) create the new indicator as an instance of `kpi:Indicator`;
- 2) create its formula in OpenMath and convert the formula in RDF;
- 3) check consistency, to verify if the new formula is coherent with or equivalent to already existing formulas;
- 4) link the measureProperty in the dataset at hand to the new indicator definition.

As for step 2, by resorting to the standard language OpenMath, several available tools can be used as formula editor. The conversion between OpenMath and RDF is realized through a set of services, including:

- `OM_to_RDF(openMath_representation)`, to codify formulas according to the RDF model starting from its OpenMath representation;
- `RDF_to_OM(rdf_representation)`, to retrieve the OpenMath representation from RDF triples, typically for presentation purposes.

Alternatively, given its popularity and for compatibility reasons, also MathML3 can be used as its Strict Content subpart can be directly mapped to OpenMath. In this case, through XSLT (eXtensible Stylesheet Language Transformations) it is possible to convert to and from MathML3, as well as from MathML3 Content to MathML3 Presentation (for visualization in compatible browsers). Step 3 concerns consistency checking, which is required to assure that the new formula does not contradict, from a mathematical point of view, other previously defined formulas. To implement such a functionality we resort to a `check_consistency` and other logic-based reasoning functions that will be discussed in Section 6 and are available as support services.

The possibility to define new customized indicators can contribute to increase the overall degree of user awareness. Indeed, if the user repository is a knowledge base shared within a community instead of a local repository, the contribution of a mathematically skilled user in the definition of a formula can be a valuable benefit also for others. Users with mathematic background could for instance reuse an indicator to define a new formula and check its consistency. For example, through the consistency check service a user can discover that its newly defined formula $NormalizedTotalEmissions = \frac{SOXEmissions}{TotalPopulation} + \frac{NOXEmissions}{TotalPopulation} + \frac{NonMethaneVolatileOrgEmissions}{TotalPopulation} + \frac{NH3Emissions}{TotalPopulation}$ is actually equivalent to $TotalPopulationPerCapita$, and therefore reuse the existing definition. Domain experts, on the other hand, could exploit the new indicator to make dynamic analysis and comparisons (see below).

5.2. Analysis of custom indicators

Linked Data repositories that are available either as RDF files or through an endpoint can be queried through the SPARQL query language. If such repositories are compliant with the Data Cube format, the query can be expressed in a standard form, by referring to elements of the QB vocabularies to express dimensions, measures and observations. Obviously, if a given indicator is not associated to any dataset, no query can be expressed. However, by exploiting our approach, a query about a compound indicator can be dynamically composed by exploiting the representation of its formula in terms of other indicators. Indeed, if every atomic indicator in a formula has a related dataset, then it is possible to build a query for its calculation. By considering the Eurostat example, we defined two compound indicators, and linked the four indicators about pollutants to corresponding datasets. In Appendix A.2 we give an example of federated SPARQL query performed across the five Eurostat datasets, that calculates the values for indicator `TotalEmissionsPerCapita`, for each year available in datasets (2003 to 2012), for all sectors of emissions and for country Italy (represented by the code "IT").

In order to make this model exploitable for data analysis, a number of conditions must be met for indicators belonging to the same formula:

- all indicators in a formula should share the same dimensional schema, i.e. they are measured along the same dimensions. It may be acceptable to have different schemas, but the query must be defined against the intersection of all dimensional schemas, i.e. over the set of their common dimensions.
- Level hierarchies and member sets must be compatible each other. If not, the query can be posed only with respect to corresponding levels and members.
- Unit of measurements of indicators and their scaling must be the same. If not, proper transformations must be applied. These can be expressed as formulas, i.e. $AvgDailyTemperature_Celsius = \frac{AvgDailyTemp_Fahrenheit - 32}{1.8}$, and $GDP_Dollars = GDP_MillionDollars * 10^6$. Please note that the unit of measurement can be expressed also through a proper KPIOnto property ($\forall kpi:hasUnitOfMeasurement.kpi:UnitOfMeasurement$, see also [31]).

In case these conditions cannot be satisfied, an integration phase is needed to reconcile heterogeneous dimensional schemas, dimension hierarchies or different sets of members. Please note that the formal, explicit representation of formulas proposed in this work, apart from supporting the functionalities shown in this Section, also addresses the heterogeneity at measure level, by recognizing indicators with the same name but different formulas, or indicators with same formula and different definitions [32].

5.3. Diving into the graph of formulas

The availability of explicit formulas for indicators enables a set of support functionalities aimed at increasing awareness of how indicators are computed and how the operands of their formulas (i.e., their dependencies) affect their values. In most cases, however, users lack of support to interpret data. To address this issue, the exploitation of the proposed model enables a novel type of operator, namely *indicator drill-down*, which relies on formula manipulation functionalities and reasoning. Like the usual drill-down, indicator expansion increases the level of detail of a measure in the data cube, but instead of disaggregating along the levels of dimensions, it expands an indicator into the components of its formula. This allows users both (1) to browse formulas in order to understand how indicators are related to each other, and (2) to perform a rewriting of the query through `expandIndicator` service (see Section 6). In this last case, the two notions of drill-down are then integrated, hence allowing a novel way of exploring data cubes. For instance, let us suppose a user wants to conduct an investigation to understand the reasons behind a certain negative trend (e.g., a decrease of the values for a certain year) for `TotalEmissions`. By using this service it is possible to break down the formula and analyze the contribution of each pollutant separately.

Similarly, other typologies of analysis can be performed to obtain all dependencies of a given indicator, through service `getDependencies` to get more insights from data.

5.4. Comparisons among data sources

In a distributed setting with multiple datasets it is important to understand if and to what extent data are reliable. In many cases it depends on the quality of data gathering, cleaning, production and publication processes. Here we consider data completeness and consistency as measures of reliability, and show how the model and some reasoning services can be used for its evaluation.

As for data completeness, by using the functions mentioned in the previous subsection, users that are domain experts can understand for instance if a certain negative trend for an indicator is due to the decrease of some pollutants or just because some data points are missing and the calculation has been done only on fewer components, e.g. on three pollutants out of four. In case of queries across multiple datasets, then, availability of comparable data values becomes critical. In fact, if the sources are very sparse and with many missing values, the capability to combine them and calculate a compound indicator is limited. As an example, for `TotalPopulation`, data before 2003 and after 2014 are missing, while all the datasets about emissions include data from 1990 to 2012. Hence, the calculation of `TotalPopulationPerCapita` can be done only for the intersection of these ranges, i.e. from 2003 to 2012. Ranges like these can be analyzed even before running a

query, through some exploratory queries aimed at evaluating maximum and minimum values for ranges (e.g., for time dimension) or listing the available members for a dimension (e.g., all the countries considered in the dataset). See [33] for an example of how to automate this task.

A further type of evaluation can be done regarding *data consistency*: when multiple alternative data sources are available for the same `kpi:Indicator` and refer to the same data, such sources can be compared in order to check whether they are consistent and comparable. For instance, we compared Eurostat data with data provided by data.gov.uk about SOX emissions* and we recognized several incongruities by running the same query over the two sources.

Apart from this type of direct comparison between datasets referring to the same indicator, domain experts can also exploit the model to examine alternative data sources, to check if and how they differ. For instance, the total emissions per capita as calculated by two organizations, in order to verify whether they are identical or differ in something.

Finally, different data sources may exist for a certain phenomenon but containing compound indicator data. In these last cases, the availability of explicit formulas becomes important as no dataset for such indicators is actually published and values can only be calculated dynamically by exploiting the `expand_indicator` service (see Section 6). For instance, emissions in Europe versus emissions in USA. Let us assume that, beside the formula for `TotalEmissions`, also a new formula for an indicator `TotalEmissions_USA` is defined to describe a dataset produced by US administration:

- $TotalEmissions_USA = SOXEmissions + NOXEmissions + NH3Emissions$

It is easily recognizable that the two formulas (and therefore the two indicators) are not equivalent, as in the second the contribution of the pollutant `NonMethaneVolatileOrgEmissions` is not taken into account. To address issues like this, in our approach we exploit the model and the reasoning service `get_common_dependencies` in order to derive the (possible) implicit relations existing between formulas, or formulas that are included (like in this case) in other formulas. In the example, the reasoner concludes that $TotalEmissions_USA = TotalEmissions - NonMethaneVolatileOrgEmissions$. Only once such a relation is discovered, a meaningful comparison between the two indicators can be realized by executing a proper query, e.g., by removing the contribution of non-methane volatile emissions from the `TotalEmissions` indicator, or by adding (if available) the same contribution to the `TotalEmissions_USA`.

6. REASONING ON FORMULAS

A set of logic-based functionalities are defined to enable an easy and transparent management and browsing of the indicator formulas that have been translated from their RDF representation. Indeed, for practical usage of formula representation in RDF, a mathematical application is needed to interpret and apply the formula. For many common computer algebra systems, a set of mappings (or phrasebooks) between their native language and OpenMath has been provided, while in more complex cases the OpenMath operator could be decomposed (if possible) in more elementary functions. The viability of this approach is proven by the efforts made to define standard protocols for OpenMath-aware computation services. For instance, SCSCP (Symbolic Computation Software Composability Protocol) [34] allows exchange of expressions, request of calculation and storage of resources by mathematical applications. Although libraries for SCSCP or other protocols are available for a variety of computer algebra systems, in this work we developed customised math-aware services to provide special non-standard reasoning capabilities. In particular, we refer to Prolog as logic language for its versatility, capability of symbolic manipulation as well as for the wide availability of well-documented reasoners and tools. In the following we show how to translate RDF formulas to Prolog facts, and we detail the main reasoning services.

*https://data.gov.uk/dataset/emissions_of_air_pollutants. Please note that this dataset is published in tabular form and was converted by us in QB for test purposes.

6.1. Converting formulas from RDF to Prolog

A language translator is devised to convert an RDF expression of a formula to an infix notation understandable by Prolog. For operators, the translation is realized by means of mapping rules between OpenMath Content Dictionary operators and functions in Prolog (arithmetic operators, logarithm, trigonometric functions, square-root, hyperbolic functions). By referring to the previously defined prefix “om:” for the OpenMath namespace, these are a few examples of mappings:

- om:divide \rightarrow “/”
- om:plus \rightarrow “+”
- om:power \rightarrow “^”

On the other hand, indicator URIs are managed as Prolog atoms and as such they are left unchanged. Finally, a Prolog term `ind_formula(indicator, formula, type)` is introduced to state that *formula* is an infix representation of the formula for *indicator* (where all operators have been properly translated), and *type* specifies whether the indicator is atomic (“leaf_node”) or not (“branch_node”). For instance, given the indicator *TotalEmissionsPerCapita*, the corresponding Prolog representation of $TotalEmissionsPerCapita = \frac{TotalEmissions}{TotalPopulation}$ is as follows: `ind_formula('TotalEmissionsPerCapita', 'TotalEmissions'/'TotalPopulation', 'branch_node')`.

6.2. Reasoning services

In the following we discuss the main reasoning services reported in Table II, that are devised to support the higher-level functionalities discussed in Section 5. Services are arranged in a set of packages according to their purpose, namely mathematical manipulation, consistency check and formula graph analysis. We refer the interested reader to previous work [31, 35] for further details on some of the services.

Table II. Main reasoning services.

Service	Description
<code>solve_equation(equation,x)</code>	Solves a generic <i>equation</i> w.r.t. the variable <i>x</i>
<code>consistency_check(ind,formula))</code>	Checks the consistency of a new <i>formula</i> for <i>ind</i> with respect to the others in the repository
<code>expand_indicator(ind)</code>	Returns the formula defined for the indicator <i>ind</i>
<code>get_formulas(ind)</code>	Computes the set of all equivalent formulas for <i>ind</i>
<code>get_dependencies(ind)</code>	Returns all the indicators that are direct or indirect dependencies of <i>ind</i>
<code>get_common_dependencies(ind₁,ind₂)</code>	Computes the set of all indicators that are in common between <i>ind₁</i> and <i>ind₂</i>

6.2.1. Mathematical manipulation. A first type of reasoning service is devoted to manipulate formulas according to strict mathematical axioms including commutativity, associativity and distributivity of binary operators, and properties of equality needed to solve equations. For this we refer to a set of (more than 900) predicates from PRESS (PRolog Equation Solving System) [36], a formalization of algebra in Logic Programming. Although not originally meant as a conventional algebra manipulation system, PRESS is however capable of solving symbolic, transcendental and non-differential equations in one or more variables. These services are capable of analyzing the formula at hand, and to rewrite the formula in order to achieve a specific syntactic effect, such as solving equations (with one variable or polynomial), reducing the number of occurrences of a variable or moving a variable to one side.

6.2.2. Consistency Check. The service for consistency check allows to automatically check if a certain indicator is *consistent* (unique, inequivalent and coherent) with the other previously defined. More formally, given a new indicator ind_i , its formula $formula_i$ is consistent with the others if and only if the following conditions hold for the equation $ind_i = formula_i$: (1) the equation is not equivalent to any other (inequivalence) and (2) the equation does not contradict any other (coherence). Please note that a particular case for equivalence occurs when the new formula is identical to another already existing in the knowledge base. Three specific predicates are executed in sequence to support such a verification, namely `identical(ind, formula)`, `equivalent(ind, formula)` and `incoherent(ind, formula)`, which respectively return the list of indicators whose formulas are syntactically identical, mathematically equivalent or incoherent with the one at hand. Although inequivalence subsumes uniqueness, the latter condition is explicitly checked as first for optimization reasons, in order to immediately detect if the new formula is identical to an already existing one, without proceeding with a full equivalence check. Please note that if a set of formulas are different rewriting of the same formula through associative, commutative or distributive properties, they are all considered as equivalent.

Typically, these predicates are used before a new `ind_formula` fact is added to a repository. Hence, after every insertion of an indicator the knowledge base is guaranteed to be consistent. In this way, a knowledge base containing valid and usable information is always available for users.

6.2.3. Formula graph analysis. As introduced in Section 4, the set of formulas can be represented as a graph of dependencies, i.e. a lattice. This package contains a set of reasoning services capable of browsing this graph and reason on the links among indicators, in order to discover non-explicit relations. While `expand_indicator(ind)` is aimed to simply return the formula defined for a given indicator (if any), its generalized version `get_formulas(ind)` is capable of reasoning over the graph and compute all possible alternative formulas for ind that can be derived by manipulating the graph of dependencies. Please note that alternative rewritings of the same set of operands (e.g., permutations of the operands in a sum) are not returned as result. The service expands all indicators in their formulas and exploits mathematical services for formula manipulation. For instance, for `TotalEmissionsPerCapita`, it returns $\frac{TotalEmissions}{TotalPopulation}$ and $\frac{SOXEmissions+NOXEmissions+NonMethaneVolatileOrgEmissions+NH3Emissions}{TotalPopulation}$. As far as an indicator is included in a lattice and is not isolated in the graph (i.e., has a formula or is an operand for a formula), this service will return a non-empty set. Indeed, even if the indicator is atomic and a formula is not provided, like for `SOXEmissions`, the reasoner can calculate an answer by reverting other formulas, e.g. $SOXEmissions = TotalEmissions - NOXEmissions - NonMethaneVolatileOrgEmissions - NH3Emissions$ by applying `solve_equation` with respect to `SOXEmissions`.

Further services focus on recognition of dependencies, either for a single indicator, i.e. `get_dependencies(ind)`, or for couples of indicators, i.e. `get_common_dependencies(ind1, ind2)`. In this case the predicate executes, for each indicator, predicate `get_formula` in order to determine the list of all possible operands. Then, an intersection of the two results is performed.

6.3. Discussion

For lack of space we report here only the most relevant results of the evaluation of the functionalities described in this Section, that has been published in previous work of ours [31, 37] with regards to both efficiency and effectiveness. These tests have been carried out both on synthetic sets of indicators of various dimensions and complexity, and on a real-world set of indicators adopted for a European FP7 project* (for monitoring of indicators in Virtual Enterprises). Results show that execution times of the Prolog functions for consistency grow near quadratically in the number of indicators. Even with more than one thousand of PIs, the whole consistency check takes less than

*<http://www.bivee.eu>

3 seconds[†], while the `get_formulas` predicate can take up to 2 seconds for similar settings. We believe these execution times are in line with classic OLAP analysis.

For what concerns the expressiveness of the formulas taken into account in this work, the languages on which we rely cover a wide variety of real-world applications. In order for the logic-based functionalities to be executed, OpenMath formulas, represented in RDF, have to be translated into Prolog. As a consequence, we can manipulate the same class of expressions as the PRESS library. Unfortunately, even the authors of PRESS fail to precisely declare which is such a set of expressible formulas, that include, as stated above, symbolic, transcendental and non-differential equations. In this work we anyway refer to a subset of the possible PRESS expressions. Indeed, even in a real scenario[‡] the typologies of indicator formulas we deal with include, in most cases, only arithmetic operators. Similar examples can be found in popular libraries of indicators, such as `kpilibary.com`. Specific applications, however, may ask for more expressive formulas that currently can be represented only through complex workarounds. A limitation is related, for instance, to indicators that are defined over a database view, such as *PollutionInLast3Months*, which embeds a notion of temporal dimension (last three months, which can be defined only relatively to the query time).

7. CONCLUSION AND FUTURE WORK

In this work, we discussed an approach aimed at supporting communities of collaborating citizens in defining a customised model of indicators for analysis of statistical Open Government datasets published as Linked Data. New custom indicators can be defined by referring to the proposed vocabulary, specifying the mathematical formulas for their calculation. A set of services is enabled by the model, for a variety of analytical tasks, targeted to users with diverse background and abilities. Advantages of the approach are multi-fold: users can define custom indicators to mash up several Linked Data sources; formula definitions are reusable and shareable across multiple user communities and data sources; the approach can be easily adopted, as it does not require extensions of existing standards, relying only on the translation of OpenMath expressions in RDF; a novel data exploration and analysis approach is enabled, allowing to enhance awareness of communities of users about how to interpret statistical data. As such, this work is related also to the research area of Exploratory OLAP (or self-service BI) [38], which focuses on the use of semantic technologies to help the acquisition, integration and querying of disparate distributed data sources.

The possibility to cover more expressive mathematical expressions is currently under investigation, as well as novel functionalities capable of recognizing similarity, and not only equivalence or incoherence, among formulas. Also the development of specific tools implementing the described services and user-friendly interfaces will be addressed in future work. On the other side, the realisation of automated tools that can be successfully applied to a wide range of datasets is still yet to come. Several challenges have been indeed identified [39] and include, among others, different practices followed by publishers in applying the RDF Data Cube vocabulary, misuse of terms in the vocabulary and low quality datasets. Besides hampering the possibility of developing generic reusable tools to support the whole Linked Data lifecycle, this also makes hard the integration of multiple data sources, which still requires human effort for reconciliation of heterogeneities. All these issues, beyond and before a technological solution, are likely to ask for a cultural shift in the way users and institutions make use of public open data. The full potential of Linked Data will be unleashed if communities of users will keep pushing for public institutions to publish up-to-date data of good quality and following the best practices for publication and access. We believe that the approach proposed here is a contribution in this direction, by both enabling novel analytical capabilities of statistical Linked Data and enhancing user-awareness of the meaning behind indicator data.

[†]On an Intel Xeon CPU 3.60GHz with 3.50GB memory, running Windows Server 2003 SP 2.

[‡]For an example from BIVEE project, see <http://kdmg.dii.univpm.it/kpionto/examples/bivee>

A. EXAMPLES

A.1. RDF representation of Eurostat formulas

In the following we show the serialization in Turtle of the formulas introduced in Section 2 for the Eurostat case study. At first we define, as instances of `kpi:Indicator`, four indicators for the pollutants, namely `SOXEmissions`, `NOXEmissions`, `NonMethaneVolatileOrgEmissions` and `NH3Emissions`, and an indicator for `TotalPopulation`. Then, the indicator `TotalEmissions` is defined as the summation of the four contributions, i.e. `SOXEmissions+NOXEmissions+NonMethaneVolatileOrgEmissions+NH3Emissions`, with the following RDF representation. Please note that the symbol *plus* represents the summation as an n-ary commutative function in `OpenMath`, as defined in the “arith1” Content Dictionary.

```
@prefix :<http://kdmg.dii.univpm.it/kpionto/examples/eurostat_emissions/>
@prefix kpi:<http://w3id.org/kpionto/>.
@prefix om:<http://www.openmath.org/cd/arith1#>.
@prefix xsd:<http://www.w3.org/2001/XMLSchema#>.

:TotalEmissions a kpi:Indicator;
  kpi:hasFormula [
    a kpi:Formula;
    kpi:hasFunction om:plus;
    kpi:hasArgument
      [ a kpi:FormulaArgument;
        kpi:hasArgumentPosition "1"^^xsd:int ;
        kpi:hasArgumentValue :SOXEmissions ],
      [ a kpi:FormulaArgument;
        kpi:hasArgumentPosition "2"^^xsd:int ;
        kpi:hasArgumentValue :NOXEmissions ],
      [ a kpi:FormulaArgument;
        kpi:hasArgumentPosition "3"^^xsd:int ;
        kpi:hasArgumentValue :NonMethaneVolatileOrgEmissions ],
      [ a kpi:FormulaArgument;
        kpi:hasArgumentPosition "4"^^xsd:int ;
        kpi:hasArgumentValue :NH3Emissions ]
  ].
```

Please note that the needed namespaces are declared for reference in the definition of the formula. Finally, we define the `TotalEmissionsPerCapita` as the division of `TotalEmissions` by `TotalPopulation` as follows:

```
:TotalEmissionsPerCapita a kpi:Indicator;
  kpi:hasFormula [
    a kpi:Formula;
    kpi:hasFunction om:divide;
    kpi:hasArgument
      [ a kpi:FormulaArgument;
        kpi:hasArgumentPosition "1"^^xsd:int ;
        kpi:hasArgumentValue :TotalEmissions ],
      [ a kpi:FormulaArgument;
        kpi:hasArgumentPosition "2"^^xsd:int ;
        kpi:hasArgumentValue :TotalPopulation ]
  ].
```

The example discussed here has been serialized in Turtle and XML/RDF and is available at the project website [§] and on GitHub[¶].

[§]<http://kdmg.dii.univpm.it/kpionto>

[¶]<https://github.com/KDMG/kpionto>

A.2. Federated SPARQL query for calculation of TotalEmissionsPerCapita

```

PREFIX dataset: <http://eurostat.linked-statistics.org/data/>
PREFIX property: <http://eurostat.linked-statistics.org/property#>
PREFIX qb: <http://purl.org/linked-data/cube#>
PREFIX sdmx-measure: <http://purl.org/linked-data/sdmx/2009/measure#>
PREFIX sdmx-dimension: <http://purl.org/linked-data/sdmx/2009/dimension#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ((xsd:decimal(?SOXEmissions)+xsd:decimal(?NOXEmissions)
+xsd:decimal(?NonMethaneVolatileOrgEmissions)+
xsd:decimal(?NH3Emissions))/xsd:decimal(?TotalPopulation)
as ?TotalEmissionsPerCapita) ?year

FROM <http://eurostat.linked-statistics.org/data/tsdpc260.rdf>
FROM <http://eurostat.linked-statistics.org/data/tsdpc270.rdf>
FROM <http://eurostat.linked-statistics.org/data/tsdpc280.rdf>
FROM <http://eurostat.linked-statistics.org/data/tsdpc290.rdf>
FROM <http://eurostat.linked-statistics.org/data/tps00001.rdf>

WHERE {
  ?obs_tsdpc260 qb:dataSet dataset:tsdpc260;
  property:geo <http://eurostat.linked-statistics.org/dic/geo#IT>;
  property:airsect <http://eurostat.linked-statistics.org/dic/airsect#TOT_NAT>;
  sdmx-dimension:timePeriod ?year;
  sdmx-measure:obsValue ?SOXEmissions.

  ?obs_tsdpc270 qb:dataSet dataset:tsdpc270;
  property:geo <http://eurostat.linked-statistics.org/dic/geo#IT>;
  property:airsect <http://eurostat.linked-statistics.org/dic/airsect#TOT_NAT>;
  sdmx-dimension:timePeriod ?year;
  sdmx-measure:obsValue ?NOXEmissions.

  ?obs_tsdpc280 qb:dataSet dataset:tsdpc280;
  property:geo <http://eurostat.linked-statistics.org/dic/geo#IT>;
  property:airsect <http://eurostat.linked-statistics.org/dic/airsect#TOT_NAT>;
  sdmx-dimension:timePeriod ?year;
  sdmx-measure:obsValue ?NonMethaneVolatileOrgEmissions.

  ?obs_tsdpc290 qb:dataSet dataset:tsdpc290;
  property:geo <http://eurostat.linked-statistics.org/dic/geo#IT>;
  property:airsect <http://eurostat.linked-statistics.org/dic/airsect#TOT_NAT>;
  sdmx-dimension:timePeriod ?year;
  sdmx-measure:obsValue ?NH3Emissions.

  ?obs_tps00001 qb:dataSet dataset:tps00001;
  property:geo <http://eurostat.linked-statistics.org/dic/geo#IT>;
  sdmx-dimension:timePeriod ?year;
  sdmx-measure:obsValue ?TotalPopulation.
}

ORDER BY ?year

```

REFERENCES

1. Chun SA, Shulman S, Sandoval R, Hovy E. Government 2.0: Making connections between citizens, data and government. *Info. Pol.* Apr 2010; **15**(1,2):1–9.
2. Ding L, Lebo T, Erickson JS, DiFranzo D, Williams GT, Li X, Michaelis J, Graves A, Zheng JG, Shangguan Z, et al.. TWC LOGD: A portal for linked open government data ecosystems. *Web Semantics: Science, Services and Agents on the World Wide Web* 2011; **9**(3):325 – 333. Semantic Web Dynamics Semantic Web Challenge, 2010.
3. Ding L, Peristeras V, Hausenblas M. Linked open government data [guest editors' introduction]. *Intelligent Systems, IEEE* 2012; **27**(3):11–15.
4. Shadbolt N, O'Hara K, Berners-Lee T, Gibbins N, Glaser H, Hall W, Schraefel MC. Linked open government data: Lessons from data.gov.uk. *IEEE Intelligent Systems* 2012; **27**(3):16–24.
5. Cyganiak R, Reynolds D, Tennison J. The RDF data cube vocabulary. *Technical Report*, World Wide Web Consortium 2014.
6. Diamantini C, Genga L, Potena D, Storti E. Towards a customizable user-centered model for data analytics. *2015 International Conference on Collaboration Technologies and Systems, CTS 2015, Atlanta, GA, USA, June 1-5, 2015*, 2015; 249–256.

7. Kimball R, Ross M. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. 2nd edn., John Wiley & Sons: New York, NY, USA, 2002.
8. SDMX. SDMX technical specification. *Technical Report* 2013.
9. Hausenblas M, Halb W, Raimond Y, Feigenbaum L, Ayers D. Scovo: Using statistics on the web of data. *The Semantic Web: Research and Applications*. Springer, 2009; 708–722.
10. Etcheverry L, Vaisman A, Zimányi E. Modeling and querying data warehouses on the semantic web using QB4OLAP. *Data Warehousing and Knowledge Discovery, Lecture Notes in Computer Science*, vol. 8646, Bellatreche L, Mohania M (eds.). Springer International Publishing, 2014; 45–56.
11. Kämpgen B, Harth A. Transforming statistical linked data for use in OLAP systems. *Proceedings of the 7th International Conference on Semantic Systems, I-Semantics '11*, ACM: New York, NY, USA, 2011; 33–40.
12. Kämpgen B, O'Riain S, Harth A. Interacting with statistical linked data via OLAP operations. *The Semantic Web: ESWC 2012 Satellite Events*. Springer, 2012; 87–101.
13. Knuth DE, Bibby D. *The texbook*, vol. 1993. Addison-Wesley Reading, Mass., 1986.
14. Buswell S, Caprotti O, Carlisle DP, Dewar MC, Gaetano M, Kohlhasse M. The open math standard. *Technical Report*, version 2.0, The Open Math Society, 2004. <http://www.openmath.org/standard/om20> 2004.
15. Ausbrooks R, Buswell S, Carlisle D, Chavchanidze G, Dalmas S, Devitt S, Diaz A, Dooley S, Hunter R, Ion P, et al. Mathematical markup language (MathML) version 3.0. W3C candidate recommendation of 15 december 2009. *World Wide Web Consortium* 2009; **13**.
16. Lange C. Ontologies and languages for representing mathematical knowledge on the semantic web. *Semantic Web* 2013; **4**(2):119–158.
17. Marchiori M. The mathematical semantic web. *Mathematical Knowledge Management*, Springer, 2003; 216–223.
18. Ferr S. Representation of complex expressions in RDF. *Extended Semantic Web Conf. (ESWC Satellite Events)*, Cimiano P, Fernández M, Lopez V, Schlobach S, Völker J (eds.), LNCS 7955, Springer, 2013; 273–274.
19. Sánchez-Macián A, Pastor E, López Vergara JE, López D. Extending SWRL to enhance mathematical support. *Proceedings of Web Reasoning and Rule Systems: First International Conference, RR 2007, Innsbruck, Austria, June 7-8, 2007.*, Marchiori M, Pan JZ, Marie CdS (eds.), Springer Berlin Heidelberg: Berlin, Heidelberg, 2007; 358–360.
20. Robbins A. Semantic MathML. <http://straym mindcough.blogspot.com/2009/06/semantic-mathml.html> 2009.
21. Wenzel K, Reinhardt H. Mathematical computations for linked data applications with OpenMath. *24th OpenMath Workshop*, 2012.
22. Vrandečić D, Lange C, Hausenblas M, Bao J, Ding L. Semantics of governmental statistics data. *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line*, 2010.
23. Supply Chain Council. *Supply chain operations reference model*. SCC, 2008.
24. Popova V, Sharpanskykh A. Modeling organizational performance indicators. *Information Systems* 2010; **35**(4):505–527.
25. del Río-Ortega A, Resinas M, Cabanillas C, Ruiz-Cortés A. On the definition and design-time analysis of process performance indicators. *Information Systems* 2013; **38**(4):470–490.
26. Horkoff J, Barone D, Jiang L, Yu E, Amyot D, Borgida A, Mylopoulos J. Strategic business modeling: Representation and reasoning. *Softw. Syst. Model.* Jul 2014; **13**(3):1015–1041.
27. Neumayr B, Schütz C, Schrefl M. Semantic enrichment of OLAP cubes: Multi-dimensional ontologies and their representation in SQL and OWL. *On the Move to Meaningful Internet Systems: OTM 2013 Conferences, Lecture Notes in Computer Science*, vol. 8185. Springer Berlin Heidelberg, 2013; 624–641.
28. Huang SM, Chou TH, Seng JL. Data warehouse enhancement: A semantic cube model approach. *Information Sciences* June 2007; **177**(11):2238–2254.
29. Nebot V, Berlanga R. Building data warehouses with semantic web data. *Decision Support Systems* 2012; **52**(4):853–868.
30. Golfarelli M, Mandreoli F, Penzo W, Rizzi S, Turricchia E. OLAP Query Reformulation in Peer-to-peer Data Warehousing. *Information Systems* Jul 2012; **37**(5):393–411.
31. Diamantini C, Potena D, Storti E. SemPI: A semantic framework for the collaborative construction and maintenance of a shared dictionary of performance indicators. *Future Generation Computer Systems* 2015; **54**:352–365.
32. Diamantini C, Potena D, Storti E. Data mart reconciliation in virtual innovation factories. *Advanced Information Systems Engineering Workshops, Lecture Notes in Business Information Processing*, vol. 178, Iliadis L, Papazoglou M, Pohl K (eds.). Springer International Publishing, 2014; 274–285.
33. Daga E, d'Aquin M, Gangemi A, Motta E. Early analysis and debugging of linked open data cubes. *Second International Workshop on Semantic Statistics*, 2014. At the 13th International Semantic Web Conference (ISWC 2014).
34. Freundt S, Horn P, Kononov A, Linton S, Roozmond D. Symbolic computation software composability protocol (SCSCP) specification, version 1.3 2009.
35. Diamantini C, Potena D. Thinking structurally helps business intelligence design. *Information Technology and Innovation Trends in Organizations*, D'Atri A, al (eds.). Physica-Verlag HD, 2011; 109–116.
36. Sterling L, Bundy A, Byrd L, O'Keefe R, Silver B. Solving symbolic equations with PRESS. *J. Symb. Comput.* Jan 1989; **7**(1):71–84.
37. Diamantini C, Potena D, Storti E. Extended drill-down operator: Digging into the structure of performance indicators. *Concurrency and Computation: Practice and Experience* 2016; **28**(15):3948–3968.
38. Abello A, Romero O, Pedersen T, Berlanga R, Nebot V, Aramburu M, Simitsis A. Using semantic web technologies for exploratory OLAP: A survey. *Knowledge and Data Engineering, IEEE Transactions on* Feb 2015; **27**(2):571–588.
39. Kalampokis E, Roberts B, Karamanou A, Tambouris E, Tarabanis K. Challenges on developing tools for exploiting linked open data cubes. *Third International Workshop on Semantic Statistics*, 2015. At the 14th International Semantic Web Conference (ISWC 2015).