Modeling and Querying Spatial Data Warehouses on the Semantic Web

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

[1] Aalborg University, Aalborg, Denmark
{nurefsan,khose,tbp}@cs.aau.dk
[2] Université Libre de Bruxelles, Brussels, Belgium
ezimanyi@ulb.ac.be

Abstract. The Semantic Web (SW) has drawn the attention of data enthusiasts, and also inspired the exploitation and design of multidimensional data warehouses, in an unconventional way. Traditional data warehouses (DW) operate over static data. However multidimensional (MD) data modeling approach can be dynamically extended by defining both the schema and instances of MD data as RDF graphs. The importance and applicability of MD data warehouses over RDF is widely studied yet none of the works support a spatially enhanced MD model on the SW. Spatial support in DWs is a desirable feature for enhanced analysis, since adding encoded spatial information of the data allows to query with spatial functions. In this paper we propose to empower the spatial dimension of data warehouses by adding spatial data types and topological relationships to the existing QB4OLAP vocabulary, which already supports the representation of the constructs of the MD models in RDF. With QB4SOLAP, spatial constructs of the MD models can be also published in RDF, which allows to implement spatial and metric analysis on spatial members along with OLAP operations. In our contribution, we describe a set of spatial OLAP (SOLAP) operations, demonstrate a spatially extended metamodel as, QB4SOLAP, and apply it on a use case scenario. Finally, we show how these SOLAP queries can be expressed in SPARQL.

1 Introduction

The evolution of the Semantic Web (SW) and its tools allow to employ complex analysis over multidimensional (MD) data models via On-Line Analytical Processing (OLAP) style queries. OLAP emerges when executing complex queries over data warehouses (DW) to support decision making. DWs store large volumes of data which are designed with MD modeling approach and usually perceived as *data cubes*. Cells of the cube represent the observation *facts* for analysis with a set of attributes called *measures* (*e.g.* a sales fact cube with measures of product quantity and prices). Facts are linked to *dimensions* which give contextual information (*e.g.* sales date, product, and location). Dimensions are perspectives which are used to analyze data, organized into *hierarchies* and

G. Qi et al. (Eds.): JIST 2015, LNCS 9544, pp. 3–22, 2016.
DOI: 10.1007/978-3-319-31676-5_1

levels that allow users to analyze and aggregate measures at different levels of detail. Levels have a set of *attributes* that describe the characteristics of the level members.

In traditional DWs, the "location" dimension is widely used as a conventional dimension which is represented in an alphanumeric manner with only nominal reference to the place names. This neither allow manipulating location-based data nor deriving topological relations among the hierarchy levels of the location dimension. This issue yields a demand for truly spatial DWs for better analysis purposes. Including encoded geometric information of the location data significantly improves the analysis process (*i.e.* proximity analysis of the locations) with comprehensive perspectives by revealing dynamic spatial hierarchy levels and new spatial members. The scope of this work is first focuses on enhancing the spatial characteristics of the cube members on the SW, and then describing and utilizing SOLAP operators for advanced analysis and decision making.

In our approach we consider enabling SOLAP capabilities directly over Resource Description Framework (RDF) data on the SW. Importance and applicability of performing OLAP operations directly over RDF data is studied in [9,12]. To perform SOLAP over the SW consistently, an explicit and precise vocabulary is needed for the modeling process. The key concepts of spatial cube members need to be defined in advance to realize SOLAP operations since they employ spatial measures with spatial aggregate functions (*e.g.* union, buffer, and, convex-hull) and topological relations among spatial dimension and hierarchy level members (*e.g.* within, intersects, and, overlaps). Current state of the art RDF and OLAP technologies is limited to support conventional dimension schema and analysis along it's levels. Spatial dimension schema and SOLAP require an advanced specialized data model. As a first effort to overcome the limitations of modeling and querying spatial data warehouses on the Semantic Web we give our contributions in the following.

Contributions. We propose an extended metamodel solution that enables representation and RDF implementation of spatial DWs. We base our metamodel on the most recent QB4OLAP vocabulary and present an extension to support the spatial functions and spatial elements of the MD cubes. We discuss the notion of a SOLAP operator and observe it with examples, then we give the semantics of each SOLAP operator formally and finally, show how to implement them in SPARQL by using sub-queries and nested set of operators.

In the remainder of the paper, we first present the state of the art, in Sect. 2. As a prerequisite for our contribution in Sect. 3, we give the preliminary concepts and explain the structure of a SOLAP operator. Then, in Sect. 4 we define the semantics of MD data cube elements in RDF, present QB4SOLAP and formalize the SOLAP operators over MD data cube elements. We present a QB4SOLAP use case in Sect. 5 and then, we show how to write the defined SOLAP queries over this use case in SPARQL in Sect. 6. Finally, in Sect. 7, we conclude and remark to the future work directions.

2 State of the Art

DW and OLAP technologies have been proven a successful approach for analysis purposes on large volumes of data [1]. Aligning DW/OLAP technologies with RDF data makes external data sources available and brings up dynamic scenarios for analysis. The following studies are found concerning DW/OLAP with the SW.

DW/OLAP and Semantic Web: The potential of OLAP to analyze SW data is recognized in several approaches, thus MD modeling from ontologies is studied in the works of [8,17]. However these approaches do not support standard querying of RDF data in SPARQL but require a MD or a relational database query engine, which limits the access to frequently updated RDF data. Kämpgen *et al.* propose an extended model [12] on top of RDF Data Cube Vocabulary (QB) [6] for interacting with statistical linked data via OLAP operations in SPARQL, but it has the limitations of the QB and thus cannot support full OLAP dimension structure with aggregate functions. It also has only limited support for complete MD data model members (*e.g.* hierarchies and levels). Etcheverry *et al.* introduce QB4OLAP [9] as an extended vocabulary of QB with a full MD metamodel, which supports OLAP operations directly over RDF data with SPARQL queries. However, none of these vocabularies and approaches support spatial DWs, unlike our proposal.

Spatial DW and OLAP: The constraint representation of spatial data has been focus in many fields from databases to AI [18]. Extending OLAP with spatial features has attracted the attention of data warehousing communities as well. Several conceptual models are proposed for representing spatial data in data warehouses. Stefanovic *et al.* [11] investigates on constructing and materializing the spatial cubes in their proposed model. The MultiDim conceptual model is introduced by Malinowski and Zimányi [16] which copes with spatial features and extended in [20], to include complex geometric features (*i.e.* continuous fields), with a set of operations and MD calculus supporting spatial data types. Gómez *et al.* [10] propose an algebra and a very general framework for OLAP cube analysis on discrete and continuous spatial data. Even though spatial data warehousing is widely studied, it has not implemented yet on the Semantic Web.

Geospatial Semantic Web: The Open Geospatial Consortium – OGC pursue an important line of work for geospatial SW with GeoSPARQL [3] as a vocabulary to represent and query spatial data in RDF with an extension to SPARQL. Kyzirakos *et al.* presents a comprehensive survey in data models and query languages for linked geospatial data in [14], and propose a semantic geospatial data store - Strabon with an extensive query language – stSPARQL in [13], which is yet limited to a specific environment. LinkedGeoData is a significant contribution on interactively transforming OpenStreetMap[1] data to RDF data [19]. GeoKnow [15] is a more recent project with focus on linking geospatial data from heterogeneous sources.

[1] http://www.openstreetmap.org.

The studies shows that, SW and RDF technologies evolve to give better functionality and standards for spatial data representation and querying. It is also argued above that spatial data is very much needed for DW/OLAP applications. However modeling and querying of spatial DWs on the SW is not addressed in any of the above papers. There are recent efforts on creating an Extract-Load-Transform (ETL) framework from semantic data warehouses [7] and publishing/-converting open spatial data as Linked Open Data [2], which motivates modeling and querying spatial data warehouses on the Semantic Web. Spatial data requires specific treatment techniques, particular encoding, special functions and different manipulation methods, which should be considered during the design and modeling process. Current state of the art geospatial Semantic Web focuses on techniques for publishing, linking and querying spatial data however does not elaborate on analytical spatial queries for MD data. In order to address these issues we propose a generic and extensible metamodel based on the best practices of MD data publishing in RDF. Then we show how to create spatial analytical queries with SOLAP on MD data models. We base ourselves on existing works by extending the most recent version of the QB4OLAP vocabulary with spatial concepts. Furthermore, we introduce the new concept of SOLAP operators that navigate along spatial dynamic hierarchy levels and implement these analytical spatial queries in SPARQL.

3 Spatial and OLAP Operations

In this section we give define the spatial and spatial OLAP (SOLAP) operations.

3.1 Spatial Operations

In order to understand spatial operations, it is important to understand what is a *spatial object*. A spatial object is the data or information that identifies a real-world entity of geographic features, boundaries, places etc. Spatial objects can be represented in object/vector or image/raster mode. Database applications that can store spatial objects need to specify the spatial characteristics, encoded as specific information such as *geometry* data type which is the most common and supports planar or Euclidean (flat-earth) data. *Point, Line, and, Polygon* are the basic instantiable types of the geometry data type.

Geometries are associated with a *spatial reference system* (SRS) which describes the coordinate space in which the geometry is defined. There are several SRSs and each of them are identified with a *spatial reference system identifier* (SRID). The World Geodetic System (WGS) is the most well-known SRS and the latest version is called WGS84, which is also used in our use case.

Spatial data types have a set of operators that can function among applications. We grouped these operations into classes. Our classification is based on the common functionality of the operators. These classes are defined as follows:

Spatial Aggregation. The operators in the spatial aggregation, $\mathcal{S}_{agg}$ class aggregate two or more spatial objects. The result of these operators returns

a new composite spatial object. Union, Intersection, Buffer, ConvexHull, and, MBR - Minimum Bounding Rectangle are example operators of this class.

Topological Relation. The operators in the topological relation, $\mathcal{T}_{rel}$ class are commonly contained in the RCC8[2] and DE-9DIM[3] models. Topological relations are standardized by OGC as Boolean operators which specify how two spatial objects are related to each other with a set of spatial predicates for example: Intersects, Disjoint, Equals, Overlaps, Contains, Within, Touches, Covers, CoveredBy, and, Crosses.

Numeric Operation. The operators in the class of numeric operation, $\mathcal{N}_{op}$ take one or more spatial objects and return a numeric value. Perimeter, Area, # of Interior Rings, Distance, Haversine Distance, Nearest Neighbor (NN), and # of Geometries are some of the example operators of this class.

3.2 SOLAP Operations

OLAP operations emerge when executing complex queries over multidimensional (MD) data models. OLAP operations let us interpret data from different perspectives at different levels of detail. *Spatially extended multidimensional models* incorporate spatial data during the analysis and decision making process by revealing new patterns, which are difficult to discover otherwise. In connection with our definition of MD models in the first paragraph of Sect. 1, hereafter we enhance and describe the spatially extended MD data cube elements.

A spatially extended MD model contains both conventional and spatial dimensions. A *spatial dimension* is a dimension which includes at least one *spatial hierarchy*. Dimensions usually have more than one level which are represented through hierarchies and there is always a unique top level *All* with just one member. A hierarchy is a *spatial hierarchy* if it has at least one *spatial level* in which the application should store the spatial characteristics of the data, which is captured by it is geometry and can be recorded in the *spatial attributes* of the level. A *spatial fact* is a fact that relates several dimensions in which, two or more are spatial. For example, consider a "Sales" *spatial fact* cube, which has "Customer" and "Supplier" (company) as *spatial dimensions* with a *spatial hierarchy* as "Geography" that expands into (hierarchic) *spatial levels*; "City → State → Country → Continent → *All*" from the customer and supplier's location. All these *spatial levels* record the spatial characteristics *i.e.* with a *spatial attribute* of a city (center) as "point" coordinates. Measures in the cube express additional and essential information for each MD data cell which is not exhibited through the dimensions and level attributes. Typically, spatially extended MD models have *spatial measures* which are represented by a geometry *i.e.* point, polygon, etc.

[2] RCC8 – Region Connection Calculus describes regions in Euclidean space, or in a topological space by their possible relations to each other.

[3] DE-9DIM – Dimensionally Extended Nine-Intersection Model is a topological model that describes spatial relations of two geometries in two-dimensions.

Spatial OLAP operates on spatially extended MD models. SOLAP enhances the analytical capabilities of OLAP with the spatial information of the cube members. The term SOLAP used in [4] and their similar works as a visual platform, which is designed to analyze huge volumes of geo-referenced data by providing visualization of pivot tables, graphical displays and interactive maps. We define the term SOLAP concisely as a platform (and query language) independent high-level concept, which is applicable on any spatial multidimensional data. We explain and exemplify in the following how SOLAP operators are interpreted.

Each operator in SOLAP should include at least one *spatial condition* by using the aforementioned operators from the spatial operation classes defined in Sect. 3.1. Spatial operations in SOLAP create a dynamic interpretation of the cube members as a *dynamic spatial hierarchy or level*. These interpretations allow new perspectives to analyze the spatial MD data which cannot be accessed in a traditional MD model. For instance, the classical OLAP operator *roll-up* aggregates measures along a hierarchy to obtain data at a coarser granularity. In the spatial dimension schema Fig. 1, the (classical) roll-up relation, *Customer to City* is shown with black straight arrows. On the other hand, in SOLAP, a new "dynamic spatial hierarchy" is created on the fly to roll-up among spatial levels by a spatial condition (closest distance), which is given as *Customer to Closest-City (of the Supplier)*, shown with curved arrows in gray. The details of this operator in comparison with OLAP and SOLAP are given in the following example.

Example: Roll-up. The user wants to sum the total amount of the sales to customers up to the city level with the roll-up operator. The instance data for *Sales fact* is given in Table 1 and shown on the map in Fig. 2.

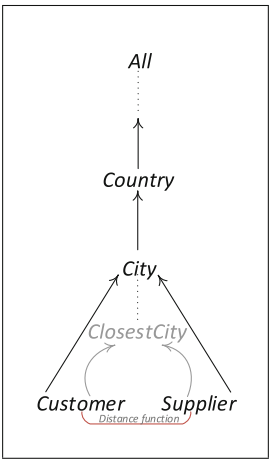

Fig. 1. S-Dim. Schema

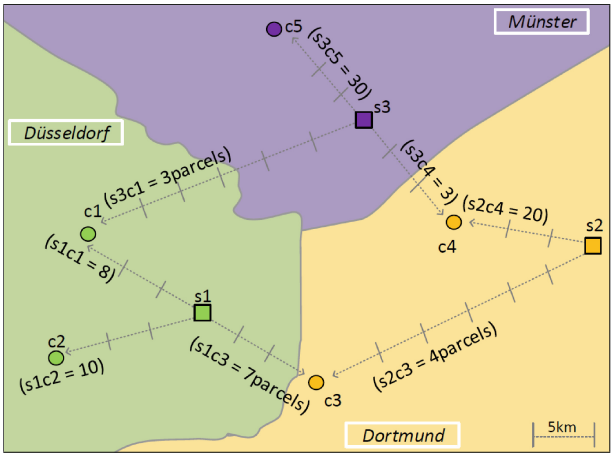

Fig. 2. Example map of sales (instance) data

Table 1. Sample (instance) data for sales

City	Customer	Supplier sales			Total sales
		s1	s2	s3	
Düsseldorf	c1	8pcs.	–	3pcs.	11pcs.
	c2	10pcs.	–	–	10pcs.
Dortmund	c3	7pcs.	4pcs.	–	11pcs.
	c4	–	20pcs.	3pcs.	23pcs.
Münster	c5	–	–	30pcs.	30pcs.

Table 2. Roll-up

City	Sales
Düsseldorf	21pcs.
Dortmund	34pcs.
Münster	30pcs.

Table 3. S-Roll-up

City	Sales
Düsseldorf	25pcs.
Dortmund	20pcs.
Münster	33pcs.

Table 4. Customer to Supplier Distance (km.s)

	Sup. City	**Düsseldorf**	**Dortmund**	**Münster**
Cust. City	**Sup.** / **Cust.**	s1	s2	s3
Düsseldorf	c1	15 km.s	45km.s	30 km.s
	c2	15 km.s	60 km.s	60 km.s
Dortmund	c3	*15 km.s*	30 km.s	45 km.s
	c4	45 km.s	15 km.s	*15 km.s*
Münster	c5	60 km.s	45 km.s	15 km.s

The amount of the sales are shown in parentheses along with the quantities of the sold parcels (from supplier to customer). The arrows on the map, between the supplier and customer locations are used to represent the distance. The summarized data for sale instances (Table 1) does not originally contain the records of the supplier – customer distance (as given in Table 4) which can lead to increase in the storage space. If there are no sales to customers from the corresponding suppliers, a dash (–) is used in Table 1. The syntax of the traditional roll-up operator is **ROLLUP(Sales, (Customer → City), SUM(SalesAmount))** which aggregates the *"total sales to customers up to city level"* (results in Table 2). Alternatively, the user may like to view the "*total sales to customers by city of the closest suppliers*", in which some customers can be closer to their suppliers from other cities, as emphasized in Table 4. This query is possible with traditional OLAP, if only Table 4 is recorded in the base data which requires extra storage space. For a better support and flexibility we define a spatial roll-up operator that aggregates the total sales along the dynamic spatial hierarchy, which is created based on a spatial condition (in this example, distance between customer and supplier locations). The syntax for s-roll-up is; **S-ROLLUP(Sales,[CLOSEST(Customer,Supplier)] → City',SUM(SalesAmount))**. The spatial condition transforms the customer→city hierarchy as a dynamic spatial hierarchy, which depends on the proximity of the suppliers that is calculated during runtime. The user has the flexibility to make analyses with different spatial operations in SOLAP. Spatial extensions of common OLAP operators (roll-up, drill-down, slice, and dice) are formally defined in Sect. 4.3.

4 Semantics of Spatial MD Data and OLAP Operations

In this section, we first present our approach on how to support spatial MD data in RDF by using the QB4SOLAP vocabulary. Afterwards, we define the general semantics of each SOLAP operator to be implemented in SPARQL as a proof of concept to the QB4SOLAP metamodel. The concepts introduced in this metamodel are an extension to the most recent QB4OLAP vocabulary [9].

Figure 3 shows the proposed and extended QB4OLAP vocabulary for the *cube schema* RDF triples. Capitalized terms in the figure represent RDF classes and non-capitalized terms represent RDF properties. Classes in external vocabularies are depicted in light gray background and font. RDF Cube (QB), QB4OLAP, QB4SOLAP classes are shown with white, light gray, dark gray backgrounds, respectively. Original QB terms are prefixed with `qb:`. QB4OLAP and QB4SOLAP classes and properties are prefixed with `qb4o:` and `qb4so:`. In order to represent spatial classes and properties an external prefix from OGC, `geo:` is used in the metamodel. Since QB4OLAP and QB4SOLAP are RDF-based multidimensional model schemas, we first define formally what an RDF triple is, and then discuss the basics of describing MD data using QB4OLAP and spatially enhanced MD data in QB4SOLAP.

An *RDF triple* t consists of three components; s is the subject, p is the predicate, and o is the object, which is defined as: *triple (s,p,o)* $\in t = (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}i)$ where the set of *IRIs* is $\mathcal{I}$, the set of *blank nodes* is $\mathcal{B}$, and the set of *literals* is $\mathcal{L}i$. Given an MD element x of the cube schema $\mathcal{CS}$, $\mathcal{CS}(x) \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}i)$ returns a set of triples $\mathcal{T}$, with IRIs, blank nodes and literals. The notation of the triples in the following definitions is given as (x `rdf:type ex:SomeProperty`). If the concepts are defined with a set of triples, after the first triple, we use a semicolon "`;`" to link predicates (p) and objects (o) with the subject (s) concept x. The blank nodes $\mathcal{B}$ are expressed as `_:` and nesting unlabeled blank nodes are abbreviated in square brackets "`[]`".

4.1 Defining MD Data in QB4OLAP

In order to explain the spatially enhanced MD models we first describe MD elements described in Sect. 1 with the RDF formalization in QB4OLAP vocabulary.

Cube Schema. A data structure definition (DSD) specifies the schema of a data set (*i.e.*, a cube, which is an instance of the class `qb:DataSet`). The DSD can be shared among different data sets. The DSD of a data set represents dimensions, levels, measures, and attributes with component properties. The DSD is defined through a conceptual MD cube schema $\mathcal{CS}$, which has a set of dimension types $\mathcal{D}$, a set of measures $\mathcal{M}$ and, with a fact type $\mathcal{F}$ as $\mathcal{CS} = (\mathcal{D}, \mathcal{M}, \mathcal{F})$. For example, a cube schema $\mathcal{CS}$ can be used to define a physical structure of a company's sales data to be represented as a MD data cube. We define the cube schema elements in the following definitions.

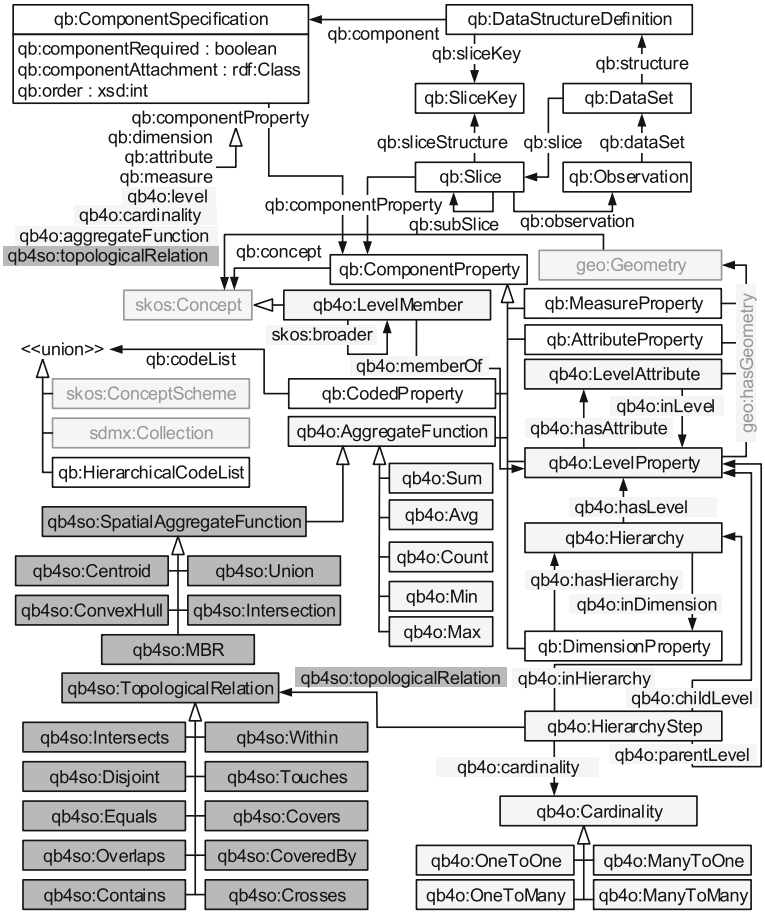

Fig. 3. QB4SOLAP vocabulary meta-model

Attributes. An attribute $a \in \mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ has a domain $\langle a : dom \rangle$ in the cube schema $\mathcal{CS}$ with a set of triples $t_a \in \mathcal{T}$ where t_a is encoded as (a `rdf:type qb:AttributeProperty; rdfs:domain xsd:Schema`)

The domain of the attribute is given with the property `rdfs:domain`[4] from the corresponding schema and `rdfs:range` defines what values the property can take *i.e.*; *integer*, *decimal*, etc. from the given, *xsd:Schema* elements[5]. Attributes are the finest granular elements of the cube, which exists in levels to describe the characteristics of level members *e.g.*, customer level attributes could be as; name, id, address, etc.

Levels. A level $l \in \mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ consists of a set of attributes $\mathcal{A}_l$, which is defined by a schema $l(a_1 : dom_1, \ldots, a_n : dom_n)$, where l is the level and each attribute a is defined over the domain dom. For each level $l \in \mathcal{L}$

[4] RDF Schema http://www.w3.org/TR/rdf-schema/.
[5] XML Schema http://www.w3.org/TR/xmlschema11-1/.

in the cube schema $\mathcal{CS}$, there is a set of triples $t_l \in \mathcal{T}$ which is encoded as (l `rdf:type qb4o:LevelProperty; qb4o:hasAttribute` a). Relevant levels for customer data include; customer level, city level, country level, etc.

Hierarchies. A hierarchy $h \in \mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ in the cube schema $\mathcal{CS}$, is defined with a set of triples $t_h \in \mathcal{T}$, and encoded as (h `rdf:type qb4o:HierarchyProperty; qb4o:hasLevel` l; `qb4o:inDimension` $\mathcal{D}$).

Each hierarchy $h \in \mathcal{H}$ is defined as $h = (\mathcal{L}_h, \mathcal{R}_h)$; with a set of $\mathcal{L}_h$ (hierarchy) levels, which is a subset of the set $\mathcal{L}_d$ levels of the dimension $\mathcal{D}$ where $\mathcal{L}_h \subseteq \mathcal{L}_d \in \mathcal{D}$. $\mathcal{L}_d$ contains the initial base level of the dimension in addition to hierarchy levels L_h. For example, customer–location hierarchy can be defined by the levels; customer, city, country, etc. where customer is the base level and contained only in $\mathcal{L}_d$.

Due to the nature of the hierarchies, a hierarchy entails a roll-up relation $\mathcal{R}_h$ between its levels, $\mathcal{R}_h = (\mathcal{L}_c, \mathcal{L}_p, card)$ where $\mathcal{L}_c$ and $\mathcal{L}_p$ are respectively child and parent levels, where the lower level is called child and higher level is called parent. Cardinality $card \in \{1-1, 1-n, n-1, n-n\}$ describes the minimum and maximum number of members in one level that can be related to a member in another level, *e.g.*, $\mathcal{R}_h = (city, country, many - to - one)$ shows that the roll-up relation between the child level city to parent level country is many-to-one, which means that each country can have many cities. In order to represent cardinalities between the child and the parent levels, blank nodes are created as hierarchy steps, $:\_h_{hs} \in \mathcal{B}$. Hierarchy steps relate the levels of the hierarchy from a bottom (child) level to an upper (parent) level, which is defined with a set of triples $t_{hs} \in \mathcal{T}$ and encoded as ($:\_h_{hs}$ `rdf:type qb4o:HierarchyStep; qb4o:childLevel` lh_c; `qb4o:parentLevel` lh_p; `qb4o:cardinality` $card$) where $lh_c \in \mathcal{L}_c$, $lh_p \in \mathcal{L}_p$ and $card \in \{1-1, 1-n, n-1, n-n\}$.

Dimensions. An n-dimensional cube schema has a set of dimensions $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$. And each $d \in \mathcal{D}$ is defined as a tuple $d = (\mathcal{L}, \mathcal{H})$; with a set of $\mathcal{L}_d$ levels, organized into $\mathcal{H}_d$ hierarchies. Dimensions, inherently have all the levels from the hierarchies they have, and an initial base level. For each dimension $d \in \mathcal{D}$, in the cube schema $\mathcal{CS}$, there is a set of triples $t_d \in \mathcal{T}$, which is encoded as (d `rdf:type qb:DimensionProperty; qb4o:hasHierarchy` h). For example, customerDim is a dimension with a location and a customer type hierarchy where location expands to levels of customer's location (e.g., city, country, etc.) and customer type expands to levels of customer's type (e.g.,profession, branch, etc.).

Measures. A measure $m \in \mathcal{M} = \{m_1, m_2, \ldots, m_n\}$, is a property, which is associated to the facts. Measures are given in the cube schema $\mathcal{CS}$ with a set of triples $t_m \in \mathcal{T}$, which is encoded as (m`rdf:type qb:MeasureProperty; rdfs:subPropertyOf sdmx-measure:obsValue; rdfs:domain xsd:Schema`).

Measures are defined with a sub-property from the Statistical Data and Metadata Exchange - ($sdmx$) definitions, `sdmx-measure:obsValue` which is the value of a particular variable for a particular observation[6]. Similarly to

[6] http://sdmx.org/.

the attributes `rdfs:domain` specifies the schema of the measure property and, `rdfs:range` defines what values the property can take *i.e.*; *integer*, *decimal*, *etc.* in the instances. For example, quantity and price are measures of a fact (e.g., sales) where the instance values can be given respectively, in the form: "13" ^^xsd:positiveInteger and "42.40"^^xsd:decimal. Measures are associated with observations (facts) and related to dimension levels in the DSD as explained in the following.

Facts. A fact $f \in \mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is related to values of dimensions and measures. The relation is described in *components* in the schema level of the facts cube definition, by a set of triples $t_f \in \mathcal{T}$, which is encoded as ($\mathcal{F}$ `rdf:type qb:DataStructureDefinition; qb:component[qb4o:level` l`; qb4o:cardinality` $card$`]; qb:component[qb:measure` m`; qb4o:aggregate Function` BIF`]`). Cardinality, $card \in \{1-1, 1-n, n-1, n-n\}$ represents the cardinality of the relationship between facts and level members. The specification of the aggregate functions for measures is required in the definition of the cube schema. Standard way of representing typical aggregate functions is defined by QB4OLAP namely built-in functions such as; $BIF \in \{Sum, Avg, Count, Min, Max\}$. For example, a fact schema $\mathcal{F}$ can be sales of a company which has associated dimensions and measures defined as components respectively e.g. product and price.

Finally, the facts $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ are given on the instance level where each fact f has a unique IRI $\mathcal{I}$, which are observations. This is encoded as (f `rdf:type qb:Observation`). An example of a fact instance f with it's relation to measure values and dimension levels is a "sale" transacted to customer "John" (value of the dimension level), for a product "chocalate" (value of another dimension level), which has a unit price of "29.99" (value of a measure) euros, and quantity of "20" (value of another measure) boxes. Cardinality of the dimension level customer and fact member is many–to–one where several sales can be transacted to the same customer (i.e. John). Specification of the aggregate function for measure unit price is "average" while quantity can be specified as "sum".

We gave the cube schema $\mathcal{CS} = (\mathcal{D}, \mathcal{M}, \mathcal{F})$ members above where dimensions $d \in \mathcal{D}$ are defined as a tuple of dimension levels $\mathcal{L}_d$ and hierarchies $\mathcal{H}$, $d = (\mathcal{L}_d, \mathcal{H})$, and a hierarchy $h \in \mathcal{H}$ is defined with hierarchy levels $\mathcal{L}_h$ such that $h = (\mathcal{L}_h)$, where $\mathcal{L}_h \subseteq \mathcal{L}_d$, and a level l contains attributes $\mathcal{A}_l$ as $l = (\mathcal{A}_l)$.

4.2 Defining Spatially Enhanced MD Data in QB4SOLAP

QB4SOLAP adds a new concept to the metamodel, which is `geo:Geometry` class from OGC schemas[7]. We define the QB4SOLAP extension to the cube schema in the description of the following spatial MD data elements, which are explained in Sect. 3.2.

Spatial Attributes. Each attribute is defined over a domain (Sect. 4.1). Every attribute with geometry domain ($a : dom_g \in \mathcal{A}$) is a member of `geo:Geometry`

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

class and they are called spatial attributes a_s, which are defined in the cube schema $\mathcal{CS}$ by a set of triples $t_{ag} \in \mathcal{T}$ and encoded as (a_s `rdf:type qb:AttributeProperty; rdfs:domain geo:Geometry`). The type (point, polygon, line, *etc.*) of the each spatial attribute is assigned with `rdfs:range` predicate in the instances. For example, a spatial attribute can be the "capital city" of a country which is represented through a point geometry.

Spatial Levels. Each spatial level $l_s \in \mathcal{L}$ is defined with a set of triples $t_{ls} \in \mathcal{T}$ in the cube schema $\mathcal{CS}$, and encoded as (l_s `rdf:type qb4o:LevelProperty; qb4o:hasAttribute` a, a_s`; geo:hasGeometry geo:Geometry`). Spatial levels must be a member of `geo:Geometry` class and might have spatial attributes. For example country level is a spatial level which has a polygon geometry and might also record geometry of the capital city in the level attributes as a point type.

Spatial Hierarchies. Each hierarchy $h_s \in \mathcal{H}$ is spatial, if it relates two or more spatial levels l_s. Spatial hierarchy step defines the relation between the spatial levels with a roll-up relation as in conventional hierarchy steps (Sect. 4.1). QB4SOLAP introduces topological relations, $\mathcal{T}_{rel}$ (Sect. 3.1) besides cardinalities in the roll-up relation which is encoded as $\mathcal{R} = (\mathcal{L}_c, \mathcal{L}_p, card, \mathcal{T}_{rel})$ for the spatial hierarchy steps.

Let $t_{shs} \in \mathcal{T}$ a set of triples to represent a hierarchy step for spatial levels in hierarchies, which is given with a blank node $:\_sh_{hs} \in \mathcal{B}$ and encoded as ($:\_sh_{hs}$ `rdf:type qb4o:HierarchyStep; qb4o:childLevel` slh_{ci}; `qb4o:parentLevel` slh_{pi}; `qb4o:cardinality` $card$; `qb4so:hasTopologicalRelation` $\mathcal{T}_{rel}$) where $slh_{ci} \in \mathcal{L}_c$, $slh_{pi} \in \mathcal{L}_p$. For example, a spatial hierarchy is "geography" which should have spatial levels (e.g. customer, city, country, and continent) with the roll-up relation $\mathcal{R}_h = (city, country, many - to - one, within)$, which also specifies that child level city is "within" the parent level country, in addition to the hierarchy steps from Sect. 4.1.

Spatial Dimensions. Dimensions are identified as spatial if only they have at least one spatial hierarchy. More than one dimension can share the same spatial hierarchy and the spatial levels, which belongs to that hierarchy. QB4SOLAP uses the same schema definitions of the dimensions as in Sect. 4.1. For example, a spatial dimension is customer dimension, which has a spatial hierarchy geography.

Spatial Measures. Each spatial measure $m_s \in \mathcal{M}$ is defined in the cube schema $\mathcal{CS}$ by a set of triples $t_{ms} \in \mathcal{T}$ and encoded as (m_s `rdf:type qb:MeasureProperty; rdfs:subPropertyOf sdmx-measure:obsValue; rdfs:domain geo:Geometry`). The class of the numeric value is given with the property `rdfs:domain` and `rdfs:range` assigns the values from `geo:Geometry` class, *i.e.*, *point*, *polygon*, *etc.* at the instance level.

Spatial measures are represented by a geometry thus they use a different schema than conventional (numeric) measures. The schemas for spatial measures

have common geometry serialization standards[8] that are used in OGC schemas. For example a spatial measure is coordinates of an accident location, which is given as a point geometry type and associated to an observation fact of accidents.

Spatial Facts. Spatial facts $\mathcal{F}_s$ relates several dimensions of which two or more are spatial. If there are several spatial dimension levels (l_s), related to the fact, topological relations $\mathcal{T}_{rel}$ (Sect. 3.1) between the spatial members of the fact instance may be required which is not necessarily imposed for all the spatial fact cubes. Ideally a spatial fact cube has spatial measures (m_s), as its members which makes it possible to aggregate along spatial measures with the spatial aggregation functions $\mathcal{S}_{agg}$ (Sect. 3.1). Representation of a complete spatial fact cube at the schema level in RDF is given by a set of triples $t_{fs} \in \mathcal{T}$, and encoded as ($\mathcal{F}_s$ `a qb:DataStructureDefinition; qb:component [qb4o:level`l_s`; rdfs:subPropertyOf sdmx-dimension:refArea; qb4o:cardinality` $card$`; qb4so:TopologicalRelation` $\mathcal{T}_{rel}$`]; qb: component[qb:measure` m_s`,sdmx-measure:obsValue;qb4o:aggregateFunction` BIF'`]`). QB4SOLAP extends the built-in functions of QB4OLAP with spatial aggregation functions as $BIF' = BIF \cup \mathcal{S}_{agg}$ which is added with a class `qb4so:SpatialAggregateFunction` to the metamodel in Fig. 3. An example of a spatial fact instance f_s with it's relation to measure values and dimension levels is a traffic "accident" incident occured on a highway "E–45" (value of the highway spatial dimension level) with coordinate points of the location "57.013, 9.939" (value of the location spatial measure). Cardinality of the dimension level highway and fact member is many–to–one where several accidents might take place in the same highway. Specification of the spatial aggregate function for spatial measure location (coordinate points) can be specified as "convex hull" area of the accident locations.

4.3 SOLAP Operators

The proposed vocabulary QB4SOLAP allows publishing spatially enhanced multidimensional RDF data which allows us to query with SOLAP operations. Subqueries and aggregation functions in SPARQL 1.1[9] make it easily possible to operate with OLAP queries on multidimensional RDF data. Moreover, spatially enhanced RDF stores, provide functions to an extent for querying with topological relations and spatial numeric operations. In the following, we define common OLAP operators with spatial conditions in order to formalize spatial OLAP query classes. Spatial conditions can be selected from a range of operation classes that can be applied on spatial data types (Sect. 3.1). Let $\mathcal{S}$ be any spatial operation where $\mathcal{S} = (\mathcal{S}_{agg} \cup \mathcal{T}_{rel} \cup \mathcal{N}_{op})$ to represent a spatial condition in a SOLAP operation. The following OLAP operators are given with a spatial extension to the well-known OLAP operators defined over cubes based in Cube Algebra operators [5].

[8] The Well Known Text (WKT) serialization aligns the geometry types with ISO 19125 Simple Features [ISO 19125-1], and the GML serialization aligns the geometry types with [ISO 19107] Spatial Schema.

[9] http://www.w3.org/TR/sparql11-query/.

S–Roll–up. Given a cube $\mathcal{C}$, a dimension $d \in \mathcal{C}$, and an upper dimension level $l_u \in d$, such that $l\,\langle a : dom_g\rangle \rightarrow^* l_u$, where $l\,\langle a : dom_g\rangle$ represents the level in dimension d with attributes (a_s) whose domain is a geometry type. Let $\mathcal{R}_s$ be the spatial roll-up relation which comprises $\mathcal{S}$ and traditional roll-up relation $\mathcal{R}$ such that $\mathcal{R}_s = \mathcal{S}(d, l\,\langle a : dom_g\rangle) \cup \mathcal{R}(\mathcal{C}, d, l_u) \rightarrow \mathcal{C}'$.

Initially, in the semantics of *S–Roll–up* above, spatial constraint $\mathcal{S}$ is applied over a dimension d on the spatial attributes a_s along levels l. As a result of the roll–up relation $\mathcal{R}$, the measures are aggregated up to level l_u along d which returns a new cube $\mathcal{C}'$. Note that applying $\mathcal{S}$, on spatial level attributes a_s of dimension $\mathcal{D}$, operates on the hierarchy step $l \rightarrow l_u$ with a dynamic spatial hierarchy (Ref. Sect. 3.2). For example, the query "total sales to customers by city of the closest suppliers" implies a S-Roll-up operator.

S–Drill–down. Analogously, *S–Drill–down* is an inverse operation of *S–Roll–up*, which disaggregates previously summarized data down to a child level. For example, the query "average sales of the employees from the biggest city in its country" implies a S-Drill-down operator by disaggregating data from (parent) country level to (child) city level by imposing also a spatial condition (area from $\mathcal{N}_{op}$ to choose the biggest city).

S–Slice. Given a cube $\mathcal{C}$ with n dimensions $\mathcal{D} = \{d_1, d_2, \ldots, d_n\} \in \mathcal{C}$, let $\mathcal{S}'$ be the traditional slice operator which removes a dimension d from the cube $\mathcal{C}$. And let $\mathcal{S}_s$ be the spatial slice operator, which comprises $\mathcal{S}$, the spatial function to fix a single value in the level $\mathcal{L} = \{l_1, l_2, \ldots, l_n\} \in d$ defined as follows; $\mathcal{S}_s = \mathcal{S}'(\mathcal{C}, d) \cup \mathcal{S}(d, l\,\langle a : dom_g\rangle) \rightarrow \mathcal{C}'$.

Note that the spatial function is applied on the spatial attributes of the selected level, measures are aggregated along dimension d up to level *All*. The result returns a new cube $\mathcal{C}'$ with $n-1$ dimensions $\mathcal{D}' = \{d_1, d_2, \ldots, d_{n-1}\} \in \mathcal{C}'$. For example, the query "total sales to the customers located in the city within a 10 km buffer area from a given point" implies a S-Slice operator, which dynamically defines the city level by (fixing) a specified buffer area around a given custom point in the city.

S–Dice. Dice operation is analogous to relational algebra - *R selection*; $\sigma_\phi(R)$, instead the argument is a cube $\mathcal{C}$; $\sigma_\phi(\mathcal{C})$. In SOLAP dice is not a select operation rather a nested "select" and a "spatial filter" operation. S-Dice $\mathcal{D}_s$ keeps the cells of a cube $\mathcal{C}$ that satisfy a spatial Boolean $\mathcal{S}(\phi)$ condition over spatial dimension levels, attributes and measures which is defined as; $\mathcal{D}_s = (\mathcal{C}, \mathcal{S}(\phi)) \rightarrow \mathcal{C}'$ where $\mathcal{S}(\phi) = \mathcal{S}(\sigma_{a\phi b}(\mathcal{C})) \vee \mathcal{S}(\sigma_{a\phi v}(\mathcal{C}))$ and a, b are spatial levels (l_s), geometry attributes $(a : dom_g)$ or measures (m, m_s) while v is a constant value and the result returns a sub-cube $\mathcal{C}' \subset \mathcal{C}$. For example, the query "total sales to the customers which are located less than 5 Km from their city" implies a S-Dice operator.

In this paper, we focus on direct querying of single data cubes. The integration of several cubes through *S-Drill-across* or set-oriented operations such as *Union, Intersection, and Difference*[5] is out of scope and remained as future work. The actual use of these query classes in SPARQL with the instance data is given in Sect. 6.

5 Use Case Scenario: GeoNorthwind Data Warehouse

Figure 4 consists of the conceptual schema of the GeoNorthwind DW use case. GeoNorthwind DW has synthetic data about companies and their sales, however it is well suited for representing MD data modeling concepts due to its rich dimensions and hierarchies. It is a good proof of concept use case to show how to implement spatial data cube concepts on the SW. We show next how to express the conceptual schema of GeoNorthwind in QB4SOLAP.

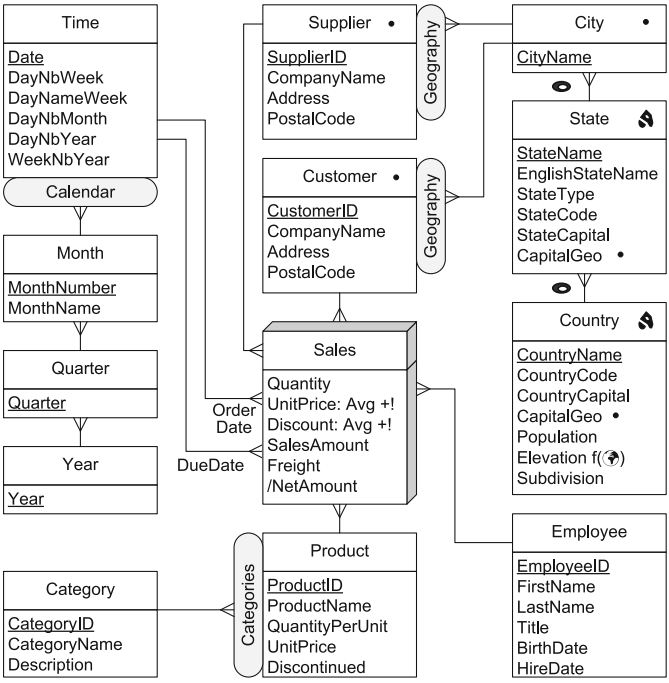

Fig. 4. Conceptual MD schema of the GeoNorthwind DW

In the use case, measures are given in the Sales cube. All measures are conventional. The members of the GeoNorthwind DW are given with `gnw:` prefix. The underlying syntax for RDF representation is given in Turtle[10] syntax in the boxes. An example of a measure in the cube schema is given in the following as defined in Sect. 4.1.

```
gnw:quantity a rdf:Property, qb:MeasureProperty;
rdfs:subPropertyOf sdmx-measure:obsValue;rdfs:range xsd:integer.
```

[10] http://www.w3.org/TR/turtle/.

In the following, a spatial attribute of a spatial level `gnw:state` is given along with the level and attribute properties. Spatial level has a geometry as `gnw:statePolygon` independently having a spatial attribute `gnw:capitalGeo`. Each spatial attribute in the schema is defined separately by using common RDF and standard spatial schemas[11] to represent their domain and data type as described in Sect. 4.2.

```
gnw:state a qb4o:LevelProperty; qb4o:hasAttribute gnw:stateName,
   gnw:stateType, gnw:stateCapital, gnw:capitalGeo;
   geo:hasGeometry gnw:statePolygon.
gnw:captialGeo a  qb:AttributeProperty;
  rdfs:domain geo:Geometry; rdfs:range geo:Point, geo:wktLiteral, virtrdf:Geometry.
```

In the next listing, an example of a spatial dimension from the use case data is `gnw:customerDim`, which is given with its spatial hiearchy `gnw:geography` (Sects. 4.1 and 4.2). The spatial hierarchy is organized into levels (*i.e.* city, state, country *etc.*) where `qb4o:hasLevel` predicate indicates the levels that compose the hierarchy. Each hierarchy in dimensions is represented with `qb4o:inDimension` predicate, referring to the dimension(s) it belongs to. The levels given in the dimension hierarchy are all spatial, and the sample representation of a spatial level is given above.

```
gnw:customerDim a rdf:Property, qb:DimensionProperty;
   qb4o:hasHierarchy gnw:geography.
gnw:geography a qb4o:HierarchyProperty; qb4o:hasLevel gnw:city,
   gnw:state, gnw:region, gnw:country, gnw:continent;
   qb4o:inDimension gnw:customerDim, gnw:supplierDim.
```

Each hierarchy step is added to the schema as a blank node (`_:hsi`) by `qb4o:HierarchyStep` property, in which the cardinality and topological relationships are represented in between the child and parent levels as follows;

```
_:hs1 a qb4o:HierarchyStep; qb4o:inHierarchy gnw:geography;
   qb4o:childLevel gnw:customer, gnw:supplier;
   qb4o:parentLevel gnw:city; qb4o:cardinality qb4o:ManyToOne;
   qb4so:hasTopologicalRelation qb4so:Within.
```

The components of the facts are described at the schema level in the cube definition. The dimension level for `gnw:customer` is given with `sdmx-dimenson:refArea` property, which indicates the spatial characteristic of the dimension. Measures require the specification of the aggregate functions in the cube definition. As there are only numeric measures in the use case data,

[11] For our tests we used Virtuoso Universal Server and `virtrdf:Geometry` is a special RDF typed literal which is used for geometry objects in Virtuoso. Normally, WGS84 (EPSG:4326) is the SRID of any such geometry.

aggregate function for the sample measure `gnw:quantity` is given as `qb4o:sum`. The general overview of the cube schema $\mathcal{CS}$ which is given with the related components as follows:

```
### Cube definition ###
gnw:GeoNorthwind rdf:type qb:DataStructureDefinition;
### Lowest level for each dimension in the cube ###
qb:component [qb4o:level gnw:customer, sdmx-dimension:refArea;
qb4o:cardinality qb4o:ManyToOne].
### Measures in the Cube ###
qb:component [qb:measure gnw:quantity; qb4o:aggregateFunction qb4o:sum].
```

A spatial fact cube may contain spatial measure components besides spatial dimension according to QB4SOLAP. The implementation scope of this work covers only spatial facts, with spatial dimension and numerical measure components.

6 Querying the GeoNorthwind DW in SPARQL

We show next how some of the spatial OLAP queries from Sect. 4.3 can be expressed in SPARQL[12].

Query 1 (S-Roll-Up): Total sales to customers by city of the closest suppliers.

```
SELECT ?city (SUM(?sales) AS ?totalSales)
WHERE {?o a qb:Observation; gnw:customerID ?cust;
        gnw:supplierID ?sup; gnw:salesAmount ?sales.
        ?cust qb4o:inLevel gnw:customer;gnw:customerGeo ?custGeo;
        gnw:customerName ?custName; skos:broader ?city.
        ?city qb4o:inLevel gnw:city.?sup gnw:supplierGeo ?supGeo.
#Inner Select:Distance to the closest supplier of the customer
  {SELECT ?cust1 (MIN(?distance) AS ?minDistance)
  WHERE{?o a qb:Observation; gnw:customerID ?cust1;
        gnw:supplierID ?sup1. ?sup1 gnw:supplierGeo ?sup1Geo.
        ?cust1 gnw:customerGeo ?cust1Geo.
  BIND (bif:st_distance( ?cust1Geo, ?sup1Geo ) AS ?distance)}
  GROUP BY ?cust1 }
  FILTER (?cust = ?cust1 && bif:st_distance(?custGeo, ?supGeo)=
  ?minDistance)} GROUP BY ?city ORDER BY ?totalSales
```

The query above shows the spatial roll-up operation example from Sect. 3.2 with the actual use case data. We have explained the semantics of s-roll-up operator in Sect. 4.3. The inner select verifies the spatial condition in order to find the closest distance to suppliers from the customers. The outer select prepares the traditional roll up of the total sales from customer (child) level to the city (parent) level. Filter on customer and supplier distance creates the aforementioned dynamic spatial hierarchy based on the proximity of the suppliers.

Query 2 (S-Slice): Total sales to the customers located in the city within a 10 km buffer area from a given point.

[12] SPARQL endpoint is available at: http://extbi.ulb.ac.be:8890/sparql.

```
SELECT ?custName ?cityName (SUM(?sales) AS ?totalSales)
WHERE {?o rdf:type qb:Observation; gnw:customerID ?cust;
   gnw:salesAmount ?sales. ?cust gnw:customerName ?custName;
   skos:broader ?city. ?city gnw:cityGeo ?cityGeo;
        gnw:cityName ?cityName.
FILTER(bif:st_within(?cityGeo, bif:st_point(2.3522,48.856),10))}
GROUP BY ?custName ?cityName ORDER BY ?custName
```

The semantics of the above (s-slice) operator is given in Sect. 4.3. Traditional slice operator removes a dimension, by fixing a single value in a level of dimension with a given fixed value (*i.e.* CityName = "Paris"). On the other hand, s-slice dynamically defines the city level, by a specified buffer area around a given custom point in the city. Thus, s-slice removes the dimension customer and its instance in city Paris, but only the customer instances within 10 km buffer area of the desired location. The project content with corresponding data sets and full query examples are available at: http://extbi.cs.aau.dk/QB4SOLAP/index.php.

7 Conclusion and Future Work

In this paper, we studied the modeling issues of spatially enhanced MD data cubes in RDF, defined the concept of SOLAP operators and implemented them in SPARQL. We showed that in order to model spatial DWs on the SW, an extended representation of MD cube elements was required. We based our representation on the most recent QB4OLAP vocabulary and make it viable for spatially enhanced MD data models through the new QB4SOLAP metamodel. This allows users to publish spatial MD data in RDF format. Then, we define well-known OLAP operations on data cubes with spatial conditions, in order to introduce spatial OLAP query classes and formally define their semantics. Subsequently, we present a use case and implement real-world SOLAP queries in SPARQL, to validate our approach.

Future work will be conducted in two areas: (1) defining complete formal techniques and algorithms for generating SOLAP queries in SPARQL based on a high-level MD Cube Algebra as in [5], and extending the coverage of SOLAP operations over multiple RDF cubes in SPARQL, *i.e.*, to support *S-Drill-Across*; (2) implement our QB4SOLAP approach on a more complex case study with spatial measures and facts which can support spatial aggregation (*S-Aggregation*) operator over measures with geometries. In order to support this *S-Aggregation* operator in SPARQL we will also investigate on creating user-defined SPARQL functions.

Acknowledgment. This research was 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., Berlanga Llavori, R., Nebot, V., Aramburu, M., Simitsis, A.: Using semantic web technologies for exploratory OLAP: a survey. TKDE **99**, 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)
3. Battle, R., Kolas, D.: GeoSPARQL: enabling a geospatial SW. Seman. Web **3**(4), 355–370 (2012)
4. Bimonte, S., Johany, F., Lardon, S.: A first framework for mutually enhancing chorem and spatial OLAP systems. In: DATA (2015)
5. Ciferri, C., Gómez, L., Schneider, M., Vaisman, A.A., Zimányi, E.: Cube algebra: a generic user-centric model and query language for OLAP cubes. IJDWM **9**(2), 39–65 (2013)
6. Cyganiak, R., Reynolds, D., Tennison, J.: The RDF Data Cube Vocabulary. W3C (2014)
7. Deb Nath, R.P., Hose, K., Pedersen, T.B.: Towards a programmable semantic extract-transform-load framework for semantic data warehouses. In: DOLAP (2015)
8. Diamantini, C., Potena, D.: Semantic enrichment of strategic datacubes. In: DOLAP (2008)
9. 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)
10. Gómez, L.I., Gómez, S.A., Vaisman, A.A.: A generic data model and query language for spatiotemporal OLAP cube analysis. In: EDBT (2012)
11. Han, J., Stefanovic, N., Koperski, K.: Selective materialization: an efficient method for spatial data cube construction. In: Wu, X., Kotagiri, R., Korb, K.B. (eds.) PAKDD 1998. LNCS, vol. 1394, pp. 144–158. Springer, Heidelberg (1998)
12. Kämpgen, B., O'Riain, S., Harth, A.: Interacting with statistical linked data via OLAP operations. In: Simperl, E., Norton, B., Mladenic, D., Valle, E.D., Fundulaki, I., Passant, A., Troncy, R. (eds.) ESWC 2012. LNCS, vol. 7540, pp. 87–101. Springer, Heidelberg (2012)
13. Koubarakis, M., Karpathiotakis, M., Kyzirakos, K., Nikolaou, C., Sioutis, M.: Data models and query languages for linked geospatial data. In: Eiter, T., Krennwallner, T. (eds.) Reasoning Web 2012. LNCS, vol. 7487, pp. 290–328. Springer, Heidelberg (2012)
14. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Strabon: a semantic geospatial DBMS. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 295–311. Springer, Heidelberg (2012)
15. Le Grange, J.J., Lehmann, J., Athanasiou, S., Rojas, A.G., et al.: The GeoKnow generator: managing geospatial data in the linked data web. In: Linking Geospatial Data (2014)
16. Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications. Springer, Heidelberg (2008)
17. Nebot, V., Berlanga, R., Pérez, J.M., Aramburu, M.J., Pedersen, T.B.: Multidimensional integrated ontologies: a framework for designing semantic data warehouses. In: Spaccapietra, S., Zimányi, E., Song, I.-Y. (eds.) Journal on Data Semantics XIII. LNCS, vol. 5530, pp. 1–36. Springer, Heidelberg (2009)

18. Revesz, P.: Introduction to Databases: From Biological to Spatio-Temporal. Springer, Heidelberg (2010)
19. Stadler, C., Lehmann, J., Hffner, K., Auer, S.: Linkedgeodata: a core for a web of spatial open data. Semant. Web **3**(4), 333–354 (2012)
20. Vaisman, A.A., Zimányi, E.: A multidimensional model representing continuous fields in spatial data warehouses. In: ACM SIGSPATIAL (2009)