On Modeling Linked Open Statistical Data

Evangelos Kalampokis[a,b], Dimitris Zeginis[a,b,*], Konstantinos Tarabanis[a,b]

[a]*University of Macedonia, Information Systems Lab, Egnatia 156, Thessaloniki 54006, Greece*
[b]*Centre for Research & Technology – Hellas, Information Technologies Institute, 6th km Xarilaou - Thermi, Thessaloniki 57001, Greece.*

Abstract

A major part of Open Data concerns statistics such as economic and social indicators. Statistical data are structured in a multi-dimensional manner creating data cubes. Recently, National Statistical Institutes and public authorities adopted the Linked Data paradigm to publish their statistical data on the Web. Many vocabularies have been created to enable modeling data cubes as RDF graphs, and thus creating Linked Open Statistical Data (LOSD). However, the creation of LOSD remains a demanding task mainly because of modeling challenges related either to the conceptual definition of the cube, or to the way of modeling cubes as linked data. The aim of this paper is to identify and clarify (a) modeling challenges related to the creation of LOSD and (b) approaches to address them. Towards this end, nine LOSD experts were involved in an interactive feedback collection and consensus-building process that was based on Delphi method. We anticipate that the results of this paper will contribute towards the formulation of best practices for creating LOSD, and thus facilitate combining and analysing statistical data from diverse sources on the Web.

Keywords: Open Data, Linked Open Statistical Data, Modeling Challenges, Delphi Method

1. Introduction

International organizations, governments, and companies are increasingly opening up their data for others to reuse [1, 2, 3]. A major part of open data concerns statistics [4, 5, 6] such as demographics (e.g., census data), economic, and social indicators (e.g., number of new businesses, unemployment). Statistical data are organized in a multidimensional manner, and thus they can be conceptualized as data cubes, where a measured variable (e.g., unemployment) is described based on a number of dimensions (e.g., geography and time). These data can be an important primary material for added value services and products, which can increase government transparency, contribute to economic growth and provide social value to citizens [7, 8].

Linked data has been introduced as a promising paradigm for opening up data [9] because it facilitates data integration on the Web [10]. In statistics, linked data enable performing analytics on top of disparate and previously isolated datasets [11, 12, 13]. As a result, many National Statistical Institutes and public authorities (e.g., Scottish Government, Flemish Government, Italian National Institute of Statistics) have already used the linked data paradigm to publish statistical data on the Web [14, 15].

Many vocabularies have been created to enable modeling data cubes as RDF graphs. Examples include the RDF data cube (QB) vocabulary [16], the Simple Knowledge Organization System (SKOS) vocabulary [17], and the Extended Knowledge Organisation System (XKOS) vocabulary [18]. Some of them (e.g., QB and SKOS) have been also proposed as recommendations by the World Wide Web Consortium (W3C), and thus they are widely adopted. In addition, controlled vocabularies that define widely used statistical concepts have been created. Examples include the QUDT units vocabulary[1] and a linked data transformation of the Statistical Data and Metadata eXchange (SDMX) standard[2].

However, the creation of Linked Open Statistical Data (LOSD) remains a demanding task mainly because of modeling challenges related either to the conceptual definition of a cube, or to the way of modeling cubes as linked data. The former regards challenges such as the number of measures or the number of units to include in a cube, while the latter is related to (a) the lack of clarity on the way to apply the proposed vocabularies, e.g., how to model spatio-temporal information [19], and (b) the lack of specialized standards [20], e.g., there is no standard controlled vocabulary for the measures of a cube. All the above modeling challenges are currently addressed by data publishers in an ad hoc manner, and thus they hinder publishing LOSD in a uniform way that would facilitate their wide exploitation [21].

The aim of this paper is to identify and clarify modeling challenges related to the creation of LOSD and approaches to address them. Towards this end, nine LOSD experts were

*Corresponding author at: University of Macedonia, Information Systems Lab, Egnatia 156, Thessaloniki 54006, Greece

Email addresses: `ekal@uom.edu.gr` (Evangelos Kalampokis), `zeginis@uom.gr` (Dimitris Zeginis), `kat@uom.edu.gr` (Konstantinos Tarabanis)

[1]`http://qudt.org/`
[2]`https://github.com/UKGovLD/publishing-statistical-data`

involved in an interactive feedback collection and consensus-building process. The experts indicated and evaluated modeling challenges and approaches to address them, facilitating this way their understanding.

The method employed for the experts involvement was Delphi [22], which uses a questionnaire with multiple iterations to collect feedback until a stability in the responses is attained. The goal is to build consensus among experts on the approaches that can be adopted to address LOSD modeling challenges. The results presented in this paper can be used as a guidance for LOSD publishers. We also anticipate that our analysis will contribute towards the formulation of best practices for publishing LOSD, and thus it will facilitate combining and analyzing statistical data from diverse sources on the Web.

The rest of the paper is organized as follows: Section 2 presents the method that was followed, Section 3 presents the state of the art analysis regarding LOSD standards, Section 4 presents the results of the Delphi method including an analysis of the LOSD modeling challenges and the recommended approaches in which consensus was reached among the experts. Finally, Section 6 summarizes the results and discusses open challenges remaining to be explored.

2. Method

In order to identify and clarify LOSD modeling challenges and approaches to address them, we involved experts. The method that was employed is Delphi [22], which facilitates consensus-building [23] by using a questionnaire with multiple iterations to collect feedback until a stability in the responses is attained [24]. It also enables experts to re-assess an initial judgement based on the feedback provided by other experts.

One of the characteristics of Delphi is that although participants remain anonymous to each other, they are not anonymous to the researcher [25]. Their identity is not revealed, even after the completion of the final report. This prevents the domination of some participants (e.g., because of their personality or reputation), which is usually a concern when using group-based methods to collect information.

Delphi can be continuously iterated until consensus is achieved. However, literature has pointed out that two or three iterations are often enough to collect the needed information and reach sufficient consensus [26]. The two rounds of the presented study are the following:

Round 1: Traditionally the first round of the Delphi method begins with an open-ended questionnaire. However, a common modification uses a structured (aka closed) questionnaire that is based upon a preparatory phase. We adopted this modification to create a structured questionnaire. The preparatory phase contained: (i) analysis of the literature on the data cube model (Section 3.1) to identify the main modeling constructs related to statistical data, (ii) involvement of experts to identify the main LOSD modeling challenges, and (iii) analysis of LOSD standards (e.g., QB and SKOS vocabularies) to identify approaches related to the modeling challenges (Section 3.2).

The structured questionnaire of the first round asked experts to review, select or rank the initially identified approaches related to the modeling challenges. As a result, areas of disagreement/agreement were identified. The results of Round 1 included advantages/disadvantages of the publishing approaches as well as other publishing approaches not identified at the preparatory phase.

Round 2: The collected feedback of the first round was organized and a second questionnaire was created. This questionnaire was re-structured to be more comprehensive and incorporated the advantages/disadvantages identified at the first round to provide additional insights to the experts. It also contained approaches of the first round in which consensus was achieved so that experts can review them. In every question, the experts were asked to state the rationale behind their choice. The result of Round 2 included all the LOSD modeling challenges, an analysis of the approaches related to these challenges, and all the approaches where consensus was achieved.

The selection of appropriate experts is very important in a Delphi study since it affects the quality of the produced results. Usually, around ten experts are sufficient if their background is homogeneous. In our study, we included 9 experts in the area of LOSD:

- An expert involved in the creation of the LOSD portals of the Scottish Government[3] and the UK Department for Communities and Local Government (DCLG)[4].
- An expert involved in publishing of LOSD for the Flemish Government[5].
- An expert involved in the creation of the LOSD portal for the European Commission's Digital Agenda[6].
- An expert involved in the creation of the portal of the Italian National Institute of Statistics (ISTAT)[7].
- An expert involved in the creation of the QB vocabulary.
- An expert who created LOSD using data from international organizations such as Eurostat, OECD, IMF and World Bank.
- An expert working for the National Institute of Statistics and Economic Studies (INSEE).
- An expert working in academia.
- An expert working in industry.

The study took place in 2017 and comprised two Delphi rounds that lasted two months each. In order to facilitate the process we exploited Mesydel[8] an online service that supports Delphi enabling the participation of multiple experts around the globe.

[3]http://statistics.gov.scot
[4]http://opendatacommunities.org
[5]https://id.milieuinfo.be
[6]http://digital-agenda-data.eu/data
[7]http://datiopen.istat.it
[8]https://mesydel.com

3. Preparatory phase: State of the art analysis

3.1. Data Cube model

Statistical data usually concern aggregated data monitoring social and economic indicators (e.g., population size, inflation, trade, unemployment) [6]. Such kind of data can be described in a multidimensional way, where a measure is described based on a number of dimensions, e.g., unemployment rate can be described based on geography and time. Thus, statistical data can be conceptualized as a data cube [27]. Hence, we onwards refer to statistical data as "data cubes" or just "cubes".

The data cube model has already been defined in the literature [28, 29, 30, 31], and comprises a set of elements including measures (e.g., unemployment rate), which represent numerical values, and dimensions (e.g., geography and time), which provide contextual information for the measures. Each dimension comprises a set of distinct values (e.g., for the geographical dimension the values "Greece", "France", "Italy") that can be hierarchically organized into levels representing different granularities (e.g., the geographical dimension may have both countries and regions). The location of each cell of the cube is specified by the values of the dimensions, while the value of a cell specifies the measure (e.g., the unemployment rate of "Greece" in "2016" is "23.1%").

3.2. Standards for Linked Open Statistical Data

Linked data paradigm is based on semantic Web philosophy and technologies but it is mainly about publishing structured data in RDF using URIs rather than focusing on the ontological level or inferencing [32]. A good approach in linked data is to re-use standard vocabularies to encode data and meta-data [33]. In the following paragraphs we describe standard vocabularies for publishing LOSD.

The QB vocabulary [16] is a *W3C* standard for publishing statistical data on the Web using the linked data principles. The core class of the vocabulary is the *qb:DataSet* that represents a cube, which comprises a set of dimensions (*qb:DimensionProperty*), measures (*qb:MeasureProperty*), and attributes (*qb: AttributeProperty*) to represent structural meta-data such as the unit of measurement. The declaration of the dimensions, attributes, and measures is done at the *qb:DataStructureDefinition* (abbreviated as DSD) which defines the structure of the cube. Finally a *qb:DataSet* has multiple *qb:Observation* that describe the cells of the cube.

At linked data cubes it is a common practice to re-use predefined code lists to populate the dimension values. For example, the values of the time dimension can be obtained from the code list defined by `reference.data.gov.uk` or the values of the unit of measure can be obtained from the QUDT units vocabulary[9]. However, predefined code lists does not always exist, so new code lists should be specified using standards such as the QB vocabulary or the Simple Knowledge Organization System (SKOS) [17] vocabulary. The values of the code list may also have hierarchical relations which can be expressed using the SKOS vocabulary (e.g., *skos:narrower*), the QB vocabulary (e.g., *qb:parentChildProperty*) or the XKOS [18] vocabulary (e.g., *xkos:isPartOf*). XKOS is an unofficial extension of SKOS proposed by the DDI alliance, that enables the modeling of hierarchical structures in levels (*xkos:ClassificationLevel*).

Finally, a UK Government Linked Data Working Group[10] has developed a set of common concepts like dimensions, measures, attributes and code lists that are intended to be reusable across data sets. The definition of these concepts is based on the SDMX guidelines[11]. For example, dimensions like the *sdmx-dimension:timePeriod*, *sdmx-dimension:refArea*, *sdmx-dimension:sex*, *sdmx-dimension:age*, measures like *sdmx-measure:obsValue* and attributes like *sdmx-attribute:unitMeasure* have been proposed. These resources are not part of the QB vocabulary, however they are widely used by existing linked statistical data portals.

All the above standard vocabularies facilitate the publishing of LOSD. However, in some cases there is lack of clarity on how to apply these standards because they allow the adoption of different valid publishing approaches.

4. Delphi Results: Challenges and Approaches

This section presents the results of the Delphi method. It contains all the identified LOSD modeling challenges, an analysis (advantages/disadvantages) of the approaches related to these challenges and the approaches where consensus was achieved among the experts. We should also note that many challenges have not been addressed, and thus open challenges remain to be explored.

We use the following example throughout the Section to clarify the modeling challenges:
John works for the National Statistical Institute of Belgium. He is in charge of publishing last year's data about unemployment and poverty in the regions of Belgium. These data refer to various groups of people based on their age and gender. John decided to exploit linked data technologies in order to improve the quality and reusability of the data. The data at hand are of multi-dimensional nature, and thus they should be modeled as a cube. John needs to define the measures, units, dimensions and code lists. Some of the challenges that he faces include: (i) the definition of multiple measures (unemployment, poverty) per cube, (ii) the definition of multiple units (percentage, count) per measure, (iii) reuse standard vocabularies and code lists.

4.1. Defining a measure

Example: John needs to model the measures of the cube at hand as linked data. He wonders what is the best way to do so.

The cube measure represents the phenomenon being observed (e.g., unemployment). At the QB vocabulary measures are RDF properties of *qb:MeasureProperty* type, are defined at the cube structure (*qb:DataStructureDefinition*), and are used

[9]`http://qudt.org/`

[10]`https://github.com/UKGovLD/publishing-statistical-data`
[11]`https://sdmx.org/`

to assign numerical values to the observations (e.g., unemployment = 7.8%).

Challenge 1: What property should be used to model a measure of a cube? A common property for modeling a measure is *sdmx-measure:obsValue*, which is included in the linked data transformation of SDMX. This property is typically used in two ways:

- Re-using *sdmx-measure:obsValue* as a measure. In this case, however, the same measure property might be used to represent different measures. As a result, this approach has the following disadvantages: (i) it requires additional meta-data to indicate what is measured, (ii) it does not enable the use of multiple measures in a cube, and (iii) it prevents the linking with other cubes. However, *sdmx-measure:obsValue* facilitates the conversion of existing SDMX data to linked data using the QB vocabulary.
- Defining a new measure property that is sub-property of *sdmx-measure:obsValue*. In this case, two challenges may appear. The first appears when the new measure property is generic (e.g., count, ratio) and is used to represent different measures. The second challenge appears when two different (synonym) measure properties (e.g. test: unemployment and eg: unemployment) model the same measure (e.g., unemployment). Moreover, defining the new measure property as sub-property of *sdmx-measure:obsValue* is a redundancy because it does not add any additional semantics than defining the measure as a *qb:MeasureProperty*.

Based on the above analysis, experts came into consensus regarding the following approach:

Approach 1. *A new measure property should be defined that is not sub-property of sdmx-measure:obsValue. The new measure enables the annotation with additional properties (e.g., labels, comments).*

The proposed approach does not fully address the described challenges (e.g. generic measure properties). There are still open issues related with the definition of code lists for measures that are discussed in detail at Section 5.

4.2. Defining the unit

Example: John has already defined unemployment as the measure of his cube. Now he wonders (i) whether or not to include the unit of the measure in the cube, (ii) what RDF property to use to define the unit (iii) where to define the unit, and (iv) what values to assign.

Unit is the quantity or increment by which the measure is counted or described. The modeling of the unit raises four challenges:

Challenge 2.1: Should a cube include the unit of the measure? The measure on its own is a plain numerical value. To correctly interpret this value we need to define the unit. However, it is a common approach not to use units of measure.

Challenge 2.2: What RDF property should be used to define the unit? At the QB vocabulary, units can be modeled as RDF properties of *qb:AttributeProperty* type that are defined at the cube structure (*qb:DataStructureDefinition*). The selection of the unit property may lead to synonym challenges since different properties can be used for the same purpose. An approach to address this challenge is to re-use *sdmx-attribute:unitMeasure* property. In practice however, new properties are defined as sub-properties of *sdmx-attribute:unitMeasure* as shown in Listing 1.

```
:unit  a  qb: AttributeProperty ;
    rdfs :subPropertyOf  sdmx−attribute :unitMeasure.
:dsd  a  qb: DataStructureDefinition  ;
    qb:component [qb: attribute   : unit ].
```

Listing 1: Challenge 2.2 - Unit as sub-property of sdmx-attribute:unitMeasure

Challenge 2.3: Where should the unit be defined? The QB vocabulary enables the definition of units at different levels, i.e., the *qb:DataSet*, the *qb:MeasureProperty*, and the *qb:Observation*. By default, units are assigned to qb:Observation, however using the property *qb:componentAttachment* different levels can be also used. In particular, the identified approaches for addressing this challenge are:

- Defining the unit at the *qb:DataSet* (Listing 2). This approach facilitates the retrieval of the available units since units can be easily identified directly from the structure of the cube. However, if multiple units are defined at the same cube, then there is no way to map the unit to the values at the *qb:Observation*. Another disadvantage of this approach is that *qb:Observation* cannot be easily re-used at another context, because they do not contain all the relevant information, i.e., the unit is missing.

```
:unemployment a qb:MeasureProperty.
:dsd  a  qb:  DataStructureDefinition  ;
  qb:component [
      qb:  attribute   sdmx−attribute :unitMeasure;
      qb:componentAttachment qb:DataSet];
  qb:component [qb:measure  :unemployment].
:cube  a  qb:DataSet;
      qb: structure   :dsd;
      sdmx−attribute :unitMeasure  qudt: Percent .
```

Listing 2: Challenge 2.3 - Define the unit at qb:DataSet

- Defining the unit at the *qb:MeasureProperty* (Listing 3). This approach facilitates the retrieval of the available units since units can be identified directly from the structure. Moreover, multiple units can be assigned to a cube. However, different *qb:MeasureProperty* should be defined for different units, although they represent the same measure, e.g., two separate *qb:MeasureProperty* should be defined for unemployment, one assigned to percentage and another assigned to the absolute number. In this case, *qb:Observations* cannot easily be re-used at another context, because they do not contain all the relevant information, i.e., the unit is missing.

```
:unemploymentPerc a qb:MeasureProperty;
  sdmx-attribute:unitMeasure qudt:Percent.
:dsd a qb: DataStructureDefinition ;
  qb:component [
    qb: attribute  sdmx-attribute:unitMeasure;
    qb:componentAttachment qb:MeasureProperty];
  qb:component [qb:measure :unemploymentPerc].
```

Listing 3: Challenge 2.3 - Define the unit at qb:MeasureProperty

- Defining the unit at the *qb:Observation* (Listing 4). In this case, multiple units can be assigned to a cube and there is no need to define different *qb:MeasureProperty* for different units. Moreover, *qb:Observations* can be re-used at another context since they contain all relevant information. However, the retrieval of the available units is not efficient because it requires iterating over all *qb:Observation*.

```
:unemployment a qb:MeasureProperty.
:dsd a qb: DataStructureDefinition ;
  qb:component [qb: attribute  sdmx-attribute:unitMeasure];
  qb:component [qb:measure :unemployment].
:cube a qb:DataSet;
  qb: structure  :dsd.
:obs1 a qb:Observation;
  qb: dataset  :cube;
  unemployment: ‘‘16.8’’^^ xsd:decimal;
  sdmx-attribute:unitMeasure qudt:Percent.
```

Listing 4: Challenge 2.3 - Define the unit at qb:Observation

Table 1 presents a comparison between the three different levels to define the unit.

Table 1: Comparison matrix of the levels to define the unit

	Dataset	Measure	Observation
Query optimization	Yes	Yes	No
Multiple units at one cube	No	Yes	Yes
Use one qb:MeasureProperty with different units	Yes	No	Yes
Re-use qb:Observations at another context	No	No	Yes

Challenge 2.4: What values should be used? The values of the unit property (e.g., percentage) can be expressed using predefined URIs (e.g., `http://qudt.org/vocab/unit#Percent`). However, the use of different URIs to express the same unit leads to synonym challenges, e.g., `http://qudt.org/vocab/unit#Percent` and `http://statistics.gov.scot/def/concept/measure-units/percentage` define percentage. Three approaches have been identified to express the unit values:

- Use URIs from the QUDT units vocabulary
- Use URIs from DBpedia
- Define a new code list

Based on the above analysis, the experts came into consensus regarding the following approaches:

Approach 2.1. *A unit of measure should always be included in the cube. The measure on its own is a plain numerical value and thus unit is required to correctly interpret this value.*

Approach 2.2. *sdmx-attribute:unitMeasure should always be re-used to define units. This property can be used directly to assign values that are not part of a code list (e.g., QUDT). However, when annotation with additional properties (e.g., labels, code-list, etc.) is required, then new units that are sub-properties of sdmx-attribute:unitMeasure should be defined.*

Approach 2.3. *The unit should be defined at the qb:Observation. The unit can be additionally defined at the qb:DataSet in order to facilitate the retrieval of the available units in a cube.*

Approach 2.4. *URIs from QUDT should be re-used. If QUDT is not sufficient, then DBpedia or other code lists can be used.*

Fig.1 and Listing 5 present an example that covers Approaches 2.1 – 2.4.

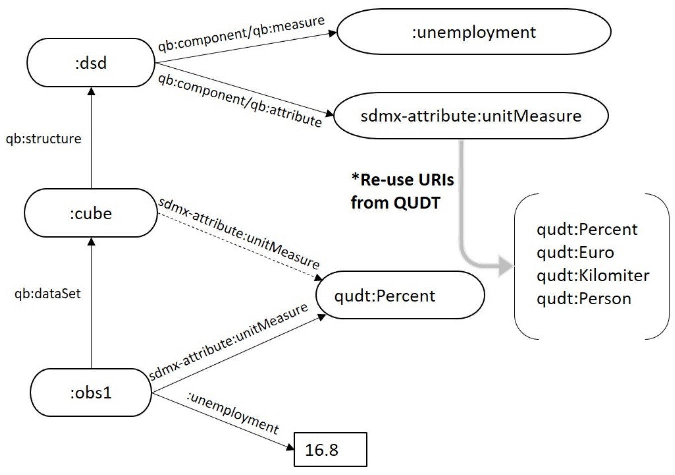

Figure 1: Approaches 2.1 – 2.4 - Defining the unit

```
:unemployment a qb:MeasureProperty.
:dsd a qb: DataStructureDefinition ;
  qb:component [qb: attribute  sdmx-attribute:unitMeasure];
  qb:component [qb:measure :unemployment].
:cube a qb:DataSet;
  qb: structure  :dsd;
  sdmx-attribute:unitMeasure qudt:Percent.
:obs1 a qb:Observation;
  qb: dataset  :cube;
  :unemployment ‘‘16.8’’^^xsd:decimal;
  sdmx-attribute:unitMeasure qudt:Percent.
```

Listing 5: Approaches 2.1 – 2.4 - Defining the unit

4.3. Defining multiple units per measure

Example: John realizes that the data he wants to publish contain unemployment as both rate, i.e., percentage of the labor force, and count, i.e., the actual number of unemployed people. As a result, he needs to include both units. Now he wonders (i) whether to include both units at the same cube or define separate cubes for each unit and (ii) where to define multiple units, e.g., at the structure or at the observation.

Challenge 3.1: Should one cube include multiple units for the same measure? It is common to have datasets that describe a measure (e.g., unemployment) using multiple units (e.g., rate and count). In this case, there are two modeling approaches that can be followed and are valid according to the QB voc.:

- Include all units at one cube
- Publish multiple cubes with one unit each

Challenge 3.2: Where to define multiple units? The selection of the approach to be followed is affected by the level where the unit is defined as described in Challenge 2.3. For example, if multiple units for the same measure are defined at the *qb:DataSet*, then there is no way to map the units to the values at the *qb:Observation*. Thus, the definition of multiple units at the *qb:DataSet* should be avoided.

Based on the above analysis the experts come into a consensus regarding the following approach:

Approach 3. *One cube with multiple units should be created and the unit should be defined at each qb:Observation (Fig. 2 and Listing 6). Conceptually, it is preferable to have all related units of the same measure in the same cube. The unit can be additionally defined at the qb:DataSet in order to facilitate the retrieval of the available units in a cube.*

```
:unemployment a qb:MeasureProperty.
:dsd a qb: DataStructureDefinition ;
  qb:component [qb: attribute  sdmx−attribute:unitMeasure];
  qb:component [qb:measure :unemployment].
:cube a qb:DataSet;
  qb: structure  :dsd.
:obs1 a qb:Observation;
  qb: dataset  :cube;
  :unemployment ‘‘16.8’’^^ xsd:decimal;
  sdmx−attribute:unitMeasure qudt:Percent.
:obs2 a qb:Observation;
  qb: dataset  :cube;
  unemployment ‘‘89600’’^^ xsd: int ;
  sdmx−attribute:unitMeasure qudt:Person.
```

Listing 6: Approach 3 - Defining multiple units per measure

4.4. Defining multiple measures

Example: John wants to publish also data about poverty in Belgium. John wonders whether to publish the data about unemployment and poverty in the same or separate cubes. In case both measures are included in the same cube, he also wonders what is the best way to do so, considering that the measures have multiple units (count and rate).

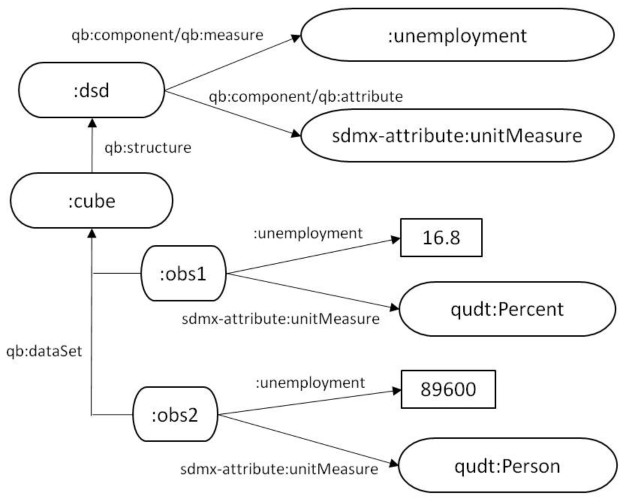

Figure 2: Approach 3 - Defining multiple units per measure

It is common to have datasets comprising multiple measures (e.g., unemployment and poverty). In this case, there are two modeling approaches that can be followed and are valid according to the QB vocabulary:

- Publish multiple cubes with one measure each
- Include all measures in one cube

The first approach is covered at Section 4.2 (measure with one unit) and at Section 4.3 (measure with multiple units).

Challenge 4: How to model multiple measures per cube? Considering the second approach (multiple measures at one cube), the QB vocabulary proposes two approaches to represent multiple measures:

- Multi-measure observations (Listing 7): Define multiple *qb:MeasureProperty* in the data structure definition and use all measures in every *qb:Observation*. Defining all the measures in one observation reduces the size of the produced cube. However, a limitation is that an attribute property (e.g., unit) cannot be associated to a single measurement. An attribute property attached to the observation will apply to all measurements, thus it cannot represent multiple measures with multiple units.

```
:unemployment a qb:MeasureProperty.
:poverty a qb:MeasureProperty.
:dsd a qb: DataStructureDefinition ;
  qb:component [qb:measure :unempolyment];
  qb:component [qb:measure :poverty ].
:cube a qb:DataSet;
  qb: structure  :dsd.
:obs1 a qb:Observation;
  qb: dataset  :cube;
  :unemployment ‘‘89600’’^^xsd: int ;
  :poverty  ‘‘153000’’^^ xsd: int .
```

Listing 7: Multi-measure observations

- Measure dimension (Listing 8): Define multiple *qb:MeasureProperty* at the data structure definition, but restrict observations to having a single measure. This is achieved by defining an extra dimension, the *qb:measureType*, to denote which particular *qb:MeasureProperty* is included at the *qb:Observation*. The use of a single measure at observations produces a large number of observations, thus increasing the size of the produced cube. However, this approach enables the definition of multiple units and multiple measures.

```
:unemployment a qb:MeasureProperty.
:poverty  a  qb:MeasureProperty.
:dsd  a  qb: DataStructureDefinition  ;
   qb:component [qb:measure :unempolyment];
   qb:component [qb:measure : poverty ];
   qb:component [qb:dimension qb:measureType].
:cube  a  qb:DataSet;
   qb: structure   :dsd.
:obs1  a  qb:Observation;
   qb: dataset   :cube;
   :unemployment ‘‘89600’’^^xsd: int ;
   qb:measureType :unemployment.
:obs2  a  qb:Observation;
   qb: dataset   :cube;
   :poverty   ‘‘153000’’^^ xsd: int ;
   qb:measureProperty  : poverty .
```

Listing 8: Measure dimension

Table 2 presents a comparison of the two approaches to define multiple measures.

Table 2: Comparison matrix of approaches to define multiple measures

	Multi-measure observations	Measure dimension
Size of produced cube	Small	Large
Support of multiple units and measures	No	Yes
Support of multiple measures with the same unit	Yes	Yes

Approach 4.1. *If the data have multiple measures, then it is common to publish cubes with multiple measures only when measures are closely related to a single observational event (e.g. sensor network measurements). However, the approach to be followed is up to the data cube publisher. In case of modeling multiple measures in multiple cubes with one measure each, then Approach 2 (if the measures have one unit) and Approach 3 (if the measures have multiple units) should be followed.*

Approach 4.2. *In case of modeling multiple measures in one cube then the measure dimension approach (i.e. observations with a single measure) should be followed and the unit should be defined in each observation (as already explained in Approach 3) (Fig. 3 and Listing 9).*

There are still open issues related to the above approaches since there is not clear definition of an observational event. These open issues are discussed in detail at section 5.

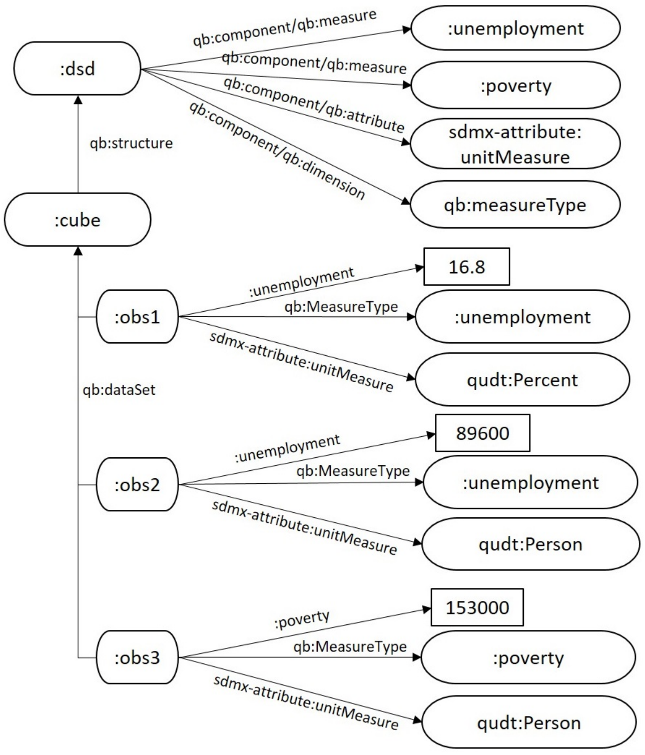

Figure 3: Approach 4.2 - Defining multiple measures in one cube

```
:unemployment a qb:MeasureProperty.
:poverty  a  qb:MeasureProperty.
:dsd  a  qb: DataStructureDefinition  ;
   qb:component [qb:measure :unemployment];
   qb:component [qb:measure : poverty ];
   qb:component [qb:dimension qb:measureType];
   qb:component [qb: attribute   sdmx−attribute :unitMeasure ].
:cube  a  qb:DataSet;
   qb: structure   :dsd.
:obs1  a  qb:Observation;
   qb: dataset  :cube.
   :unemployment  ‘‘16.8’’^^ sxsd:decimal;
   qb:measureType :unemployment;
   sdmx−attribute :unitMeasure  qudt: Percent.
:obs2  a  qb:Observation;
   qb: dataset  :cube;
   :unemployment  ‘‘89600’’^^ xsd: int ;
   qb:measureType :unemployment;
   sdmx−attribute :unitMeasure  qudt:Person.
:obs3  a  qb:Observation;
   qb: dataset  :cube;
   :poverty    ‘‘153000’’^^ xsd: int ;
   qb:measureType:poverty;
   sdmx−attribute :unitMeasure  qudt:Person.
```

Listing 9: Approach 4.2 - Defining multiple measures in one cube

4.5. Defining dimension properties

Example: John has already defined the measures and the units of his cube. Now he needs to define the dimensions including *time, geography, age and gender. He wonders what RDF properties to use for these dimensions.*

Dimensions (e.g., geography and time) provide contextual information for the measures of the cube. In the QB vocabulary, dimensions are RDF properties of *qb:DimensionProperty* type and are defined in the cube structure (*qb:DataStructureDefinition*).

Challenge 5: What *rdf:Properties* should be (re-)used for common dimensions? The time, geography, age, and gender dimensions are common in statistical data. A linked data conversion of SDMX has already defined *qb: DimensionProperty* to express them, i.e., *sdmx-dimension:refPeriod*, *sdmx-dimension: refArea*, *sdmx-dimension:age*, and *sdmx-dimension:sex*. The re-use of these properties could alleviate synonym challenges, which appear when multiple properties are used for the same dimension.

These SDMX dimensions, however, are not associated to controlled vocabularies that define dimension values. An exception is *sdmx-dimension:sex*, which is associate with a defined code list.

In practice, however, new dimensions are defined as subproperties of the SDMX dimensions. An advantage of this approach is the fact that it enables the addition of extra annotation to the dimensions (e.g., *rdfs:label*, code lists, *rdfs:range*).

Based on the above analysis, the experts came into a consensus regarding the following approaches (Fig.4):

Approach 5.1. *If a dimension refers to time, geography, or age, then a new qb:DimensionProperty should be defined. This new qb:DimensionProperty should be also defined as rdfs:subPropertyOf the corresponding SDMX dimension. For example, a geospatial dimension of a cube should be defined as sub-property of sdmx-dimension:refArea (Listing 10). Sections 4.6 and 4.7 describe the way the values of a new dimension can be defined.*

```
:geo a qb:DimensionProperty;
  rdfs :subPropertyOf  sdmx−dimension:refArea.
:dsd a qb: DataStructureDefinition ;
  qb:component [qb:dimension :geo].
```

Listing 10: Approach 5.1 - Defining a dimension that is sub-property of the corresponding SDMX dimension

Approach 5.2. *If a dimension refers to gender, then sdmx-dimension:sex should be reused provided that the associated code list addresses the modeling needs, e.g., more notions of sex such as hermaphroditism, transgender, and asexual are not needed (Listing 11). Otherwise, a new dimension should be defined along with a controlled vocabulary (Sections 4.6 and 4.7).*

```
:dsd a qb: DataStructureDefinition ;
  qb:component [qb:dimension sdmx−dimension:sex].
```

Listing 11: Approach 5.2 - Re-using the sdmx-dimension:sex

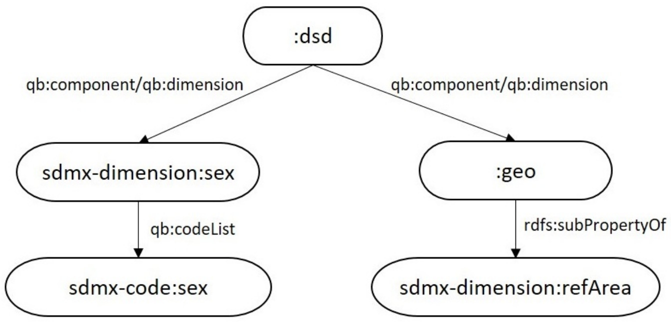

Figure 4: Approaches 5.1 and 5.2 - Defining common dimension properties

4.6. Associating dimensions with their values

Example: John needs to associate dimensions with their potential values. He wonders what is the best way to do so.

Dimension values can be either data types (e.g., *xsd:dateTime*) or URIs. In the latter case, URIs can be grouped in either code lists that are modeled as *skos:ConceptScheme*, *skos:Collection*, or *qb:HierarchicalCodeList* or as reference datasets (e.g., `http://reference.data.gov.uk`) that are not modelled in one of the above ways.

Challenge 6: How to associate a dimension to its values? The QB vocabulary allows two complementary approaches for associating dimensions with their potential values:

- Using the property *rdfs:range* to define the class of the values of a *qb:DimensionProperty* following the typical RDF practice. For example, the values of the temporal dimension of a cube can be defined by setting the *rdfs:range* of the dimension to *xsd:dateTime*.
- Using the property *qb:codeList* to associate a *qb:DimensionProperty* with a code list (i.e., potential values). In statistical datasets it is common for values to be encoded using a code list and it is useful to easily identify the overall code list with a URI. The object of the *qb:codeList* property can be a *skos:ConceptScheme*, *skos:Collection* or *qb:HierarchicalCodeList*. In such case the *rdfs:range* can be a *skos:Concept*. According to the QB vocabulary a useful design pattern is to define an *rdfs:Class* whose members are all the *skos:Concepts* within a particular code list. In this way the *rdfs:range* can be made more specific.

Based on the above analysis the experts came into a consensus regarding the following approaches (Fig.5):

Approach 6.1. *The rdfs:range of a qb:DimensionProperty should always be defined (Listing 12).*

```
:d a qb:DimensionProperty
       rdfs :range  xsd:dateTime.
:dsd a qb: DataStructureDefinition ;
       qb:component [qb:dimension :d].
```

Listing 12: Approach 6.1 - Associating a dimension with its values using rdfs:range

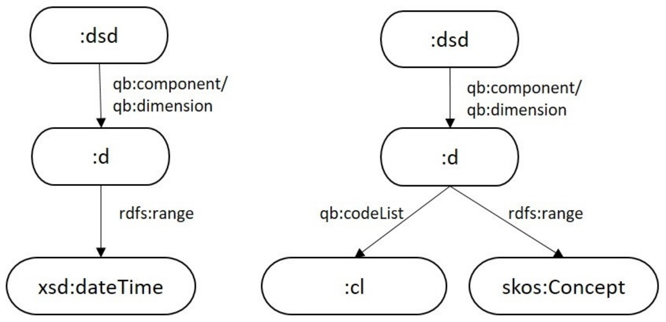

Figure 5: Approaches 6.1 and 6.2 - Associating dimensions with their values using (a) rdfs:range and (b) both rdfs:range and qb:codeList

Approach 6.2. *If a code list is modelled as skos:ConceptScheme, qb:HierarchicalCodeList, or skos:Collection, then it should be associated with the qb:DimensionProperty using the qb:codeList property. In addition, the object that is related to the rdfs:range property should be set to skos:Concept (Listing 13). Section 4.9 describes how a new code list can be created.*

```
:d a qb:DimensionProperty;
        rdfs :range  skos:Concept;
        qb: codeList  : cl .
:dsd a qb: DataStructureDefinition ;
        qb:component [qb:dimension :d ].
```

Listing 13: Approach 6.2 - Associating a dimension with its values using both rdfs:range and qb:codeList

4.7. Defining values of common dimensions

Example: John now knows how to associate his cube's dimensions with their values. However, he wonders (i) whether to use data types or URIs and (ii) in case of URIs, what code lists to use to define values of common dimensions including time, geography, age, and gender.

The definition of values of these common dimensions raises three challenges:

Challenge 7.1: What values should be used in time related dimensions? The values of time dimension can be either periods of time (e.g., 2016) or specific points in time (e.g., 01/01/2016), and thus they can be represented either as URIs (e.g., `http://reference.data.gov.uk/id/year/2016`) or as data types (e.g., “2016-01-01”^^xsd:date). The selection of the approach to represent the time values may lead to synonym challenges since different values can be used for the same purpose. Four approaches are commonly used to express time values:

- Using URIs from `http://reference.data.gov.uk`, which, however, is not defined as a *skos:ConceptScheme*
- Using resources from DBpedia, which are not included in a *skos:ConceptScheme*
- Defining a new code list

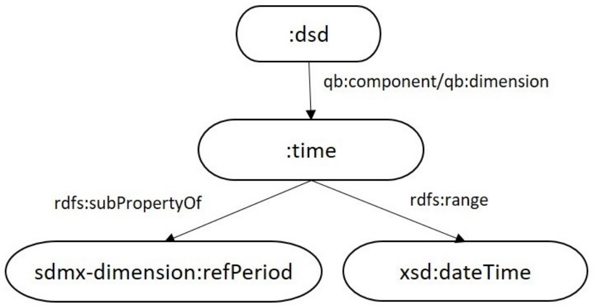

Figure 6: Approach 7.1a - Describing a specific point in time

- Using XSD date and time data types

Challenge 7.2: What values should be used in geospatial dimensions? The values of geospatial dimensions are usually represented as URIs. Although some official location-specific code lists have been defined in specific countries, regions, etc., the selection of URIs to represent these values may lead to synonym challenges since different values can be used for the same purpose.

Challenge 7.3: What values should be used in age related dimensions? Currently, there are no standardized code lists that can be used for these dimensions. As a result, synonym challenges may also appear.

Challenge 7.4: What values should be used in gender related dimensions? The values of the gender dimension are usually represented as URIs. The linked data version of SDMX has already defined a code list for gender including the values *sdmx-code:sex-F* (female), *sdmx-code:sex-M* (male), *sdmx-code:sex-U* (undefined), *sdmx-code:sex-N* (not applicable) and *sdmx-code:sex-T* (total). In some cases however, the SDMX code list cannot cover the modeling needs, e.g., more notions of sex are needed like hermaphroditism, transgender, asexual. This challenge is addressed by Approach 5.2.

Based on the above analysis the experts came into a consensus regarding the following approaches:

Approach 7.1a. *In case of a specific point in time a new dimension should be defined. This dimension should be rdfs:subPropertyOf sdmx-dimension:refPeriod and have rdfs:range xsd:dateTime (Fig.6 and Listing 14).*

```
:time a qb:DimensionProperty;
        rdfs :subPropertyOf  sdmx−dimension:refPeriod;
        rdfs :range  xsd:dateTime.
:dsd a qb: DataStructureDefinition ;
        qb:component [qb:dimension :time ].
```

Listing 14: Approach 7.1a - Describing a specific point in time

Approach 7.1b. *In case of a period of time, a new dimension should be defined. This dimension should be rdfs:subPropertyOf sdmx-dimension:refPeriod and have rdfs:range the interval:Interval class of the* `http://reference.data.gov.uk`*, which uses this class to define*

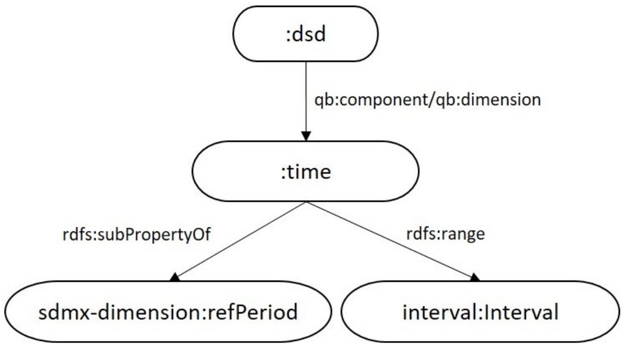

Figure 7: Approach 7.1b - Describing a period of time

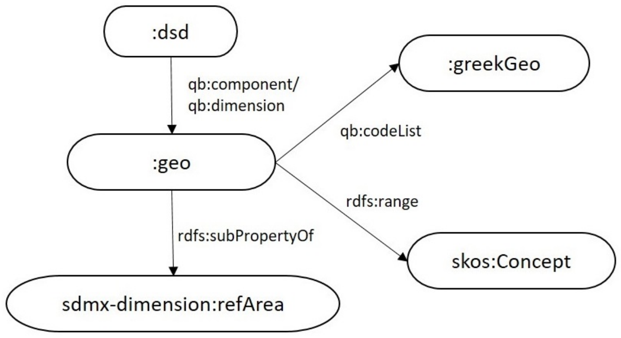

Figure 8: Approach 7.2 - Defining values of geography and age

years (Fig.7 and Listing 15). However, if the approach of `http://reference.data.gov.uk` *is not sufficient, then new code lists can be also created and used (Section 4.9).*

```
:time a qb:DimensionProperty;
        rdfs:subPropertyOf  sdmx−dimension:refPeriod;
        rdfs:range  interval:Interval .
:dsd a qb: DataStructureDefinition ;
        qb:component [qb:dimension  :time ].
```

Listing 15: Approach 7.1b - Describing a period of time

Approach 7.2. *In case of a geography or age related dimension, a new dimension should be defined. This dimension should be rdfs:subPropertyOf the sdmx-dimension:refArea or sdmx-dimension:age respectively (Fig.8 and Listing 16). Moreover, the rdfs:range and/or qb:codeList of this dimension should be defined as described in Section 4.6. If a code list or reference dataset that addresses the modeling needs exists, then it should be re-used. Otherwise, a new code list should be created (see Section 4.9).*

```
:geo a qb:DimensionProperty;
      rdfs:subPropertyOf  sdmx−dimension:refArea;
      qb:codeList  :BEgeo;
      rdfs:range  skos:Concept.
:dsd a qb: DataStructureDefinition ;
      qb:component [qb:dimension  :geo]
```

Listing 16: Approach 7.2 - Defining values of geography and age

4.8. Modeling single value dimensions

Example: The cube at hand includes data only for one year, i.e., 2016. John wonders (i) whether or not to include this single value in the data cube and (ii) if so, what modeling approach to follow.

Some datasets describe a measure using only a single value of a dimension. For example, census data describe multiple measures for a specific year.

Challenge 8: How to model single value dimensions? The QB vocabulary enables defining this single value at different levels:

- Including the single value at the *qb:Dataset*. In this case, the single value is easily identified directly from the structure. However, it does not enable future addition of observations with a different value for that particular dimension. Another disadvantage of this approach is that *qb:Observations* cannot easily be re-used at another context, because they do not contain all the relevant information, i.e., the dimension with the single values is missing.
- Creating a *qb:Slice* for the single value. In this case, the single value is easily identified from the structure. Observations with a different dimension value can be latter added by creating a new *qb:Slice* for this value. A disadvantage of this approach is that it imposes the extra burden of defining *qb:Slices* when creating the cube. Moreover, *qb:Observations* cannot easily be re-used at another context, because they do not contain all the relevant information, i.e., the dimension with the single values is missing.
- Including the single value at every *qb:Observation*. This approach enables simple and flexible data maintenance. In particular, it enables the addition of more observations with different dimension values in the same dataset and the easy re-use of qb:Observations at another context. This approach has, however, the cost of an increased number of triples.

```
:obs1 a qb:Observation;
        sdmx−dimension:sex sdmx−code:sex−M;
        :age :15−24;
        :geo :Brussels;
        :time year:2017;
        sdmx−attribute:unitMeasure qudt:Percent;
        :unemployment ‘‘35.1’’^^xsd:decimal.
:obs2 a qb:Observation;
        sdmx−dimension:sex sdmx−code:sex−F;
        :age :15−24;
        :geo :Brussels;
        :time year:2017;
        sdmx−attribute:unitMeasure qudt:Percent;
        :unemployment ‘‘36.7’’^^xsd:decimal.
```

Listing 17: Approach 8 - Modeling the single value of a dimension at each qb:Observation

Based on the above analysis the experts come into a consensus regarding the following approach:

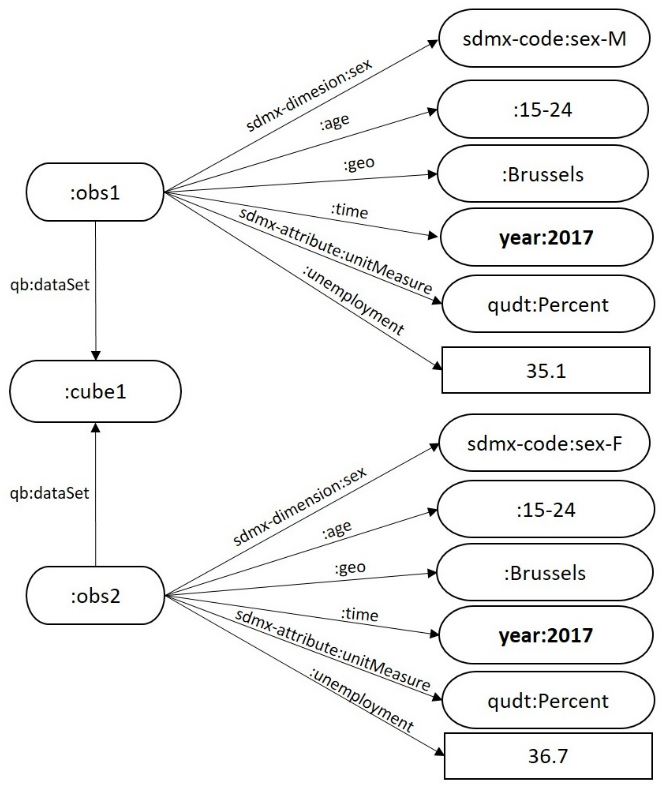

Figure 9: Approach 8 - Modeling the single value of a dimension at each qb:Observation

Approach 8. *A single value dimension should be always included in all observations of the cube (Fig.9 and Listing 17).*

4.9. Creating code lists

Example: John has already defined dimension properties and decided the code lists to use for two dimensions, namely time and gender. However, there are no appropriate code lists for age and geography related dimensions, and thus John has to create them.

The creation of code lists raises three challenges:

Challenge 9.1: How to model a new code list? Existing code lists do not always address the modeling needs of a specific case. As a result, new code lists should be created. The QB vocabulary recommends using SKOS to model a code list. In particular, it recommends representing the individual code values using *skos:Concept* and the overall set of values using *skos:ConceptScheme* or *skos:Collection*.

Challenge 9.2: How to model hierarchical structures in a code list? Typically, cubes include data with hierarchical structures (e.g., geographical or administrative divisions). In practice, however, these structures are not always explicitly defined in code lists. Hierarchical structures comprise generalization/specialization relations (e.g., Brussels *is part of* Belgium) and hierarchical levels (e.g., country, region, city). Various properties can be used for modeling these relations and levels. The following approaches can be used to represent hierarchical code lists:

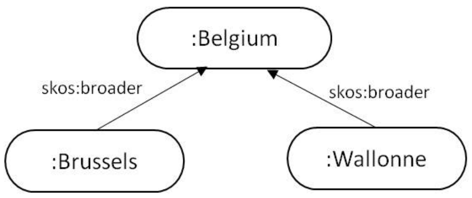

Figure 10: Using SKOS vocabulary to model hierarchical code lists

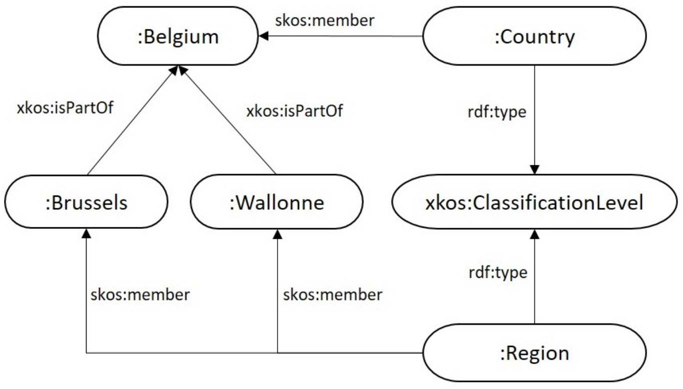

Figure 11: Using XKOS vocabulary to model hierarchical code lists

- Using SKOS vocabulary. This approach defines only generalization/specialization relations, i.e., *skos:broader* and *skos:narrower* between *skos:Concepts* (Fig.10 and Listing 18).

```
:Belgium  a  skos:Concept.
: Brussels  a  skos:Concept;
               skos:broader  :Belgium.
:Wallonne a  skos:Concept;
               skos:broader  :Belgium.
```

Listing 18: Using SKOS vocabulary to model hierarchical code lists

- Using XKOS vocabulary. XKOS is an extension of SKOS that enables modeling both generalization/specialization relations (i.e., *xkos:isPartOf*, *xkos:hasPart*) and hierarchical levels (i.e., *xkos: ClassificationLevel*) of *skos: Concepts* (Fig.11 and Listing 19).

```
:Belgium  a  skos:Concept.
: Brussels  a  skos:Concept;
               xkos: isPartOf  :Belgium.
:Wallonne a  skos:Concept;
               xkos: isPartOf  :Belgium.
:Country   a  xkos: ClassificationLevel ;
               skos:member :Belgium.
:Region    a  xkos: ClassificationLevel ;
               skos:member :Brussels;
               skos:member :Wallonne.
```

Listing 19: Using XKOS vocabulary to model hierarchical code lists

- Using the QB vocabulary. In this case, a *qb:HierarchicalCodeList* should be defined to

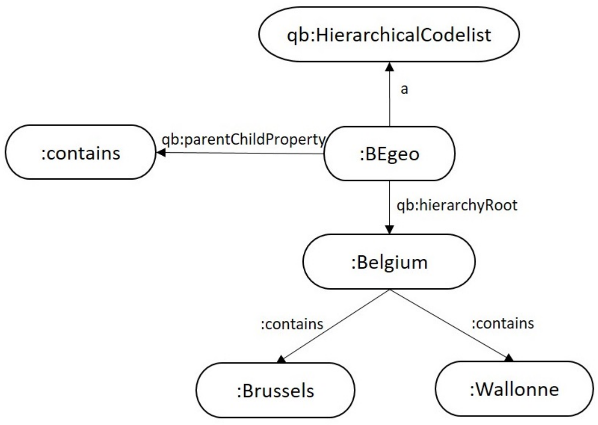

Figure 12: Using the QB vocabulary to model hierarchical code lists

represent the overall set of values (similar to *skos:ConceptScheme*). The *qb:hierarchyRoot* serves the same purpose as *skos:hasTopConcept*, and the value of *qb:parentChildProperty* serves the same purpose as *skos:narrower* (Fig.12 and Listing 20). This approach is provided for cases where the terms are not available as SKOS but are available in some other RDF representation suitable for reuse.

- Sometimes, case specific properties are exploited to express hierarchies. This approach enables the definition of generalization/specialization relations that are not covered by SKOS and XKOS (e.g., administered by and within). However, without using any standard vocabulary the interpretation of the data is not always feasible.

```
:BEgeo a qb: HierarchicalCodelist ;
                qb: parentChildProperty   : contains ;
                qb:hierarchyRoot  :Belgium.
:Belgium a  skos:Concept;
               : contains   : Brussels ;
               : contains   :Wallonne.
: Brussels  a  skos:Concept.
:Wallonne a  skos:Concept.
```

Listing 20: Using the QB vocabulary to model hierarchical code lists

Challenge 9.3: Should aggregate (e.g., "Total") values be included as dimension values? A common practice suggests that aggregate values should be included in a dimension (e.g., male, female and total). These values that are pre-computed using various aggregate functions facilitate performing complex operations (e.g., OLAP operations) on top of data cubes. In practice, however, aggregate values cannot be differentiated in a code list. For example, the linked data version of SDMX contains the values *sdmx-code:sex-F*, *sdmx-code:sex-M*, and *sdmx-code:sex-T* (i.e., Total) without any extra semantics that would facilitate distinguishing the last one as an aggregate value. It is important these values to be differentiated from other values, otherwise there is a risk of inaccurate interpretation of the data (e.g., duplicate the values).

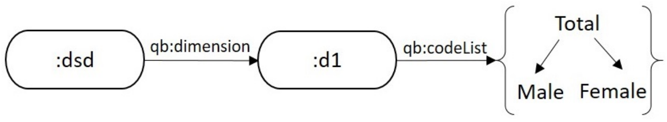

Figure 13: Define aggregate values at the top of the hierarchy

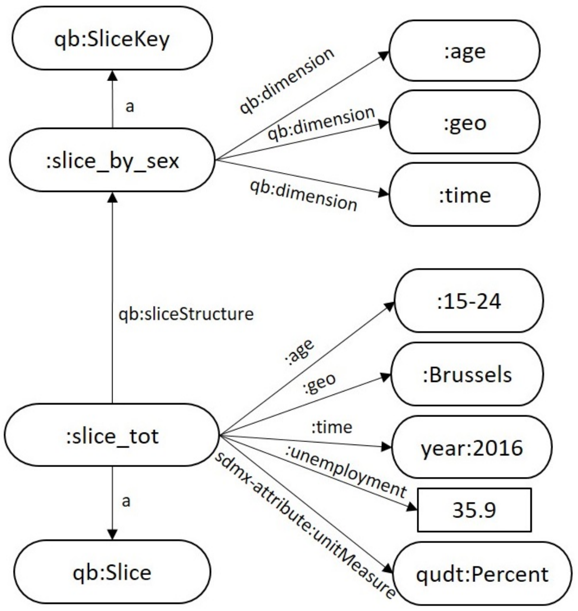

Figure 14: Defining aggregate values on slices

The following approaches have been proposed to differentiate aggregate values.

- Use a hierarchy and define aggregate values on top of the hierarchy (Fig.13). The approach to express the hierarchy is based on Challenge 9.2.
- Define aggregate values on slices. The aggregate value of a dimension can be also represented by excluding the specific dimension from the slice. In Fig.14 and Listing 21 a slice that defines the "Total" unemployment (35.9%) for both genders (i.e., male and female) with age 15-24, in Brussels in 2016 is presented.

```
:age  a  qb:DimensionProperty.
:geo  a  qb:DimensionProperty.
:time  a  qb:DimensionProperty.
:  slice_by_sex   a  qb:SliceKey
         qb:dimension  :age;
         qb:dimension  :geo;
         qb:dimension  :time.
:  slice_tot    a  qb: Slice ;
         qb:  sliceStructure    :  slice_by_sex  ;
          :age  :15−24;
          :geo  : Brussels ;
          :time  year:2016;
          :unemployment  ''35.9''^^ xsd:decimal;
         sdmx−attribute :unitMeasure  qudt: Percent .
```

Listing 21: Defining aggregate values on slices

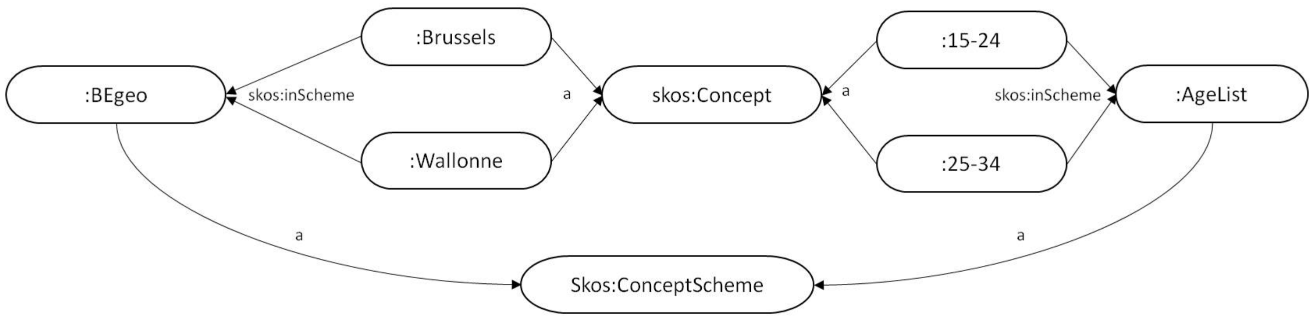

Figure 15: Approach 9.1 - Use SKOS to model code lists

Based on the above analysis, the experts came into consensus regarding the following approach:

Approach 9.1. *A code list should be modelled using SKOS. This is also suggested by the QB vocabulary. Specifically, individual code values should be modelled using skos:Concept and the overall set of values should be modelled using skos:ConceptScheme or skos:Collection (Fig.15 and Listing 22). Always define a separate code list for each distinct set of values (e.g., age groups and geographical areas).*

Approach 9.2. *In case of hierarchical data, hierarchical code lists should be always used to describe them. SKOS should be preferred when the hierarchies are simple. In case where the hierarchical levels are fully separated and depth is a meaningful concept then XKOS is appropriate. Finally, when there is a need to express more relations that are not covered by SKOS or XKOS (e.g., administeredBy in contrast to within) then the QB vocabulary should be preferred.*

Approach 9.3. *Aggregate values (e.g., "Total") should be included in a dimension if the measured variable in this dimension can be aggregated. The aggregate value should be modelled on the top a hierarchy (Approach 9.2).*

```
:15−24     a  skos:Concept.
:25−34     a  skos:Concept.
:Brussels  a  skos:Concept.
:Wallone   a  skos:Concept.
:AgeList   a  skos:ConceptScheme;
              skos:hasTopConcept :15−24;
              skos:hasTopConcept :25−34.
:BEgeo     a  skos:ConceptScheme;
              skos:hasTopConcept :Brussels ;
              skos:hasTopConcept :Wallone.
```

Listing 22: Approach 9.1 - Using SKOS to model code listss

5. Open challenges

During the Delphi process experts and practitioners indicated a number of open challenges. These open challenges regard, limitations of existing standards, lack of standards and modeling decisions. In the following paragraphs we present the identified open challenges.

An important challenge is the definition of code lists for measures that could be re-used by LOSD publishers. Currently each LOSD publisher defines its own *qb:MeasureProperty* for the same measure (e.g. *p1:unemployment* and *p2:unemployment*). The definition of a "standard" code list would enable the publishing of LOSD in a uniform way, thus facilitating the integration and combination of related statistical data from different sources.

Another challenge is related with the method (e.g. formula) a measure is calculated. For example, unemployment can be calculated based on different methods or different base periods (e.g. relative changes with respect to certain point in time). In this case there is a discussion whether to use the same *qb:MeasureProperty* or define different.

Composite measures may be derived from other measures [34, 35] e.g. "Unemployment Rate" as ratio of the number of unemployed people in the labour force to the total labour force. This relation should somehow be expressed by linking to the number of unemployed people and the total labour force. In this case the computation of aggregated "total" values for composite measures would also be possible. For example the computation of the "total unemployment rate" based on "male" and "female" unemployment rate. Currently, there is no LOSD standard that can express these relations.

Additionally, having explicitly defined which aggregation functions (e.g. sum, average, mean, min, max) are applicable to a measure is useful for presentation and further processing purposes. QB4OLAP [20] has proposed an extension of QB vocabulary to express aggregation function. The applicability of aggregate functions to measures depends on various factors [36] (e.g. cube dimensions, units) and needs to be further explored.

A modeling challenge is related with the definition of multiple measures at a cube. Our study has shown that cubes with multiple measures should be published only when measures are closely related to a single observational event (e.g. sensor network measurements). If the measures are essentially independent then they should be modelled at separate cubes. However, there's a large grey area between the two since it depends on what is defined as an observational event and at the actual usage of the data.

Finally, there are some challenges related with the performance of applications that consume LOSD. For instance, the use of the *qb:codeList* indicates all the potential values of a *qb:DimensionProperty*. However, it is common to not use all the values at the cube e.g. a code list may contain values for the geography of Europe, but the cube uses only values for Greece. In this case there is no way to retrieve only the used values from the cube structure. They can be only retrieved by demanding SPARQL queries that iterate over all the cube observations. The same case also applies to units of measure e.g. the units used at a cube versus all the units of the code list.

6. Conclusion

A major part of open data concern statistics. Recently many National Statistical Institutes and public authorities have adopted the linked data paradigm to publish Linked Open Statistical Data (LOSD) because it facilitates data integration on the Web. Towards this direction many standard vocabularies have been proposed (i.e., QB, SKOS, XKOS).

The publication of high quality LOSD can be an important primary material for added value services and products, which can increase government transparency, contribute to economic growth and provide social value to citizens. However, the creation of LOSD remains a demanding task mainly because of modelling challenges related either to the conceptual definition of the cube, or to the way of modelling cubes as linked data. These challenges are usually addressed by data publishers in an ad hoc manner thus hindering the publishing of LOSD in a uniform way and lead to the creation of LOSD silos. As a result LOSD from different sources cannot be easily integrated and generic software tools cannot be developed.

Towards this direction, experts and practitioners, that directly participate at the publishing of LOSD, are involved through an iterative approach in order to comprehend the modeling challenges, identify relevant publishing approaches and propose way to address these challenges. The result is a set of proposed approaches that support LOSD publishers to model their data and to apply common standards. However a set of open challenges related with the limitations of existing standards, lack of standards and modeling decisions still remain to be explored.

We anticipate that the analysis of the modelling challenges as well as the proposed approaches presented at this paper will trigger and contribute towards a discussion on the development of best practices for publishing LOSD, facilitating the combining and analysing of linked statistical data from diverse sources.

Acknowledgement

Part of this work was funded by the European Commission within the H2020 Programme in the context of the project OpenGovIntelligence (`http://OpenGovIntelligence.eu`) under grant agreement no. 693849. The authors would like to cordially thank all the experts who participated in the study.

References

[1] E. Kalampokis, E. Tambouris, K. Tarabanis, A classification scheme for open government data: towards linking decentralised data, Int. J. Web Eng. Technol 6 (3) (2011) 266—285.

[2] J. Attard, F. Orlandi, S. Scerri, S. Auer, A systematic review of open government data initiatives, Government Information Quarterly 32 (4) (2015) 399 – 418. doi:https://doi.org/10.1016/j.giq.2015.07.006.

[3] A. Zuiderwijk, M. Janssen, Open data policies, their implementation and impact: A framework for comparison, Government Information Quarterly 31 (1) (2014) 17 – 29. doi:https://doi.org/10.1016/j.giq.2013.04.003. URL `http://www.sciencedirect.com/science/article/pii/S0740624X13001202`

[4] E. Commission, Commission notice — guidelines on recommended standard licences, datasets and charging for the reuse of documents (2014).

[5] S. Capadisli, S. Auer, A.-C. Ngonga Ngomo, Linked sdmx data, Semantic Web 6 (2).

[6] R. Cyganiak, M. Hausenblas, E. McCuirc, Official statistics and the practice of data fidelity (2011) 135–151.

[7] E. Kalampokis, E. Tambouris, A. Karamanou, K. Tarabanis, Open statistics: The rise of a new era for open data?, in: H. J. Scholl, O. Glassey, M. Janssen, B. Klievink, I. Lindgren, P. Parycek, E. Tambouris, M. A. Wimmer, T. Janowski, D. Sá Soares (Eds.), Electronic Government, Springer International Publishing, Cham, 2016, pp. 31–43.

[8] M. Janssen, J. van den Hoven, Big and open linked data (bold) in government: A challenge to transparency and privacy?, Government Information Quarterly 32 (4) (2015) 363 – 368. doi:https://doi.org/10.1016/j.giq.2015.11.007. URL `http://www.sciencedirect.com/science/article/pii/S0740624X15001069`

[9] B. F. Lóscio, C. Burle, N. Calegari, Data on the web best practices:w3c recommendation, 2017.

[10] C. Bizer, T. Heath, T. Berners-Lee, Linked data-the story so far, Semantic Services, Interoperability and Web Applications: Emerging Concepts (2009) 205–227.

[11] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J.-N. Mazón, F. Naumann, T. Pedersen, S. B. Rizzi, J. Trujillo, P. Vassiliadis, G. Vossen, Fusion cubes: Towards self-service business intelligence, Int. J. Data Warehous. Min. 9 (2) (2013) 66–88. doi:10.4018/jdwm.2013040104.

[12] E. Kalampokis, E. Tambouris, K. Tarabanis, Linked open cube analytics systems: Potential and challenges, IEEE Intelligent Systems 31 (5) (2016) 89–92.

[13] S. Bischof, A. Harth, B. Kampgen, A. Polleres, P. Schneider, Enriching integrated statistical open city data by combining equational knowledge and missing value imputation, Web Semantics: Science, Services and Agents on the World Wide Web-doi:https://doi.org/10.1016/j.websem.2017.09.003.

[14] J. Kli-mek, J. Kucera, M. Necasky, D. Chlapek, Publication and usage of official czech pension statistics linked open data, Journal of Web Semantics 48 (2018) 1 – 21. doi:https://doi.org/10.1016/j.websem.2017.09.002.

[15] E. Chaniotaki, E. Kalampokis, E. Tambouris, K. Tarabanis, A. Stasis, Exploiting linked statistical data in public administration: The case of the greek ministry of administrative reconstruction, in: 23rd Americas Conference on Information Systems (AMCIS2017), 2017.

[16] R. Cyganiak, D. Reynolds, The rdf data cube vocabulary:w3c recommendation, 2014.

[17] A. Miles, S. Bechhofer, Skos simple knowledge organization system reference: W3c recommendation, 2009.

[18] R. Cyganiak, D. Gillman, R. Grim, Y. Jaques, W. Thomas, Xkos: An skos extension for representing statistical classifications, 2017.

[19] V. Mijovic, V. Janev, D. Paunovic, S. Vranes, Exploratory spatio-temporal analysis of linked statistical data, Journal of Web Semantics 41 (Supplement C) (2016) 1 – 8. doi:https://doi.org/10.1016/j.websem.2016.10.002.

[20] J. Varga, A. A. Vaisman, O. Romero, L. Etcheverry, T. B. Pedersen, C. Thomsen, Dimensional enrichment of statistical linked open data, Journal of Web Semantics 40 (2016) 22 – 51. doi:http://dx.doi.org/10.1016/j.websem.2016.07.003.

[21] E. Kalampokis, B. Roberts, A. Karamanou, E. Tambouris, K. Tarabanis, Challenges on developing tools for exploiting linked open data cubes, in: 3rd International Workshop on Semantic Statistics (SemStats2015) co-located with the 14th International Semantic Web Conference (ISWC2015), CEUR-WS, Vol.1551., 2015.

[22] C.-C. Hsu, B. A. Sandford, The delphi technique: making sense of consensus, Practical assessment, research & evaluation 12 (10) (2007) 1–8.

[23] S. J. Young, L. M. Jamieson, Delivery methodology of the delphi: A comparison of two approaches., Journal of Park & Recreation Administration 19 (1) (2001) 42 – 58.

[24] H. A. Linstone, M. Turoff, Delphi: A brief look backward and forward, Technological Forecasting and Social Change 78 (9) (2011) 1712–1719.

[25] C. Okoli, S. D. Pawlowski, The delphi method as a research tool: an example, design considerations and applications, Information & management 42 (1) (2004) 15–29.

[26] R. L. Custer, J. A. Scarcella, B. R. Stewart, The modified delphi technique: A rotational modification., Journal of Vocational and Technical Education 15 (2) (1999) 1 – 10.

[27] R. Gnanadesikan, Methods for statistical data analysis of multivariate observations, Vol. 321, John Wiley & Sons, 2011.

[28] F. S. Tseng, C.-W. Chen, Integrating heterogeneous data warehouses using xml technologies, Journal of Information Science 31 (3) (2005) 209–229. doi:10.1177/0165551505052467.

[29] S. Berger, M. Schrefl, From federated databases to a federated data warehouse system, in: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, IEEE, 2008, pp. 394–394.

[30] A. Datta, H. Thomas, The cube data model: a conceptual model and algebra for on-line analytical processing in data warehouses, Decision Support Systems 27 (3) (1999) 289–301.

[31] L. Cabibbo, R. Torlone, A logical approach to multidimensional databases, in: Advances in Database Technology–EDBT'98, Springer, 1998, pp. 183–197.

[32] M. Hausenblas, Exploiting linked data to build web applications, IEEE Internet Computing 13 (4) (2009) 68–73. doi:http://doi.ieeecomputersociety.org/10.1109/MIC.2009.79.

[33] B. F. Loscio, C. Burle, N. Calegari, Data on the web best practices: W3c recommendation, 2017.

[34] S. F. Pileggi, J. Hunter, An ontological approach to dynamic fine-grained urban indicators, Procedia Computer Science 108 (2017) 2059 – 2068, international Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

[35] M. Denk, W. Grossmann, Towards a best practice of modeling unit of measure and related statistical metadata, IMF working paper.

[36] E. Kalampokis, E. Tambouris, K. Tarabanis, ICT tools for creating, expanding, and exploiting statistical linked open data, Statistical Journal of the IAOS 33 (2) (2017) 503–514.