GeoSemOLAP

Geospatial OLAP on the Semantic Web Made Easy

Gur, Nurefsan; Nielsen, Jacob; Hose, Katja; Pedersen, Torben Bach

Published in:
Proceedings of the 26th International Conference on World Wide Web Companion

DOI (link to publication from Publisher):
10.1145/3041021.3054731

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

GeoSemOLAP: Geospatial OLAP on the Semantic Web Made Easy

Nurefşan Gür, Jacob Nielsen, Katja Hose, Torben Bach Pedersen
Department of Computer Science, Aalborg University, Aalborg, Denmark
{nurefsan@cs, janiel13@student, khose@cs, tbp@cs}.aau.dk

ABSTRACT

The proliferation of spatial data and the publication of multidimensional (MD) data on the Semantic Web (SW) has led to new opportunities for On-Line Analytical Processing (SOLAP) over spatial data using SPARQL. However, formulating such queries results in verbose statements and can easily become very difficult for inexperienced users. Hence, we have developed GeoSemOLAP to enable users without detailed knowledge of RDF and SPARQL to query the SW with SOLAP. GeoSemOLAP generates SPARQL queries based on high-level SOLAP operators and allows the user to interactively formulate queries using a graphical interface with interactive maps.

Keywords

Multidimensional RDF Data; Spatial OLAP; SPARQL

1. INTRODUCTION

The data that is currently available on the Semantic Web (SW) has evolved from being simple, mostly alphanumeric, to also include more complex data such as spatial data [6]. Although spatial data is common on the SW, it remains difficult to utilize and analyze because spatial data requires special techniques for encoding it in RDF and evaluating spatial functions, which are often not supported by standard triple stores. On the other hand, the support of spatial data is more common in the area of relational databases, where spatial data warehouses are typically based on a multidimensional (MD) relational model involving spatial dimensions. Efficiently processing spatial data in this context is typically realized by Online Analytical Processing (OLAP) extended to support spatial analyses (SOLAP) [2].

However, with the growing popularity of the Linked Open Data (LOD) movement in the public sector, more and more spatial datasets from governmental domains [1] are becoming available on the SW. The availability of such datasets has led to novel opportunities for decision makers to analyze the growing public data with analytical data warehouse queries on SW data [4]. However, these potential users are only in very rare cases sufficiently familiar with the underlying technologies that are necessary to actually perform the desired analyses.

WWW 2017 Companion, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3041021.3054731

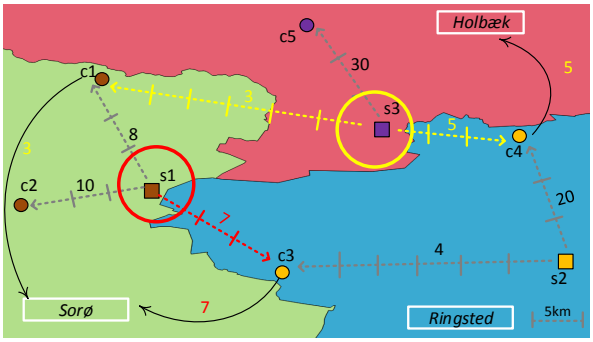


Figure 1: Example map of sales data

Let us consider an example scenario: Fig. 1 shows a map highlighting the amount of sales between customers *(c1, c2, . . . , c5)* and suppliers *(s1, s2, s3)* in three Danish cities *(Sorø, Holbæk, Ringsted)*. Let us assume that an analyst wants to obtain the total sales of customers grouped by cities of their closest supplier – Query 1 shows the corresponding example query formulated in SPARQL. This means that, first, for each customer we need to determine the closest supplier and the city in which the supplier is located. Then, we create one group for each of these cities and compute the total sales per city.

For each sales event, the dataset contains information about the involved customer and supplier; and for each customer and supplier the dataset contains the city they are located in. As the dataset does not contain any information about the distances between customers and suppliers, we have to use a *spatial function* (distance in this example) and evaluate it during runtime.

Based on the obtained information, we can aggregate the results as described above. Technically, this corresponds to a SOLAP operator: *s-roll-up* [2, 5].

As we can see in Query 1, formulating a roll-up to a higher level, e.g., from sales by supplier to sales by city, in a SPARQL query involves several triple patterns. Extending such a query with spatial aspects (as necessary for s-roll-up) requires even more triple patterns and spatial functions (such as distance), which can easily become overwhelming for inexperienced users.

The fact that data warehouse (DW) queries typically involve nesting of (S)OLAP operators, for example *(s-roll-up(s-slice(s-dice(DW))))*, makes it almost impossible for non-experts to formulate such queries with SPARQL.

Hence, we have developed *GeoSemOLAP*, a framework with an easy-to-use graphical interface that allows non-experts to query spatial semantic data warehouses using high-level SOLAP operators [5].

```
1  SELECT ?obs ?supCity (SUM(?sales) AS ?totalSales)
2  WHERE {?obs rdf:type qb:Observation ;
3             gnw:customerID ?cust ;
4             gnw:supplierID ?sup ;
5             gnw:salesAmount ?sales .
6  ?cust qb4o:memberOf gnw:customer ;
7             gnw:customerGeo ?custGeo .
8  ?sup qb4o:memberOf gnw:supplier;
9             gnw:supplierGeo ?supGeo ;
10            skos:broader ?supCity .
11 ?supCity qb4o:memberOf gnw:city .
   # Inner select for the distance function
12   {SELECT ?cust1 (MIN(?distance) AS ?minDistance)
13   WHERE {?obs rdf:type qb:Observation ;
14            gnw:customerID ?cust1 ;
15            gnw:supplierID ?sup1 .
16            ?sup1 gnw:supplierGeo ?sup1Geo .
17            ?cust1 gnw:customerGeo ?cust1Geo .
18            BIND (bif:st_distance(?cust1Geo, ?sup1Geo)
19            AS ?distance)}
20            GROUP BY ?cust1 }
21            FILTER (?cust = ?cust1 && bif:st_distance
22            (?custGeo, ?supGeo) = ?minDistance)}
23 GROUP BY ?supCity ?obs
```

Query 1: Query with s-roll-up formulated in SPARQL

2. QUERIES FOR SPATIAL SEMANTIC DATA WAREHOUSES

To formulate SPARQL queries automatically, GeoSemOLAP needs some information about what data is contained in the spatial data warehouse. Our current implementation uses metadata using the QB4SOLAP vocabulary[1] [4, 5] for this purpose. In addition to MD data elements, QB4SOLAP also describes spatial concepts and builds upon existing vocabularies: *QB*[2] and *QB4OLAP*[3].

A QB4SOLAP data cube contains a set of *observations*. Its structure is defined through a data structure definition (DSD) describing standard concepts, such as *dimensions*, *measures*, *hierarchies*, *hierarchy steps*, *levels*, and *level attributes*, as well as spatial concepts, such as spatial aggregate functions, topological relations, spatial attributes, and spatial measures.

Let us consider an example of a company (Northwind) that exports a number of goods. The company records its sales data[4] in a spatial data warehouse, which is based on an MD model enabling analyses of sales according to their geographical distribution. The company decides to share the data warehouse on the Semantic Web for further analysis of its sales (e.g., involving economic and demographic data published as Open Data on the Web) and to provide access

[1]QB4SOLAP: `https://w3id.org/qb4solap#`
Vocabulary Structure: `http://extbi.cs.aau.dk/QB4SOLAP`
[2]RDF Data Cube: `https://w3.org/TR/vocab-data-cube/`
[3]QB4OLAP: `https://lorenae.github.io/qb4olap/`
[4]`http://northwinddatabase.codeplex.com/`

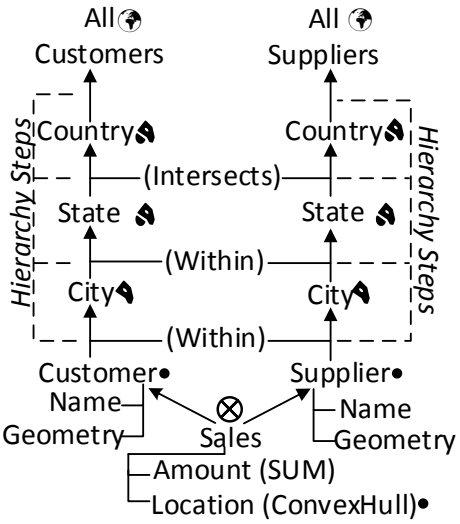


Figure 2: Northwind spatial data cube members (symbols next to level names represent spatial characteristics of level members, e.g., point, polygon, and multi-polygon.)

to all branches as well as customers and suppliers. Fig. 2 illustrates an example schema of the company's spatial data warehouse.

The example query from Section 1 (Query 1: "Total sales to customers grouped by city of their closest supplier") can on a high level be represented as: S-ROLL-UP (Sales, [DISTANCE(Customer, Supplier)] → ClosestCity, SUM(SalesAmount)). The query's s-roll-up operator takes the sales observations as input. Each sale observation has a set of associated *measures* representing quantitative descriptions, e.g., Sales Amount and Sales Location – the latter corresponds to a *spatial measure*. Measures have *aggregation* functions (e.g., SUM for Sales Amount and Convex Hull for Sales Location) that can be used to combine several measure values when rolling-up to a higher *level*, such as from City level to Country level. In Fig. 2 both Customer and Supplier are *spatial dimensions*.

The graph pattern in Lines 2–11 of Query 1 describes the roll-up path as a path-shaped join of triple patterns connecting observations to target levels but also to the corresponding measures and attributes.

Line 1 in Query 1 selects the sales observations (`?obs`) and the *spatial* level (`?supCity`) as output. It also specifies to use aggregation function `SUM` on the measure (`?sales`). Lines 2–5 describe a star-shaped join of triple patterns connecting sales observations (`?obs`) to the Sales Amount measure (`?sales`) as well as to the base level members of two spatial dimensions: Customer (`?cust`) and Supplier (`?sup`).

City→State→Country→ *All* in Fig. 2 depicts *spatial* hierarchies for the Customer and Supplier dimensions. In Query 1, Lines 6, 8, and 11 query intermediate spatial levels of these hierarchies. For each spatial level we query the spatial attributes (Lines 7 and 9), e.g., Customer and Supplier have *points*. Line 10 describes the roll-up from the lowest level in dimension Supplier to a higher level City. Each roll-up between levels is defined as a hierarchy step. Spatial hierarchy steps have a topological relation between the levels (e.g, Customer→$_{(Within)}$City or State→$_{(Intersects)}$Country,

Fig. 2), which is annotated with QB4SOLAP in the DSD schema.

The rest of Query 1 (Lines 12-23) represents an inner select operation for calculating the distances between Customer and Supplier locations, which is necessary to find the closest Supplier cities for the SOLAP operation. The inner select binds the calculated distances to the existing schema members from the outer select. Thus, it is very similar in structure to the first part of the query besides the spatial function (*st_distance*).

3. SYSTEM OVERVIEW

3.1 GeoSemOLAP Workflow

The workflow of querying spatial semantic data warehouses with GeoSemOLAP consists of six main steps, as illustrated in Fig. 3. First, the user selects a SOLAP operator. In dependence on which SOLAP operator was selected, the user can choose several items from a drop-down menu to complete the definition of the operator, e.g., schema elements (spatial levels, attributes, etc.) and spatial operations (distance, within, etc.).

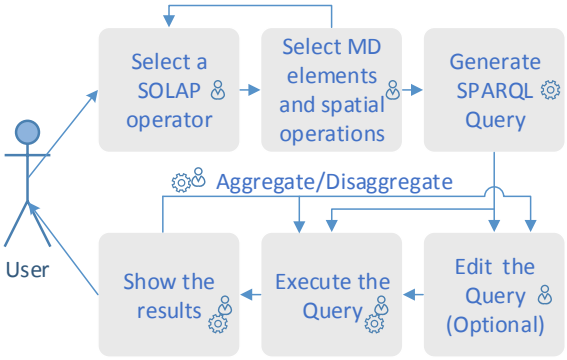


Figure 3: Workflow diagram

As some operators (e.g., s-slice, Fig. 5b) require spatial coordinates as input, GeoSemOLAP displays snippets of geographic maps so that the user can click on a position to indicate spatial coordinates that the query should operate on.

Afterwards, the user can decide to select another SOLAP operator that shall be applied on the results of the previous operator (nesting) and set its parameters.

Then, the SPARQL query is automatically generated by GeoSemOLAP. The query can optionally be edited by the user before GeoSemOLAP sends the query to a SPARQL endpoint (local or remote) for execution.

Finally, the result of the query is displayed to the user. The user may aggregate the results or decide to continue editing the query by for example adding additional SOLAP operators and repeating the steps above mentioned.

3.2 GeoSemOLAP Architecture

GeoSemOLAP consists of five architectural components: *GUI, Metadata Manager, Query Generator, Data Processor*, and *SPARQL Endpoint*. The system architecture is illustrated in Fig. 4. GeoSemOLAP is implemented in Javascript, HTML, and CSS. It uses Leaflet[5] for visualizing maps and Virtuoso Open Source Edition (7.2) for running the endpoint.

[5]`http://leafletjs.com`

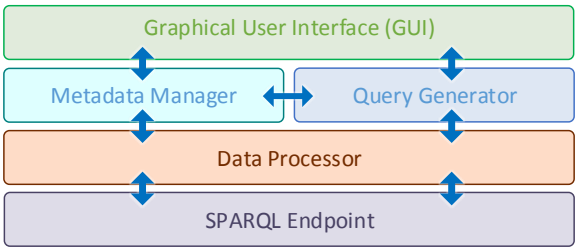


Figure 4: GeoSemOLAP architecture

4. DEMONSTRATION SCENARIO

We will demonstrate GeoSemOLAP using two datasets (GeoFarmHerdState [4] and Geo-Northwind), which are both also available at our public endpoint[6]. The Geo-Northwind data cube has already been explained in Sect. 2. GeoFarmHerdState [4] is a spatial data cube about livestock holdings in Denmark. To enable interesting analyses, we have integrated data about livestock holdings with environmental and geographical data.

At the conference, we will demonstrate GeoSemOLAP by allowing attendees to interact with the system and formulate queries. To make it easier for the audience to understand the data and the cube structure, the graphical interface displays a high-level graphical representation at the top of the page – as illustrated in Fig. 5a.

Hence, conference attendees will be able to directly interact with the system, which during the demonstration will run on a laptop connected to an external screen. Although GeoSemOLAP can use a remote endpoint to execute the formulated queries, the demonstration will use a local endpoint (running on the same laptop as the graphical interface) to avoid problems with an unstable Internet connection.

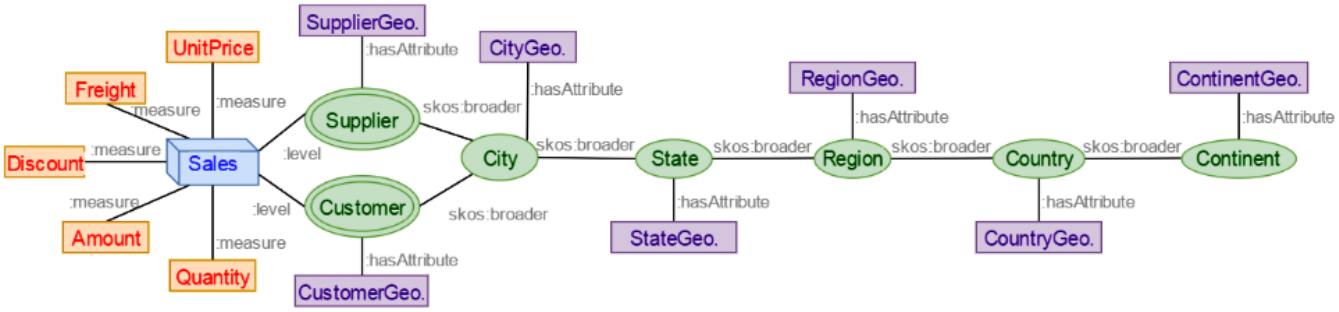
Fig. 5 shows a compact screenshot of GeoSemOLAP. At the top, we see a graphical representation of the used data cube. As mentioned above, it shows the most important concepts that are necessary to formulate a SOLAP query, e.g., dimensions, hierarchies, measures, spatial levels, and level attributes.

In this example (Fig. 5), the user has first selected the operator s-slice. The menu on the left-hand side (Fig. 5b) allows the user to set the parameters so that the s-slice operator selects geometries from a map or spatial levels from the schema. S-slice requires two spatial parameters; the first one to define a spatial location and the second one to define the slice level with respect to the given location. Hence, to help the user better select coordinates to define a location, a map is displayed so that the user can simply click on a point, which is then automatically extracted. In Fig. 5b, we can see that the user clicked on a point in Germany and then selected Country from the spatial levels to make a projection on the observations in this country.

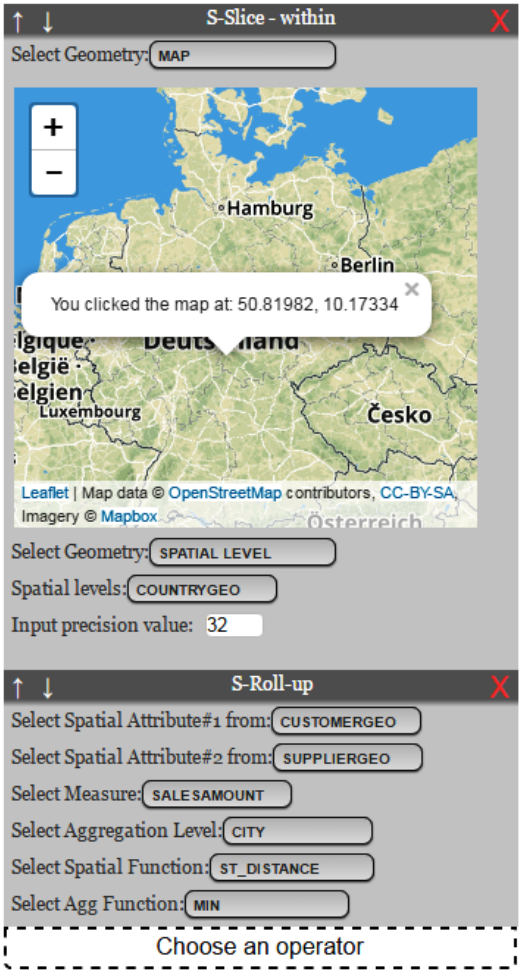
In addition, to an s-slice operator, the user has added an s-roll-up operator to aggregate measures and discover new perspectives on sales with respect to their spatial location – the s-roll-up operator with its parameters is displayed under s-slice in Fig. 5b. The obtained result (from s-roll-up) is similar to our running example introduced in Sect. 1 (Query 1).

Based on the provided operators and parameters, GeoSemOLAP will automatically create and display the corresponding SPARQL query (Fig. 5c). The user can then de-

[6]`http://lod.cs.aau.dk:8890/sparql`



(a) Graphical representation of an example use-case schema



(b) SOLAP operator configuration

(c) Generated SPARQL query for nested SOLAP

obs2	supplierCity	cityName	totalSales
http://qb4solap.org/cubes/instances/geonorthwind#sale_10643_3	http://qb4solap.org/cubes/instances/geonorthwind#city_22	"Lyngby"	18
http://qb4solap.org/cubes/instances/geonorthwind#sale_10323_2	http://qb4solap.org/cubes/instances/geonorthwind#city_6	"Berlin"	44.8
http://qb4solap.org/cubes/instances/geonorthwind#sale_10312_4	http://qb4solap.org/cubes/instances/geonorthwind#city_14	"Frankfurt"	62

(d) Example result for a nested SOLAP query (S-Roll-up (S-Slice()))

Figure 5: Screenshot of GeoSemOLAP

cide to run the query and view the results. The results for the example query are displayed at the bottom of the page (Fig. 5d). We refer to our screencast[7] for a more detailed explanation of GeoSemOLAP.

5. PERSPECTIVES AND FUTURE WORK

Fig. 6 sketches our vision of a tool-oriented future for SOLAP on the SW based on the models and algorithms proposed in [5].

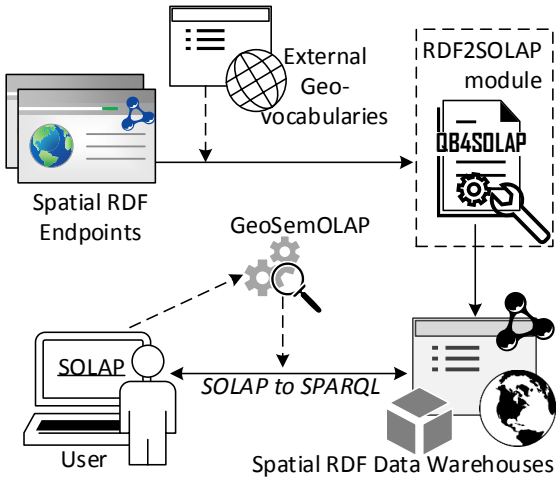


Figure 6: Vision of SOLAP on the Semantic Web [5]

As mentioned in Sect. 1, there is a growing popularity of spatial LOD datasets from various governmental domains[8,9,10,11]. These datasets contain observations and measures that are well-suited for analytical queries (e.g., water/air quality measurements, immigration rates, EU subsidies in agriculture, crop revenue, etc.). However, as observed in [5] such datasets are typically not modeled with spatial dimension levels and hierarchies. Thus, they cannot directly be queried with SOLAP on the SW.

With currently available SW technologies, a user, who would like to query available spatial RDF data with SOLAP, needs to download the datasets, map them to a relational data model, and then import the result into a traditional spatial data warehouse. Obviously, this workflow is not only slow but it is also time-consuming and requires storing the data in a non-open format on a local system.

GeoSemOLAP considerably lowers the entry barrier for advanced spatial analysis on the SW by providing a user-friendly interface to formulate SOLAP queries in SPARQL. Our future work strives at lowering the entry barrier even further by developing (semi-)automatic techniques for annotating existing spatial RDF data on the SW with QB4SOLAP and defining spatial levels and hierarchies using available datasets, such as GeoNames[12]. Furthermore, it would be interesting to extend the model proposed in [5] and GeoSemOLAP to handle highly dynamic spatio-temporal data and queries as, for instance, found in large-scale transport analytics [3].

[7] https://youtu.be/Pc3RBPPgBhA
[8] https://ec.europa.eu/eurostat
[9] https://environment.data.gov.uk
[10] https://datahub.io/dataset/govagribus-denmark
[11] https://datahub.io/dataset/acorn-sat
[12] http://www.geonames.org/

6. REFERENCES

[1] A. B. Andersen, N. Gür, K. Hose, K. A. Jakobsen, and T. B. Pedersen. Publishing Danish Agricultural Government Data as Semantic Web Data. In *Semantic Technology*, 2014.

[2] S. Bimonte, A. Tchounikine, M. Miquel, and F. Pinet. When spatial analysis meets OLAP: Multidimensional model and operators. *IJDWM*, 2011.

[3] G. Gidofalvi, T. B. Pedersen, T. Risch, and E. Zeitler. Highly scalable trip grouping for large-scale collective transportation systems. In *EDBT*, 2008.

[4] N. Gür, K. Hose, T. B. Pedersen, and E. Zimányi. Enabling Spatial OLAP over Environmental and Farming Data with QB4SOLAP. In *Semantic Technology*, 2016.

[5] N. Gür, T. B. Pedersen, E. Zimányi, and K. Hose. A Foundation for Spatial Data Warehouses on the Semantic Web. *Semantic Web Journal*, 2017.

[6] M. Koubarakis, M. Karpathiotakis, K. Kyzirakos, C. Nikolaou, and M. Sioutis. Data Models and Query Languages for Linked Geospatial Data. *Reasoning Web*, 2012.

7. ACKNOWLEDGEMENTS

This research is partially funded by the European Commission through the Erasmus Mundus Joint Doctorate Information Technologies for Business Intelligence (EM IT4BI-DC) and the Danish Council for Independent Research (DFF) under grant agreement no. DFF-4093-00301.

8. BIOGRAPHIES

Nurefşan Gür is a Ph.D. student at the department of Computer Science at Aalborg University, Denmark. Her research focuses on spatial OLAP and data warehouses on the Semantic Web. She is working on integrating Linked Open Governmental Datasets from spatial domains to enable business intelligence.

Jacob Nielsen is a student programmer at the department of Computer Science at Aalborg University, Denmark, where he currently studies Computer Science. His work currently focuses on providing a user-friendly way to formulate SOLAP queries in SPARQL in consideration of spatial levels and coordinates.

Katja Hose is an associate professor at the department of Computer Science at Aalborg University, Denmark. Her research interests include Linked Open Data, knowledge representation, query processing and optimization in distributed systems, rank-aware query operators, and energy data management.

Torben Bach Pedersen is a professor of Computer Science at Aalborg University, Denmark. His research concerns business intelligence and big data, especially "Big Multidimensional Data" – the integration and analysis of large amounts of complex and highly dynamic multidimensional data.