Enabling Spatial OLAP Over Environmental and Farming Data with QB4SOLAP

Nurefşan Gür[1(✉)], Katja Hose[1], Torben Bach Pedersen[1], and Esteban Zimányi[2]

[1] Department of Computer Science, Aalborg University, Aalborg, Denmark
{nurefsan,khose,tbp}@cs.aau.dk
[2] Department of Computer and Decision Engineering, Université Libre de Bruxelles, Bruxelles, Belgium
ezimanyi@ulb.ac.be

Abstract. Governmental organizations and agencies have been making large amounts of spatial data available on the Semantic Web (SW). However, we still lack efficient techniques for analyzing such large amounts of data as we know them from relational database systems, e.g., multidimensional (MD) data warehouses and On-line Analytical Processing (OLAP). A basic prerequisite to enable such advanced analytics is a well-defined schema, which can be defined using the QB4SOLAP vocabulary that provides sufficient context for spatial OLAP (SOLAP). In this paper, we address the challenging problem of MD querying with SOLAP operations on the SW by applying QB4SOLAP to a non-trivial spatial use case based on real-world open governmental data sets across various spatial domains. We describe the process of combining, interpreting, and publishing disparate spatial data sets as a spatial data cube on the SW and show how to query it with SOLAP operators.

1 Introduction

In late 2012, the Danish government joined the Open Data movement by making several raw digital data sets [3] freely available at no charge. These data sets span domains such as environmental data, geospatial data, business data from transport to tourism, fishery, forestry, and agriculture. GovAgriBus Denmark[1] was an initial effort in 2014 to make Danish government Open Data from various domains available as Linked Open Data (LOD) [2] on the Semantic Web in order to pose queries across domains. If the corresponding domains can be related through space and location, spatial attributes of these data sets become particularly interesting as we can derive spatial joins and containment relationships that were not encoded in the original data sets. Danish government organizations and agencies continue publishing data sets for new domains and update the corresponding data sets regularly on a yearly basis, which brings opportunities in querying the expanding spatial data with analytical perspectives on the Semantic Web. Responding to such queries is a complex task, which requires well-defined schemas to facilitate OLAP operations on the Semantic Web. QB4SOLAP [7]

[1] https://datahub.io/dataset/govagribus-denmark.

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 287–304, 2016.
DOI: 10.1007/978-3-319-50112-3_22

aims to support intelligent multidimensional querying in SPARQL by providing context to SOLAP and its elements on the SW. However, QB4SOLAP has not been applied on complex real-world data yet. This could bring particular challenges with the use of real-world spatial data. In this paper, we address the challenging problem of multidimensional querying with SOLAP operations on the SW by applying QB4SOLAP on real-world open governmental data sets from various domains. These domains span from livestock farming to environment, where many of them have spatial information.

In this paper, we design a spatial data cube schema with data from livestock farming, environment, and geographical domains. Every data set is downloaded from different governmental sources in various formats. The downloaded data is prepared and conciliated with a spatial data cube schema in order to publish it on the SW with QB4SOLAP. We use the common SOLAP operators [8] on the spatial data cube for advanced analytical queries. These analytical queries give perspective on the use case data sets that are linked and published with QB4SOLAP as a unified spatial data cube. Having the use case data sets with spatial attributes also allows us to reveal patterns across the use case domains that were not possible before. We share our experiences with the best practices and methods together with the lessons learned. Finally, we show how to formulate and execute SPARQL queries with individual and nested SOLAP operations.

The remainder of this paper is structured as follows. Section 2 presents the background and motivation, Sect. 3 discusses related work and presents the state-of-the-art spatial data cubes on the SW. Section 4 presents the data sources for the use case while Sect. 5 describes how to annotate and publish the use case data as a spatial data cube on the SW. Section 6 presents SOLAP operators and their SPARQL implementation. Section 7 presents a brief overview of the process and reflects on the problems and improvements. Finally, Sect. 8 concludes the paper with an outlook to future work.

2 Background and Motivation

The Semantic Web supports intelligent querying via SPARQL with active inference and reasoning on the data in addition to capturing its semantics. Linked Open Data on the Semantic Web is an important source to support Business Intelligence (BI). Multidimensional data warehouses and OLAP are advanced analytical tools in analyzing complex BI data. State-of-the-art SW technologies support advanced analytics over *non-spatial* SW data. QB4SOLAP supports intelligent multidimensional querying in SPARQL by providing context to spatial data warehouses and its concepts. Variety of the data is an intriguing concept on both the Semantic Web and in complex BI systems. The variety of the data and heterogeneous representation formats (e.g., CSV, JSON, PDF, XML, and SHP) require underlying conceptualizations and data models with well-defined spatial (and temporal) dependencies, which can be modeled with QB4SOLAP in order to answer complex analytical queries. Complex queries cannot be answered from within one domain alone but span over multiple disciplines and various data sources. As a result, this paper is driven by the motivation of using QB4SOLAP as a proof of concept for spatial data warehouses on the Semantic Web by using open (government) data of various domains from different sources, which creates a non-trivial spatial use case.

3 State of the Art

Data warehouses and OLAP technologies have been successful for analyzing large volumes of data [1], including integrating with external data such as XML [16]. Combining DW/OLAP technologies with RDF data makes RDF data sources more available for interactive analysis. Kämpgen et al. propose an extended model [11] on top of the RDF Data Cube Vocabulary (QB) [4] for interacting with statistical linked data via OLAP operations directly in SPARQL. In OLAP4LD [10], Kämpgen *et al.* suggest enhancing query performance of OLAP operations expressed as SPARQL queries by using RDF aggregate views. The W3C published a list of RDF cube implementations [19]. However, they all have inherent limitations of QB and thus cannot support OLAP dimensions with hierarchies and levels, and built-in aggregate functions.

Etcheverry et al. [6] introduce QB4OLAP as an extended vocabulary based on QB, with a full MD metamodel, supporting OLAP operations directly over RDF data with SPARQL queries. Matei *et al.* [12] use QB and QB4OLAP as a basis to support OLAP queries in Graph Cube [20] with the IGOLAP vocabulary. Jakobsen *et al.* [9] study OLAP query optimization techniques over QB4OLAP data cubes. However, none of these approaches and vocabularies support *spatial* DWs.

QB4SOLAP(v1) [7] is the first attempt to model and query spatial DWs on the SW, and QB4SOLAP(v2) [8] is a foundation for spatial data warehouses and SOLAP operators on the SW, which is currently under submission with completely revised formal semantics of SOLAP operators and SPARQL query generations algorithms. QB4SOLAP is an extension of QB4OLAP with spatial concepts. The QB4OLAP vocabulary is compatible with the QB vocabulary. Therefore QB4SOLAP provides backward compatibility with other statistical or MD data cube vocabularies in addition to providing spatial context for querying with SOLAP. Figure 1 depicts the QB4SOLAP(v2) vocabulary. Capitalized terms with non-italic font represent RDF classes, capitalized terms with italic font represent RDF instances, and non-capitalized terms represent RDF properties. Classes in external vocabularies are depicted in light gray background and font. QB, QB4OLAP, and QB4SOLAP classes are shown with white, light gray, and dark gray backgrounds. Original QB terms are prefixed with `qb:`[2]. QB4OLAP and QB4SOLAP terms are prefixed with `qb4o:`[3] and `qb4so:`[4]. Spatial classes are prefixed with `geo:`[5], where the spatial extension to QB4SOLAP is based on the GeoSPARQL [15] standard from the Open Geospatial Consortium (OGC) for representing and querying geospatial linked data on the SW.

[2] RDF Cube: http://purl.org/linked-data/cube#.
[3] QB4OLAP: http://purl.org/qb4olap/cubes#.
[4] QB4SOLAP: http://w3id.org/qb4solap#.
[5] GeoSPARQL: http://www.opengis.net/ont/geosparql#.

QB4SOLAP is a promising approach for modeling, publishing, and querying spatial data warehouses on the SW. However, it has only been validated with a synthetic use case. Andersen et al. [2] consider publishing/converting open Danish governmental spatial data as Linked Open Data without considering the MD aspects of geospatial data. In this paper, however, we validate QB4SOLAP with a non-trivial use case, which is created as a spatial data cube from open Danish government spatial data. Furthermore, we show how to exploit multidimensional spatial linked data on the SW, which is not solely about adding semantics and linking disparate data sets on the SW, but also about enabling analytical queries by modeling them as spatial data cubes.

4 Source Data

In order create a spatial data cube of livestock holdings in Danish farms, we have gathered data that is published by different agencies in Denmark. We have found these domains to be particularly interesting as they represent a non-trivial use case that covers spatial attributes and measures, which can be modeled in a spatial data cube for multidimensional analysis. In the following we first give a brief overview of the flat data and their sources, and then represent the whole use case data set as a spatial data cube in Sect. 5.

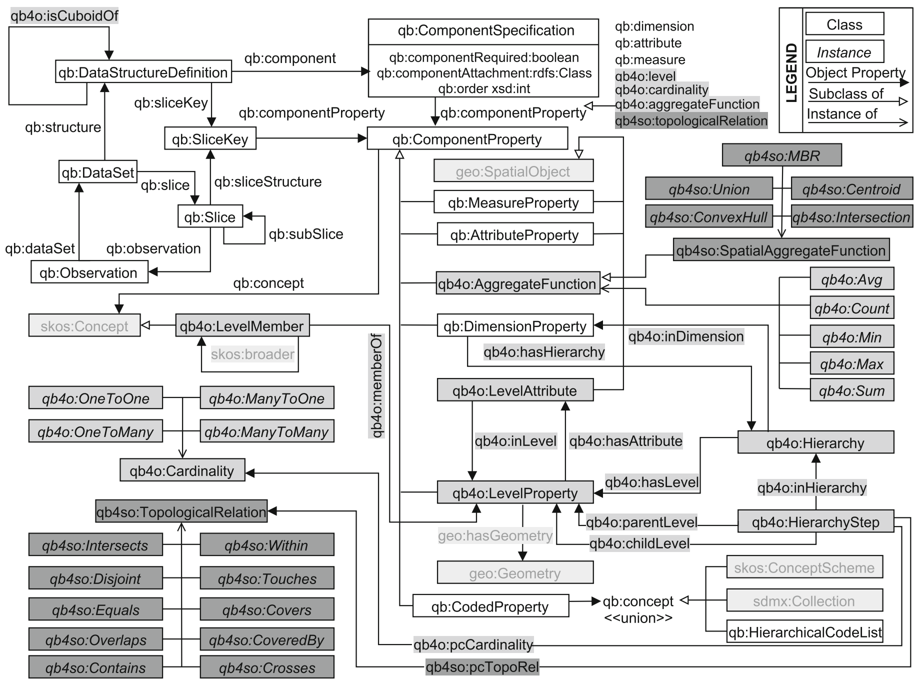

Fig. 1. QB4SOLAP vocabulary

The environmental protection agency under the Ministry of Environment and Food of Denmark regulates the livestock units (DE)[6] per area in order to keep nitrate leaching under control in vulnerable areas. Prohibition rules against the establishment of livestock farms and the siting of animal housing are determined with respect to livestock units and distance to specific natural habitats (e.g., ammonia vulnerable areas, water courses, and water supply facilities etc.) [13].

Livestock Farming (CHR) Data. The Ministry of Environment and Food of Denmark (http://en.mfvm.dk) publishes the central husbandry (livestock) registry (CHR) data, which is the central database used for registration of holdings and animals. We refer to this set of data as *CHR data*. We have downloaded several relevant data sets from http://jordbrugsanalyser.dk in livestock farming domain. The CHR data collection is downloaded in SHP format as 6 data sets, where each data set represents the state of the farms for a year between the years 2010 to 2015. SHP format is used for *shapefile*s, which store geometry information of the spatial features in a data set. In each shapefile, there is information about more than 40,000 farms. In total, the CHR data collection contains around 240,000 records. Farm locations are given as `(X,Y)` point coordinates. Each data set has 24 attributes in which the important ones are: *CHR - Central Husbandry (animal) Registry (holding) number, CVR - Central Company Registry number (owner company of the holding), DE (Livestock unit), Address of the holding (Postnr and Commune), Geographical position of the holding (X and Y coordinates), Different type of normalized herds, Number of animals for each herd, Animal code and label, Animal usage code and purpose.*

Environmental Data. Public environmental data is published on Denmark's environment portal http://www.miljoeportal.dk/, where we can find information about nitrate catchment areas and vulnerable sites. The soil measurements contain data from 2008 to 2015 [14]. We downloaded the data sets in SHP format, which have recently become available on the portal. The files record measurements of the soil quality across Denmark. The environmental data collection contains 3 data sets about nitrogen reduction potentials and phosphor and nitrate classifications of the soil. Temporal validity of the soil measurement data is recorded in the attributes with timestamps. Each data set keeps records of polygon areas. In total, the environmental data collection contains around 30,000 records. Datasets have attribute fields about the area of the polygons, CVR number of the data provider agency or company, responsible person name, etc. The important attributes, which record the soil measurement data are: *Nitrate class type, Nitrogen reduction potentials, Phosphor class type.*

Geographical (Regions) Data. The primary use case data is built around livestock farming (CHR) and environmental data as mentioned above. In sorder to pursue richer analysis upon this use case we enrich the spatiality of the use

[6] Livestock units are used to produce statistics describing the number of livestocks in farms.

case data by adding two geographical data sets; parishes and drainage areas of Denmark. These data sets are spatially and topically relevant since we have found pre-aggregated maps created by the Ministry of Environment and Food of Denmark at parish and drainage area levels for livestock farming data. We downloaded parishes and drainage areas of Denmark as SHP files from http://www.geodata-info.dk/. The total number of records of the geographical data collection are 2,300. These data sets have attribute fields such as: *Drainage area name, Parish name, Total area, etc.*

Central Company Registry (CVR) Data. Danish companies, agencies and industries are registered in the Central Company Register (CVR). Every livestock holding is owned by a company and has a CVR number. Environmental data also records the CVR number of the corresponding data provider agencies. Through this CVR number, we can access detailed information of the companies and contact details of the responsible people. This collection allows evaluating interesting queries with the selected domains given above. The CVR data is published at http://cvr.dk and can be accessed via a web service with a Danish social security number log-in. We accessed and downloaded only the data in CSV format that are accredited for publishing. This data includes attributes such as: *Company name, Phone number, and Address etc.*

5 Publishing Spatial Data Cubes with QB4SOLAP

The QB4SOLAP vocabulary allows to define *cube schemas* and *cube instances*. A cube schema defines the structure of a cube as an instance of the class `qb:DataStructureDefinition` in terms of dimension levels, measures, aggregation functions (e.g., SUM, AVG, and COUNT) on measures, spatial aggregation functions[7] on spatial measures, fact-level cardinality relationships, and topological relationships. The properties used to express these relationships are: `qb4o:level`, `qb:measure`, `qb4o:aggregateFunction`[8], `qb4o:cardinality`, and `qb4so:topologicalRelation` respectively (Fig. 1). These schema level metadata are used to define MD data sets in RDF. Cube instances are the members of a cube schema that represent level members, facts and measure values. We describe the cube schema elements in Sect. 5.1 and the cube instances in Sect. 5.2 with their examples.

5.1 GeoFarmHerdState Cube Schema in RDF

As our use case we create a spatial data cube of livestock holdings that we refer to as *GeoFarmHerdState*. The use case data cube is created from the flat data sets of the livestock farming (CHR) data, the environmental data, the

[7] Spatial aggregation functions aggregate two or more spatial objects and return a new spatial object, e.g., union, buffer, and convexHull etc.

[8] SpatialAggregateFunction is a subclass of AggregateFunction. Thus, measures and spatial measures use the same property `qb4o:aggregateFunction`.

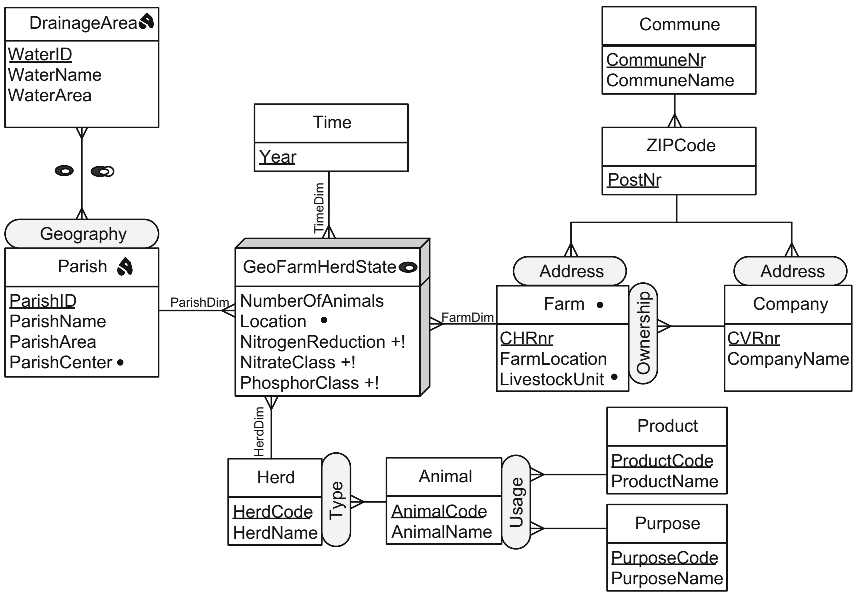

Fig. 2. GeoFarmHerdState – conceptual MD schema of livestock holdings data

geographical (regions) data, and the company registry (CVR) data collections, which are explained in Sect. 4. We create this data cube by thoroughly analyzing the attributes of the flat data sets from the collected relevant domains and conciliating them by foreign keys or by overlaying the SHP files of the spatial data sets and deriving new attributes from the intersected areas. After deriving useful spatial information across use case domains, we generalize the tabular data of the several use case data sets into the GeoFarmHerdState data cube. Figure 2 shows the multidimensional conceptual schema of the GeoFarmHerdState spatial cube. The multidimensional elements of the cube are explained in Remarks 1–7 followed by their examples in RDF. The underlying syntax for RDF examples is given in Turtle. We prefix the schema elements of the GeoFarmHerdState cube with `gfs:`.

Remark 1 (Dimensions). *Dimensions* provide perspectives to analyze the data. The GeoFarmHerdState cube has four dimensions, in which the two of them are *spatial* (FarmDim, ParishDim). All dimensions in the cube are defined with `qb:DimensionProperty`. A dimension is spatial if it has at least one spatial level (See Remark 3). Dimension hierarchies are defined with `qb4o:hasHierarchy` property. Hierarchies and their types are explained later in Remark 2.

Example 1. We give two spatial dimensions as an example.

```
gfs:farmDim rdf:type qb:DimensionProperty; qb4o:hasHierarchy gfs:ownership , gfs:address.
gfs:parishDim rdf:type qb:dimensionProperty; qb4o:hasHierarchy gnw:geography.
```

Remark 2 (Hierarchies). *Hierarchies* allow users to aggregate measures at various levels of detail. Hierarchies are composed of levels. A hierarchy is *spatial* if it has at least one spatial level (See Remark 3). Hierarchies of the GeoFarmHerdState cube are given in ellipses (Fig. 2). Each hierarchy is defined with `qb4o:Hierarchy` and linked to its dimension with the `qb4o:inDimension` property. Levels that belong to the hierarchy are defined with the `qb4o:hasLevel` property.

Example 2. We present the most interesting hierarchies from the GeoFarmHerdState cube as an example.

```
gfs:geogprahy rdf:type qb4o:Hierarchy; qb4o:inDimension gfs:parishDim; qb4o:hasLevel gfs:drainageArea.
gfs:usage rdf:type qb4o:Hierarchy; qb4o:inDimension gfs:animalDim; qb4o:hasLevel gfs:product , gfs:purpose.
gfs:address rdf:type qb4o:Hierarchy; qb4o:inDimension gfs:farmDim; qb4o:hasLevel gfs:zipCode , gfs:commune.
```

The Geography hierarchy is a *non-strict* spatial hierarchy. A spatial hiearchy is non-strict if it has at least one $(n - n)$ relationship between its levels. In the Geography hierarchy (Fig. 2) the $(n-n)$ cardinality represents that a parish may belong to more than one drainage area. Usually, non-strict spatial hierarchies arise when a partial containment relationship exists, which is given as *Intersects* in our use case. Usage hierarchy is a *generalized* hierarchy with non-exclusive paths to splitting levels (Product and Purpose) and has no joining level but the top level *All*. Finally, the Address and Ownership hierarchies are *parallel dependent* hierarchies. Parallel hierarchies arise when a dimension has several hierarchies sharing some levels. Note that the Address hierarchy has different paths from the Company and Farm levels (Fig. 2).

Remark 3 (Levels). *Levels* have a set of attributes (See Remark 4) that describes the characteristics of the level members (See Remark 9). Levels are defined with the `qb4o:LevelProperty` and their attributes are linked with the `qb4o:hasAttribute` property. A level is *spatial* if it has an associated geometry. Therefore, spatial levels have the property `geo:hasGeometry`, which defines the geometry of the spatial level in QB4SOLAP.

Example 3. We present a spatial level (Parish) as an example with its attributes. Attributes and spatial attributes of levels are further described in Remark 4.

```
gfs:parish rdf:type qb4o:LevelProperty; qb4o:hasAttribute gfs:parishID;
    qb4o:hasAttribute gfs:parishName; qb4o:hasAttribute gfs:parishArea;
    qb4o:hasAttribute gfs:parishCenter; geo:hasGeometry gfs:parishPolygon.
```

Note that the Parish level is defined as a spatial level because it has an associated polygon geometry (`gfs:parishPolygon`), which is specified with the `geo:hasGeometry` property. Some other spatial characteristics of the levels can be recorded in the spatial attributes of the level such as the center point of the parish (`gfs:parishCenter`).

Remark 4 (Attributes). Attributes and *spatial* attributes are defined with the `qb4o:LevelAttribute` property and linked to their levels with the `qb4o:inLevel` property. An attribute is spatial if it is defined over a spatial domain. Attributes are defined as ranging over XSD literals[9] and spatial attributes must be ranging over spatial literals, i.e., well-known text literals (WKT) from OGC schemas[10]. Spatial attributes are a sub-property of the `geo:Geometry` class. Further, the domain of the spatial attribute should be specified with `rdfs:domain`, which must be a geometry. Finally, the spatial attribute must be specified as an instance of `geo:SpatialObject` with the `rdfs:subClassOf` property. Examples of attributes are given in the following.

Example 4. We present spatial and some non-spatial attributes of the Parish level.

```
gfs:parishID rdf:type qb4o:LevelAttribute; qb4o:inLevel gfs:parish; rdfs:range xsd:positiveInteger.
gfs:parishName rdf:type qb4o:LevelAttribute; qb4o:inLevel gfs:parish; rdfs:range xsd:string.
gfs:parishCenter rdf:type qb4o:LevelAttribute; rdfs:subPropertyOf geo:Geometry; qb4o:inLevel gfs:parish;
    rdfs:domain geo:Point; rdfs:subClassOf geo:SpatialObject; rdfs:range geo:wktLiteral , virtrdf:Geometry.
```

We have mentioned in Remark 3 that spatial levels are defined through their associated geometries, which are not given as a level attribute. For the Parish level we present the following example of the corresponding geometry.

```
gfs:parishPolygon rdf:type geo:Geometry; rdfs:domain geo:MultiSurface;
    rdfs:subClassOf geo:SpatialObject; rdfs:range geo:wktLiteral , virtrdf:Geometry.
```

Remark 5 (Hierarchy Steps). Hierarchy steps define the structure of the hierarchy in relation to its corresponding, levels. A hierarchy step entails a roll-up relation between a lower (child) level and an upper (parent) level with a cardinality. The cardinality $(n-n, 1-n, n-1, n-n)$ relationship describes the number of members in one level that can be related to a member in the other level for both child and parent levels. A hierarchy step is *spatial* if it relates a spatial child level and a spatial parent level, in which case it entails a topological relationship between these spatial levels. Both spatial and non-spatial hierarchy steps are defined as a blank node with the `qb4o:HierarchyStep` property and linked to their hierarchies with the `qb4o:inHierarchy` property. The parent and child levels are linked to hierarchy steps with the `qb4o:childLevel` property and the `qb4o:parentLevel` property. The cardinality of a hierarchy step is defined by the `qb4o:pcCardinality` property. And finally, the topological relationship[11] of a hierarchy step is defined by the `qb4so:pcTopoRel` property.

Example 5. The following illustrates the hierarchy steps of the spatial hierarchy Geography and non-spatial hierarchy Address as it has different paths from child levels Farm and Company.

```
## Geography hierarchy structure ##
_:geography_hs1 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:geography;
    qb4o:childLevel gfs:parish; qb4o:parentLevel gfs:drainageArea;
    qb4o:pcCardinality qb4o:ManyToMany; qb4so:pcTopoRel qb4so:Intersects, qb4so:Within.
## Address hierarchy structure ##
```

[9] XML Schema Definition: http://www.w3.org/TR/xmlschema11-1/.

[10] OGC Schemas: http://schemas.opengis.net/.

[11] Topological relations are Boolean predicates that specify how two spatial objects are related to each other, e.g., within, intersects, touches, and crosses etc.

```
_:farm_address_hs1 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:address;
    qb4o:childLevel gfs:farm; qb4o:parentLevel gfs:zipCode;
    qb4o:pcCardinality qb4o:ManyToOne.
_:farm_address_hs2 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:address;
    qb4o:childLevel gfs:zipCode; qb4o:parentLevel gfs:commune;
    qb4o:pcCardinality qb4o:ManyToOne.
_:company_address_hs1 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:address;
    qb4o:childLevel gfs:company; qb4o:parentLevel gfs:zipCode;
    qb4o:pcCardinality qb4o:ManyToOne.
_:company_address_hs2 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:address;
    qb4o:childLevel gfs:zipCode; qb4o:parentLevel gfs:commune;
    qb4o:pcCardinality qb4o:ManyToOne.
```

Remark 6 (Measures). Measures record the values of a phenomena being observed. Measures and *spatial* measures are defined with `qb:MeasureProperty`. A measure is spatial if it is defined over a spatial domain. Similarly to attributes (Remark 4), measures are defined ranging over XSD literals and spatial measures must be ranging over spatial literals.

Example 6. The following shows an example of a spatial measure (Location) and a non-spatial measure (NumberOfAnimals).

```
gfs:location rdf:type qb:MeasureProperty; rdfs:subPropertyOf sdmx-measure:obsValue;
    rdfs:subClassOf geo:SpatialObject; rdfs:domain geo:Point;
    rdfs:range geo:wktLiteral , virtrdf:Geometry.
gfs:numberOfAnimals rdf:type qb:MeasureProperty;
    rdfs:subPropertyOf sdmx-measure:obsValue; rdfs:range xsd:decimal.
```

Remark 7 (Fact). Fact defines the data structure (DSD) of the cube with `qb:DataStructureDefinition`. The dimensions are given as `components` and defined with the `qb4o:level` property as the dimensions are linked to the fact at the lowest granularity level. A fact is *spatial* if it relates two ore more spatial levels. Similarly, measures are given as `components` of the fact and are defined with the `qb:measure` property. Aggregation functions on measures and spatial aggregation functions on spatial measures are also defined in the DSD with `qb4o:aggregateFunction`. Fact-level cardinality relationships and topological relationships are defined with `qb4o:cardinality` and `qb4so:topologicalRelation` in DSD, respectively (Fig. 1).

Example 7. The following shows the data structure definition of the cube GeoFarmHerdState, which is defined with corresponding measures and dimensions.

```
# – GeoFarmHerdState Cube Definition of the Fact FarmHerdState
gfs:GeoFarmHerdState rdf:type qb:DataStructureDefinition;
    # Lowest level for each dimensions in the cube
    qb:component [qb4o:level gfs:herd; qb4o:cardinality qb4o:ManyToOne ];
    qb:component [qb4o:level gfs:time; qb4o:cardinality qb4o:ManyToOne ];
    qb:component [qb4o:level gfs:farm; qb4o:cardinality qb4o:ManyToOne;
        qb4so:topologicalRelation qb4so:Equals ];
    qb:component [qb4o:level gfs:parish; qb4o:cardinality qb4o:ManyToMany;
        qb4so:topologicalRelation qb4so:Within ];
    # Measures in the cube
    qb:component [qb:measure gfs:numberOfAnimals; qb4o:aggregateFunction qb4o:Sum];
    qb:component [qb:measure gfs:location; qb4o:aggregateFunction qb4so:ConvexHull];
    qb:component [qb:measure gfs:nitrogenReduction; qb4o:aggregateFunction qb4o:Avg];
    qb:component [qb:measure gfs:nitrateClass; qb4o:aggregateFunction qb4o:Avg];
    qb:component [qb:measure gfs:phosphorClass; qb4o:aggregateFunction qb4o:Avg].
```

5.2 GeoFarmHerdState Cube Instances in RDF

Cube instances are the members of a cube schema that represent level members and facts (members), which are explained in Remarks 8 and 9 below. We prefix the instances of the GeoFarmHerdState cube with `gfsi:`.

Remark 8 (Fact members). Fact members (i.e., facts of FarmHerdState) are instances of the `qb:Observation` class. Each fact member is related to a set of dimension *base* level members and has a set of measure values. Every fact member has a unique identifier (IRI) which is prefixed with `gfsi:`.

Example 8. The following shows an example of a single fact member, which represents the state of a farm with CHR no. 39679 in the year 2015 that has the herd code 15.

```
gfsi:farm_39679_2015 rdf:type qb:Observation;
## Dimension levels and base level members associated with the fact member
    gfs:herdCode gfsi:herd_15; gfs:year gfsi:year_2015;
    gfs:chrNumber gfsi:farm_39679; gfs:parishID gfsi:parish_8311;
## Measures associated with the fact member
    gfs:numberOfAnimals "100.0"^^xsd:decimal; gfs:nitrateClass "3"^^xsd:integer;
    gfs:nitrogenReduction "0.75"^^xsd:decimal; gfs:phosporClass "3"^^xsd:integer;
    gfs:location "POINT(8.3713 56.7912)"^^geo:wktLiteral.
```

Remark 9 (Level members). Level members are defined with `qb4o:LevelMember`. They are linked to their corresponding levels from the schema with the `qb4o:memberOf` property. For each level member there is a set of attribute values. Due to the roll-up relations between levels of hierarchy steps (Remark 5), the `skos:broader` property relates a child level member to its parent level member.

Example 9. The following shows an example of a child level member in the Parish level and one of its parent level members in the DrainageArea level from the Geography dimension. Figure 3 presents a map snapshot for fact members and level members. Parish level member “Astrup” is highlighted and DrainageArea level member “Mariager Inderfjord” is marked with red borders. Note that Astrup intersects another drainage area “Langerak”, therefore it links to two parent level members via `skos:broader`.

```
## Parish level member
gfsi:parish_8648 rdf:type gfs:parish;
    qb4o:memberOf gfs:parish; skos:broader gfsi:water_3710, gfsi:water_159;
    gfs:parishID 8311; gfs:parishName"Astrup"; gfs:parishArea 46,118;
    gfs:parishCenter"POINT(8.2552, 56.8176)"^^geo:wktLiteral;
    gfs:parishPolygon"POLYGON((8.4038 56.7963, 8.3984 56.7721, 8.3689 56.7410, 8.3411 56.7372, 8.3078
    56.7281, 8.2987 56.7601, 8.2563 56.7763, 8.3112 56.8087, 8.3511 56.8137, 8.4038 56.7963))"^^geo:wktLiteral.
## DrainageArea level member
gfsi:water_159 rdf:type gfs:drainageArea;
    qb4o:memberOf gfs:drainageArea; gfs:waterID 159;
    gfs:waterName"Mariager Inderfjord"; gfs:waterArea 267,477;
    gfs:drainageGeo "POLYGON((8.6048 56.9843, 8.5908 56.8969, 8.5707 56.8664,
    8.5975 56.8519, 8.5215 56.8483, 8.3959 56.7625, 8.3938, 56.7340, 8.3613 56.6802,
    8.2584 56.7764, 8.2475 56.7051, 8.2175 56.7232, 8.3121 56.8441, 8.2806 56.8659,
    8.3602 56.9569, 8.4786 56.9713, 8.5474 56.9905, 8.6048 56.9843))"^^geo:wktLiteral.
```

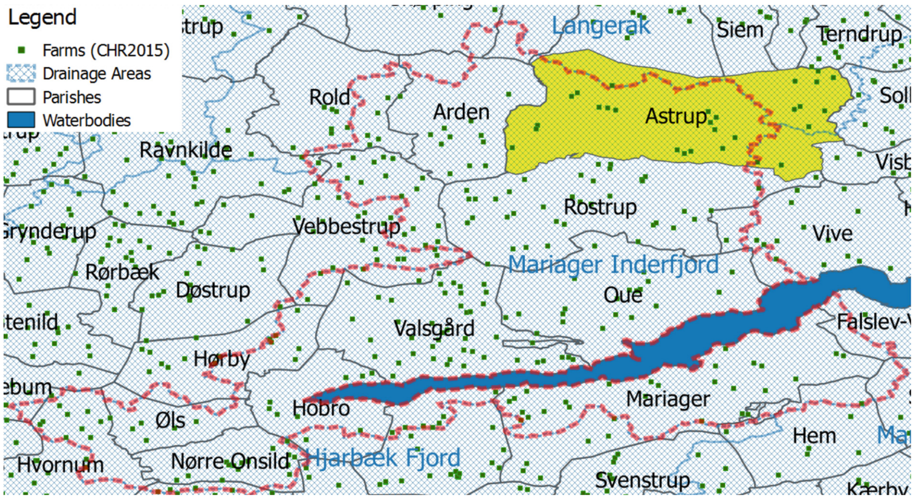

Fig. 3. GeoFarmHerdState – fact members and level members of Example 9 marked

6 SOLAP Operators over GeoFarmHerdState Cube

Spatial OLAP (SOLAP) operates on spatial data cubes. SOLAP increases the analytical capabilities of OLAP by taking into account the spatial information in the cube. SOLAP operators involve spatial conditions or spatial functions. Spatial conditions specify constraints (i.e., spatial Boolean predicates) on the geometries associated to cube members or measures, while spatial functions derive new data from the cube, which can be used, e.g., to derive dynamic spatial hierarchies.

6.1 SOLAP Operators

In what follows we present common SOLAP operators and examples of these operators on GeoFarmHerdState cube.

Remark 10 (S-Slice). The s-slice operator removes a dimension from a cube by choosing a single spatial value in a spatial level. It returns a cube with one dimension less.

Example 10. We can perform an s-slice operation in different ways.
1. Slice on farms (state of the farms) of the largest parish.
2. Slice on farms (state of the farms) of the drainage area containing `"POINT(10.43951 55.47006)"`.

The first one applies a spatial function call (for finding the *largest* parish by area) on a spatial level Parish and performs the slice. The second one applies a spatial predicate (for finding where a given point is *within* a particular drainage area) in a spatial level DrainageArea and performs the operation. The corresponding SPARQL queries are:

```
# 1 – s-slice with spatial function #
SELECT ?obs WHERE {
    ?obs rdf:type qb:Observation;
        gfs:parishID ?parish.
    ?parish gfs:parishPolygon ?parishGeo.
# Inner select for finding the largest parish
    { SELECT ?x (MAX(?area) as ?maxArea) WHERE{
    ?obs rdf:type qb:Observation;
        gfs:parishID ?parish.
    ?parish gfs:parishPolygon ?x.
    BIND (bif:st_area(?x) as ?area)}}
FILTER ?parishGeo = ?x) }
```

```
# 2 – s-slice with spatial predicate #
SELECT ?obs WHERE {
    ?obs rdf:type qb:Observation;
        gfs:parishID ?parish.
    ?parish qb4o:memberOf gfs:parish;
        skos:broader ?drainageArea.
    ?drainageArea gfs:drainageGeo ?drainageGeo.
FILTER (bif:st_within("POINT(10.43951 55.47006)",
?drainageGeo)) }
```

Remark 11 (S-Dice). The s-dice operator keeps the cells of the cube that satisfy the spatial predicate over dimension levels, attributes, or measures. It returns a subset of the cube with filtered members of the cube.

Example 11. In the following we show two examples of the s-dice operator.
1. Filter the farms located within 5 Km buffer from the center of a drainage area.
2. Filter the farms located within 2 Km distance from the center of their parish, which are in the nitrate class I areas.

In the first s-dice operation, initially, a spatial function is applied on level members of the DrainageArea level to get the *center* of their polygon geometries. Then, the level members of the Farm level are filtered with a spatial Boolean predicate with respect to the farm locations that are *within* a 5 Km buffer area of the center of the drainage areas. In the second s-dice operation, a spatial function is applied to the spatial measure farm location to get the *distance* of the farms from the center of their parish, which is followed by Boolean predicates; to filter the farms that are less than 2 Km away from the center of their parishes and are on nitrate class I areas.

```
# 1 – s-dice on dimension levels #
SELECT ?obs WHERE {
    ?obs rdf:type qb:Observation;
        gfs:farmID ?farm;
        gfs:parishID ?parish.
    ?farm gfs:farmLocation ?farmGeo.
    ?parish qb4o:memberOf gfs:parish;
        skos:broader ?drainageArea.
    ?drainageArea gfs:waterPolygon ?drainagePoly.
BIND (bif:st_centroid (?drainagePoly) as ?drainageCenter)
FILTER (bif:st_within(?drainageCenter, ?farmGeo, 5)) }
```

```
# 2 – s-dice on measures #
SELECT ?obs WHERE {
    ?obs rdf:type qb:Observation;
        gfs:location ?farmLocation;
        gfs:nitrateClass ?nitClass;
        gfs:parishID ?parish.
    ?parish gfs:parishCenter ?parishCent.
BIND (bif:st_distance (?farmLocation, ?parishCent)
AS ?distance)
FILTER (?distance < 2 && ?nitClass = 1)}
```

Remark 12 (S-Roll-up). The s-roll-up operator aggregates measures of a given cube by using an aggregate function and a spatial function along a spatial dimension's hierarchy. It returns a cube with measures at a coarser granularity for a given dimension.

Example 12. In the following, we present two examples of the s-roll-up operator.
1. Total amount of animals in the farms, which are closest to their parishes' center.
2. Average percentage of nitrogen reduction potentials in the parishes that are within and/or intersect the drainage area "Nibe-Bredning".

In the first s-roll-up operator, measures are aggregated to the Parish level after selecting the farms with respect to their proximity to the center of the parish with a spatial function. In the second s-roll-up operator, measures are aggregated to a specified drainage area ("Nibe-Bredning") at the DrainageArea level. We select all the possible topological cases where a parish intersects or within the drainage area, which means measures from the farms that are outside Nibe-Bredning are also aggregated to the level of this drainage area. In order to prevent this, the query needs to include an s-drill-down operator (Remark 13) to farms from Parish level and apply a spatial Boolean predicate to select the farms *within* the drainage area and then aggregate.

```
# 1 – s-roll-up #
SELECT ?parish (SUM(?animalCount) AS ?totalAnimals)
WHERE { ?obs rdf:type qb:Observation;
    gfs:numberOfAnimals ?animalCount;
    gfs:farmID ?farm;
    gfs:parishID ?parish.
?farm gfs:farmLocation ?farmGeo.
?parish gfs:parishCenter ?parishCent.
# Inner select for finding the
# closest farms to the parish centers #
    {SELECT ?farm1 (MIN(?distance) AS
    ?minDistance) WHERE
    { ?obs rdf:type ab:Observation;
        gfs:farmID ?farm1;
        gfs:parishID ?parish1.
    ?farm1 gfs:farmLocation ?farm1Geo.
    ?parish1 gfs:parishCenter ?parish1Cent.
    BIND (bif:st_distance (?farm1Geo, parish1Cent)
    AS ?distance) } GROUP BY ?farm1 }
    FILTER (?farm = ?farm1 && bif:st_distance
    (?farmGeo, ?parishCent) = ?minDistance )}
GROUP BY ?parish
```

```
# 2 – s-roll-up #
SELECT ?drainageArea (AVG(?nitRed) AS ?avgNitRed)
WHERE { ?obs rdf:type qb:Observation;
    gfs:location ?farmLocation;
    gfs:nitrogenReduction ?nitRed;
    gfs:parishID ?parish.
    ?parish qb4o:memberOf gfs:parish;
        gfs:parishPolygon ?parishGeo;
        skos:broader ?drainageArea.
    ?drainageArea gfs:memberOf gfs:drainageArea;
        gfs:waterPolygon ?drainageGeo;
        gfs:waterName ?drainageName.
FILTER (bif:st_within(?parishGeo, ?drainageGeo)
|| bif:st_intersects(?parishGeo, ?drainageGeo)
&& ?drainageName ="Nibe-Bredning")}
GROUP BY ?drainageArea
```

Remark 13 (S-Drill-down). The s-drill-down operator disaggregates measures of a given cube by using an aggregate function and a spatial function along a spatial dimension's hierarchy. It is the inverse operator of s-roll-up, therefore s-drill-down disaggregates the previously summarized data to a child level in order to obtain measures at a finer granularity.

6.2 Nested SOLAP Operations

A nested set of SOLAP operators can be designed with the pattern $(s\text{–}dice_2(s\text{–}roll\text{–}up_1(\ldots s\text{–}roll\text{–}up_k(s\text{–}slice_1(\ldots s\text{–}slice_n(s\text{–}dice_1(DataCube)))))))$. Initially a sub-cube is selected from the (spatial) data cube with the first s-dice. Afterwards, a number of s-slices can be applied, which is followed by a series of s-roll-ups. Finally, the expression ends with another s-dice for getting the final sub-cube at a coarser granularity by filtering the aggregated measures. In the following, we present a nested SOLAP operation example for the running case GeoFarmHerdState spatial data cube.

Example 13. $({}^3s\text{–}roll\text{–}up({}^2s\text{–}slice({}^1s\text{–}dice(GeoFarmHerdState))))$: This pattern represents a typical nested SOLAP operation that can be paraphrased for the running use case as follows: ^{1}Filter the farm states located within a 2 Km distance from the center of their parish and ^{2}slice on the parish which has the most number of topological relations (intersects, within) with a drainage area, ^{3}average the nitrogen reduction potential of the drainage areas intersecting with the parish.

7 Discussion and Perspectives

In the following, we give and evaluate the steps of our process with respect to the guidelines for publishing governmental linked data [18]. We discuss the particular challenges that we encountered and possible future improvements.

(1) Specification. The first step is to specify the scope of the data by identifying and analyzing the data sources. We identified the data sources for the domains of CHR data, Environmental data, Geographical data, and CVR data as described in Sect. 4. In order to find the correct relations between these domains we had to search documentations (i.e., [13,14]) and acquire knowledge about the domains' interests. As the purpose is to publish open data, the definition of an Open Data license is also required at this level.

(2) Modeling. We used the spatially extended MultiDim model [17] for designing the MD conceptual schema of the use case spatial data cube (Fig. 2) from the collected flat data sets. This process requires good knowledge of spatial data warehouses and its concepts. In order to model the spatial data cube in RDF, QB4SOLAP provides the state-of-the-art semantic spatial data cubes. Therefore, we annotate the designed use case conceptual schema with QB4SOLAP.

Modeling the RDF data with QB4SOLAP provides all the core concepts of spatial data warehouses (i.e., spatial dimensions, spatial levels, and spatial hierarchies) for spatial data on the SW. Therefore, QB4SOLAP conveniently handles the conceptual modeling process of DWs on the SW and clearly describes the certain relations that should be considered during the logical modeling process (e.g., cardinality and topological relationships to create integrity constraints for ER models).

(3) Generation. This step of the overall process involves the most complex tasks. In order to fully generate a spatial data cube in RDF the following subprocesses are performed: transformation and data conciliation.

(3.1) Transformation. The RDF triples were generated with ad-hoc C# code for mapping from relational CSV files to RDF. In total, 12 CSV files were organized based on the relational representation (snowflake schema) of an MD conceptual model such that: We obtained *one* fact table with foreign keys of the related dimension (base) levels and measures, *four* tables with each dimensions' base level, and *seven* tables for the remaining levels along the hierarchies. These 12 tables are related by referential integrity constraints. Every level table also records the level attributes and attribute values. In order to create this relational CSV files, we pursued a number of data conciliation activities as described in the following item.

(3.2) Data Conciliation. Initial data sets are downloaded in different formats i.e., SHP format for CHR, Environmental, and Geographical data; CSV format for CVR data. In order to create the desired relational implementation of the use case spatial data cube, we used the unique identifiers (i.e., CVR and CHR numbers) or utilized spatial joins by joining attributes from one geometry feature to another based on the spatial containment relationship. For instance, we overlaid the point coordinates of the farms from CHR data and polygon coordinates of three environmental data sets in order to intersect and find the soil quality measurements for NitrogenReduction, NitrateClass and PhosphorClass of each farm. Another interesting spatial join is utilized for relating the Parish level members with DrainageArea level members. Since there is an $(n - n)$ cardinality relationship, some parishes intersect with more than one drainage area, thus we used topological relationships (intersects and within) to find the related child and parent level members. For interacting and handling the spatial data, we used QGIS with integration to PostGIS[12]. We used PostgreSQL to create and export relational tables.

The lack of tools for mapping a spatial multidimensional model to the relational model has been an impediment since we have to use topological relationships, where there is an $(n - n)$ cardinality relationship. Therefore, semantic ETL for data warehouses [5] is an important research topic, which requires improvements also for spatial data. Semi-automated tool support of geo-semantic ETL for publishing data warehouses on the SW is a promising improvement for handling the above processes such that the spatial joins can be processed efficiently. Before publishing the final RDF data, a comprehensive data cleansing step is essential for removing redundant columns and cleaning the noise due to unescaped characters, denormalized spatial literals, and encoding problems.

(4) Publication. In order to store and publish the RDF data we chose the Virtuoso Universal Server as a triple store. The details about the SPARQL endpoint can be found on the project page http://extbi.cs.aau.dk/GeoFarmHerdState.

Publication of metadata in Danish and English languages should be completed. Also for enabling efficient discovery of published spatial data cubes, adding an entry of the data in the CKAN repository (`datahub.io`) is required.

(5) Exploitation. The goal of our research is to re-use open government data and publish it as spatial data cubes on the SW for advanced multidimensional analysis. Therefore, we show how to query in SPARQL with SOLAP operators.

We recognize the need for non-expert SW users to write their spatial analytical queries in our high-level SOLAP language instead of the lower-level complex SPARQL language. Thus, a query system with a GUI that can interpret spatial data cube schemas for allowing users to perform high level SOLAP operations is ongoing work. Performing SOLAP queries in SPARQL to work over multiple RDF cubes with *s-drill-across* and supporting spatial aggregation (*s-aggregation*) over spatial measures are other important improvements on exploitation of spatial data cubes.

[12] QGIS: http://www.qgis.org/ PostGIS: http://postgis.net/.

8 Conclusion and Future Work

The need for spatial analytical queries on the Semantic Web increases constantly with regularly published open government data, but there is a lack of effective solutions and efficient models. As a first attempt to publish spatial data cubes from open data, we have shown that the QB4SOLAP vocabulary can be used to link Danish government data that is published in different domains. First, we have studied the use case data sets thoroughly with corresponding regulations and requirements in order to satisfy cross-domain interests (e.g., tracking soil quality in livestock farms and farm animals density on drainage areas etc.). Second, we have conciliated the flat data sets in order to model the MD concepts of a spatial data cube. Third, we described the most popular individual SOLAP operators and a nested SOLAP operation pattern with examples and their SPARQL implementation.

In this paper, the QB4SOLAP vocabulary is validated by a non-trivial spatial use case. As a proof of concept, we showed that linking spatial (governmental open) data on the Semantic Web can be achieved at an advanced level, not solely linking spatial open data on the SW but also modeling this data for advanced analytical queries with SOLAP operations.

Several directions are interesting for future research: developing a geo-semantic ETL tool to support the process of creating spatial data cubes on the SW, a GUI for non-expert users to perform SOLAP operations on SW spatial cubes, extending the use case and implementing advanced SOLAP queries, such that; as s-drill-across and s-aggregation on the SW.

Acknowledgments. This research is partially funded by "The Erasmus Mundus Joint Doctorate in Information Technologies for Business Intelligence Doctoral College (IT4BI-DC)".

References

1. Abelló, A., Romero, O., Pedersen, T.B., Aramburu, M.J., et al.: Using semantic web technologies for exploratory OLAP: a survey. TKDE **27**, 571–588 (2014)
2. Andersen, A.B., Gür, N., Hose, K., Jakobsen, K.A., Pedersen, T.B.: Publishing danish agricultural government data as semantic web data. In: Supnithi, T., Yamaguchi, T., Pan, J.Z., Wuwongse, V., Buranarach, M. (eds.) JIST 2014. LNCS, vol. 8943, pp. 178–186. Springer, Heidelberg (2015). doi:10.1007/978-3-319-15615-6_13
3. Arendt, J.B.: Denmark releases its digital raw material, Ministry of Finance of Denmark, October 2012. http://uk.fm.dk/news/
4. Cyganiak, R., Reynolds, D., Tennison, J.: The RDF Data Cube Vocabulary (2014)
5. Nath, D.R.P., Hose, K., et al.: Towards a Programmable Semantic Extract-Transform-Load Framework for Semantic Data Warehouses. In: DOLAP, pp. 15–24 (2015)
6. Etcheverry, L., Vaisman, A., Zimányi, E.: Modeling and querying data warehouses on the semantic web using QB4OLAP. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 45–56. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10160-6_5

7. Gür, N., Hose, K., Pedersen, T.B., Zimányi, E.: Modeling and querying spatial data warehouses on the semantic web. In: Qi, G., Kozaki, K., Pan, J.Z., Yu, S. (eds.) JIST 2015. LNCS, vol. 9544, pp. 3–22. Springer, Heidelberg (2016). doi:10.1007/978-3-319-31676-5_1
8. Gür, N., Pedersen, T.B., Zimányi, E., Hose, K.: A foundation for spatial data warehouses on the semantic web. Journal paper, under submission (2016)
9. Jakobsen, K.A., Andersen, A.B., Hose, K., Pedersen, T.B.: Optimizing RDF data cubes for efficient processing of analytical queries. In: COLD (2015)
10. Kämpgen, B., Harth, A.: OLAP4LD – a framework for building analysis applications over governmental statistics. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8798, pp. 389–394. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11955-7_54
11. Kämpgen, B., O'Riain, S., Harth, A.: Interacting with statistical linked data via OLAP operations. In: Simperl, E., Norton, B., Mladenic, D., Della Valle, E., Fundulaki, I., Passant, A., Troncy, R. (eds.) ESWC 2012. LNCS, vol. 7540, pp. 87–101. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46641-4_7
12. Matei, A., Chao, K.-M., Godwin, N.: OLAP for multidimensional semantic web databases. In: Castellanos, M., Dayal, U., Pedersen, T.B., Tatbul, N. (eds.) BIRTE 2013-2014. LNBIP, vol. 206, pp. 81–96. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46839-5_6
13. Danish Ministry of the Environment. Consolidated Act on Livestock Farming Environmental Approvals (2012). http://eng.mst.dk/media
14. Directive, Nitrates: Danish nitrate action programme 2008–2015 regarding the nitrates directive; 91/676/eec. Technical report, Nitrates Directive (2012)
15. Open Geospatial Consortium: GeoSPARQL: A geographic query language for RDF data. W3C Recommendation (2014)
16. Pedersen, D., Riis, K., Pedersen, T.B.: Query optimization for OLAP-XML federations. In: DOLAP, pp. 57–64 (2002)
17. Vaisman, A., Zimányi, E.: Spatial data warehouses. In: Vaisman, A., Zimányi, E. (eds.) Data Warehouse Systems. Design and Implementation. DCSA, pp. 427–473. Springer, Heidelberg (2014)
18. Villazón-Terrazas, B., Vilches-Blázquez, L., Corcho, O., Gómez-Pérez, A.: Methodological guidelines for publishing government linked data. In: Wood, D. (ed.) Linking Government Data, pp. 27–49. Springer, New York (2011)
19. W3C.: Data Cube Implementations (2014). https://www.w3.org/2011/gld/wiki/Data_Cube_Implementations
20. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and OLAP multidimensional networks. In: SIGMOD, pp. 853–864 (2011)