|  |
| --- |
| *Spatial Cube using Shared Dimensions and Neighborhood Relationships Concepts* |
|  |
| Tárik de Melo e Silva Rocha3, Rodrigo Rocha Silva2, Tiago Garcia de Senna Carneiro1 and Joubert de Castro Lima1 |
| 1Department of Computer Science, Federal University of Ouro Preto (UFOP), Ouro Preto, Brazil  *2Faculty of Technology of Mogi das Cruzes (FATEC), Mogi das Cruzes, Brazil*  *3Information Technology Company of Minas Gerais (PRODEMGE), Belo Horizonte, Brazil*  {tarik.melo, rrochas, tiagogsc, joubertlima}@gmail.com |

Keywords: Spatial OLAP, Parallel Database, Neighborhood Relationship, Data Cube Operator

Abstract: A data cube has exponential storage and runtime complexity according to a linear dimension increase. A spatial data cube turns the problem even harder, since it integrates spatial types, spatial operators, spatial measures and spatial hierarchies into a data cube. In this paper, a new cube approach, named Spatial-Cubing or just S-Cubing, implements two spatial cube computation techniques and two spatial cube non-relational storage representations. S-Cubing based on shared dimensions is the first non-relational approach that computes and represents spatial cubes with continuous dimensions, resolution hierarchies, multiple spatial measures, including complex spatial measures composed of statistical functions together with a spatial operator, and capable to run over multicore computer architectures. This way, S-Cubing attempts the most recent SOLAP data model requirements to fully exploit spatial-multidimensional analysis. S-Cubing based on neighborhood relationships implements a new cube aggregation algorithm using relationships among cells of a regular space. Thus, such an algorithm avoids detailed spatial hierarchy definition, a complex and mandatory OLAP activity.

# 1 INTRODUCTION

The data cube relational operator is very useful for decision support systems, since it is optimized to read both coarse and fine grained summarized data and to support recurrent data or structure updates. Let’s suppose a decision maker of a car agency wants information about sales of a certain model of car. We can evaluate AVG, SUM, VARIANCE, MIN and MAX number of car sales using perspectives like time, agency and vendor. This way, for each perspective combination, there is a car sale value, named measure value. Perspectives are named dimensions and they are composed of attributes. A time dimension may have year, month, week, day and hour attributes, so a single time dimension has multiple hierarchies, i.e. time can be organized as {year, week and hour} or {year, month, day, hour} or any other configuration. A data cube is very fast to obtain summarized values from different hierarchies and dimensions, but it is a combinatorial problem, where a linear increase in the number of dimensions turns both data cube computation and storage exponentials.

Spatial data cubes are extensions of alphanumeric data cubes, where new types, measures and hierarchies must be considered. Spatial data cubes consider integer, double, string and many other alphanumeric data types, but also point, line, polyline and polygon spatial data types. Spatial data cubes consider numeric measures (AVG, MIN, MAX, SUM, MODA, etc.), but also spatial measures (UNION, INTERSECTION, TOUCH, BUFFER, etc.). Hybrid measures and dimensions must be supported by spatial data cubes, i.e. any spatial type may have alphanumeric attributes and, sometimes, statistical functions associated with it. Time, non-spatial dimension hierarchies and also spatial hierarchies from different spatial objects, such as polygons, lines and so on, can be implemented in spatial data cubes. In summary, data cube problem becomes even harder when spatial properties are considered.

Since the seminal paper of Gray et al. (Gray, 1996), efficient data cube approaches are reducing storage and runtime impacts to compute full or partial cubes without loss of generality. There are cube solutions for different cube types, including traditional or alphanumeric data cubes (Lima, 2011) (Sismanis, 2002) (Xin, 2007), geo data cubes (Alzate, 2012) (Moreno, 2009) (Shekhar, 2001), text data cubes (Bringay, 2011) (Lin, 2008) (Yu, 2009), graph data cubes (Zhao, 2011), RFID or stream data cubes (Gonzalez, 2006) (Nienartowicz, 2013) and image data cubes (Jin, 2010). Promising high performance computer architectures, using Graphic Processing Units (GPU), Cluster and Grid computing, demonstrate that the cube problem is also hard to be efficiently partitioned (Moreira, 2012) (Kaczmarski, 2011) (Wang, 2010) (Zhang, 2012).

Besides improvements in runtime and memory consumption, there are several improvements in spatial data cube modeling. Modern cube models include complex spatial measures, multiple spatial measures, continuous fields in dimensions, incomplete continuous field data and multi-resolution data (Bimonte, 2010) (Bimonte, 2012) (Gómez, 2012) (Sandro, 2006) (Zaamoune, 2013). Unfortunately, such spatial data cube models are implemented and tested on top of a relational database, a demonstrably efficient solution for transactional applications and not analytical ones, like Online Analytical Processing (OLAP), Data Mining, Information Retrieval (IR), Statistics, Simulation and many others.

Non-relational sequential and parallel cube approaches (Alzate, 2012) (Moreno, 2009) (Shekhar, 2001) (Zhang, 2012) do not consider all previous explained improvements in their spatial data cube models. In this paper, we implement and test a non-relational Spatial OLAP (SOLAP) approach, named Spatial-Cubing or just S-Cubing, which proves to be faster than a traditional relational implementation. S-Cubing algorithms and data structures implement the recent SOLAP models concepts to fully exploit spatial-multidimensional analysis. A parallel S-Cubing version demonstrates that it is possible to scale over multicore computer architectures.

A second contribution of this paper is the aggregation algorithm based on neighborhood relationships. From a base regular cell space, composed by several spatial dimensions and measures, it is possible to generate new aggregated cells using several neighborhood relationships. The algorithm produces an automatic hierarchy, where a base regular cell space is the lower level of the lattice (base *cuboid*) and a one cell regular space is the higher level of the lattice (apex *cuboid*). The neighborhood relationship(s) indicate(s) how many intermediate levels will exist and how they are organized in the lattice of *cuboids*. No hierarchy definition and no intermediate regular cell spaces are required, since the algorithm successively run a neighborhood calculus to produce new cells with new thematic maps in a data cube.

The rest of the paper is organized as follows: Section 2 details some promising spatial data cube prototypes and models, pointing out their benefits and limitations. Section 3 details S-Cubing approach, i.e., its architecture, data structures and algorithms. Section 4 describes experiments and results of S-Cubing. Finally, in Section 5 we conclude our work and point out the future improvements of S-Cubing.

# 2 RELATED WORK

The combination of GIS strengths with OLAP strengths introduces a new database area, named SOLAP (Bédard Y. L., 1997). In SOLAP literature, Bédard et. al (Bédard Y. M., 2001) and Han et. al (Han, 1998) introduce algorithms challenges and model concepts, such as spatial dimensions, spatial measures and spatial hierarchies. Recent studies go further and introduce new challenges like continuous data, multi-scale data, complex measures, multiple measures and incomplete data (Bimonte, 2010) (Bimonte, 2012) (Sandro, 2006) (Zaamoune, 2013).

When analyzing natural phenomena, like meteorology or pollution, the discrete structures are not adequate to spatial-multidimensional analysis (Ahmed, 2005). In general, the literature (Ahmed, 2005) (Gómez, 2012) (Zaamoune, 2013) adopts interpolation methods to enable user friendly drill-down, roll-up, slice and dice operations using continuous dimensions. Neighborhood calculus and interpolation methods are also used to compensate missing values, as (Zaamoune, 2013) details. However, interpolation hides data details, therefore sometimes it is mandatory to also store detailed spatial/temporal data series in a cube *lattice*.

A spatial cube computation algorithm can create new thematic maps and measure values. There are conventional spatial hierarchies (Ex. neighborhood, city, province, state and so many others) and scale hierarchies (Ex. neighborhood in scale 1:10000, city in scale 1:50000, province in scale 1:100000). The work (Bimonte, 2012) implements a cube model and a relational SOLAP prototype, considering scale hierarchies and their challenges, i.e., the support of non-strict and non-covering hierarchies, imprecise aggregate measures and dimension constraints (Bimonte, 2012).

Spatial measures are composed by spatial objects and a spatial operator (UNION, INTERSECTION, TOUCH, BUFFER, EUCLIDIAN DISTANCE, etc.). Complex spatial measures have spatial objects with several features, spatial operators and statistic functions. Complex spatial measures integrate spatial object features with spatial operators or statistic functions. A spatial object feature can be its projection, position, scale, name, description, humidity, temperature and many others. This way, complex spatial measures have conventional measures integrated with spatial ones, enabling hierarchies on each complex measure and the utilization of several complex spatial measures in a data cube. Complex spatial measures were detailed in (Bimonte, 2010) and any modern study should consider them to provide deep spatial-multidimensional analysis. Bimonte et al. (Bimonte, 2010) advocate that spatial dimensions and measures must have a common representation, so they can be exchanged on-the-fly by the end user by using a new SOLAP operator, named permute.

Several SOLAP tools and prototypes (Sandro, 2006) (Scotch, 2005) (Technologies, 2005) (Ferraz, 2010) (Colonese, 2006) were developed since Bédard et. al seminal paper (Bédard Y. L., 1997). They frequently support 3D map visualization and multiple complex spatial measures, but few support continuous dimensions, scale hierarchies, parallelism, permute operator, 3D spatial object trajectory visualizations and missing continuous values. None of them computes cubes from regular cell spaces using only neighborhood relationships to produce automatic spatial hierarchies. For example, a new polygon can be created in a regular cell space from uniting its eight neighbors. This neighborhood relationship was proposed by [Edward F. Moore](http://en.wikipedia.org/wiki/Edward_F._Moore) (Codd, 1968), a pioneer of cellular automata theory. Other properties of the polygon, such as temperature, humidity and many others can also be aggregated. Therefore, for each non-spatial property of the polygon a statistical function, including AVG, SUM, MIN, MAX, RANK and many others, or a text function, including CONCAT, GROUP, TRUNCATE and many others, must be defined. The above SOLAP prototypes and solutions were implemented on top of a relational database or Relational-OLAP server, like Mondrian (Mondrian, 2014). Relational databases are designed to offer [Atomicity](http://en.wikipedia.org/wiki/Atomicity_(database_systems)), [Consistency](http://en.wikipedia.org/wiki/Consistency_(database_systems)), [Isolation](http://en.wikipedia.org/wiki/Isolation_(database_systems)), [Durability](http://en.wikipedia.org/wiki/Durability_(database_systems)) (ACID)  properties, therefore are not designed for multidimensional analysis. In (Zhang, 2012), the authors reinforce this assumption.

The Map Cube operator (Shekhar, 2001) is a promising non-relational cube indexing and representation, similar to S-Cubing. Its input is composed of a map, a base relation and a cartography preference. The output is a set of thematic maps, composed of spatial types (point, polygon and so on) and their properties, initially stored in a base relation. In (Moreno, 2009), Map Cube operator was extended for multiple spatial measures, but it does not consider complex spatial measures, where we can INTERSECT polygons, SUM ages and AVERAGE salaries, for instance. Map Cube does not consider automatic hierarchies for regular cell spaces, using a neighborhood relationship (Ex. 3x3, 4x4, 8x8 neighborhood relationships and many others). In (Alzate 2012), the Map Cube operator was extended to support spatial-temporal queries. A spatial-temporal analysis observes a spatial object trajectory over time. In (Alzate 2012), the authors did not implement spatial-temporal visualizations. Moreover, map cube was not extended for SOLAP requirements like continuous dimensions, scale hierarchies, permute operator and missing continuous values.

The same modeling and algorithmic limitations, i.e. continuous dimensions, scale hierarchies, permute operator and missing continuous values, are observed in parallel spatial data cube approaches, like (Zhang, 2012). The approach S-Cubing can be considered the first high performance non-relational data cube solution considering such requirements.

# 3 S-Cubing Approach AND Prototype

S-Cubing data structures and algorithms are non-relational and support spatial-temporal data. It is designed for continuous dimensions, enabling fine grained continuous data storage and coarse grained discrete data storage, it runs in parallel, supports different scale hierarchy implementations, handles multiple complex spatial measures and has an accurate cube representation in presence of nulls.

A prototype to S-Cubing approach was implemented to validate algorithms, data structures and technologies. The prototype architecture is illustrated in Figure 1. It can load spatial data from shape files or PostGIS databases (PostGIS, 2014). SC objects are composed by features, used to build measures and dimensions. Non regular cell spaces adopt SC tree indexing and representation, composed by SC nodes. Regular cell spaces adopt SC Matrix indexing and representation, composed by SC cells. Both cube indexing use a bag-of-tasks to delegate spatial and statistic calculus to multiple threads. The hierarchies are implemented by a SC tree and multiple SC Matrixes. Finally, tabular values and thematic maps are presented using Nasa World Wind (Bell, 2007).

FIG1.

Nasa World Wind has a 3D viewer, a multiple layer explorer and a tabular view integrated with a 3D map view. It is multiplatform, i.e. it works on Web, Android and Desktop. Figure 2 illustrates S-Cubing prototype, which is able to compute data cubes with both non-spatial and spatial hierarchies. Figure 2 illustrates S-Cubing working with a spatial database of Ouro Preto, a historical city in Brazil. Buildings of Ouro Preto were grouped and aggregated according to different perspectives. Multiple scales, missing values, continuous dimensions and multiple complex measures were considered to attempt Ouro Preto city hall requirements. A regular cell space of the city was also built to test automatic spatial hierarchy construction based on neighborhood relationships. More details about S-Cubing real case study, using Ouro Preto buildings, can be found in (Silva, 2012).

FIG2.

There are two strategies to compute a cube in S-Cubing: (i) based on shared dimensions calculus and (ii) based on neighborhood relationships calculus. The first S-Cubing cube computation strategy adopts some Star-Cubing approach (Xin, 2007) innovations to compute full or iceberg spatial data cubes. An iceberg cube index only frequent measure values to reduce both runtime and memory consumption, but it is a hard challenge, as (Han, 2001) discuss, to implement efficient pruning methods that avoids unnecessary computations. The shared dimensions idea is a promising alternative to anticipate infrequent measure values pruning, but also for improvements in full data cube computation scenarios, since it enables the integration of top-down and bottom-up computation strategies to reduce indexing runtime costs.

S-Cubing shared dimensions based strategy considers the benefit of both bottom-up and top-down aggregation strategies, i.e. it computes a data cube using first a top-down strategy and the concept of shared dimensions, where intermediate aggregations are computed in a bottom-up fashion. SC tree is used to optimize cube indexing. A SC tree is composed by SC nodes, where each node has base relation attribute values, measure values and pointers to descendant nodes. Figure 3 illustrates a full SC tree representation, computed from relation R, also illustrated in Figure 3. Relation R represents some diseases (D), sex and months (M). It also illustrates a complex spatial measure composed by (GEO ID) and disease occurrences (DO). The column GEO ID can have 0 or N GEO identifiers, where N is the maximum number of spatial objects in a database schema. A GEO ID is a unique spatial key, regardless its representation, i.e. it does not matter if it is a point, a polygon or a multiline. GEO ID is also independent of its projection, scale and layer, the last one representing a set of GEO IDs organized as a spatial context, i.e. a layer for residences, a layer for water supply, a layer for the territorial satellite or airplane image and many others. The spatial object features are represented by diseases (D), sex and months (M). We omit geometry, location, projection and scale mandatory spatial object features in R to be simple.

FIG3.

The first tuple *t1=*{Chagas, M, Jan, {11, 38}, {102, 13}} is stored one value per time in S-Cubing tree, forming a tree path *root→Chagas→M→Jan*, as illustrated in Figure 3. All S-Cubing tree nodes have an attribute value (Ex. Chagasor [GEOID\_001 … GEOID\_N]), one or many conventional measure values, one or many spatial measures, including complex ones, and pointers to its descendants that are S-Cubing tree nodes. The idea of shared dimensions enables intermediate S-Cubing tree nodes to store aggregations during a single scan of a base relation. This can be seen in Figure 3 at Chagasdisease node which is associated with GEO IDs 11, 38 and 02, after a single scan of R. The disease occurrences (DO): 102, 13, 15 and 47 are stored individually to enable any statistic calculus (AVG, MAX, MODA, SUM, etc.) with them. The same is done with GEO IDs, i.e. they are stored individually to enable any spatial operation with them (UNION, INTERSECTION, TOUCH, BUFFER, EUCLIDIAN DISTANCE, etc.). Each node in Figure 3 represents a path from root to the current node, thus an aggregation. A special node, called *all*, is used when a dimension is aggregated, i.e., when its attribute values are collapsed in a unique value. Sometimes, S-Cubing tree can become a direct acyclic graph (DAG) to avoid unnecessary redundant nodes. In Figure 3 example, leaf nodes *Jan* and *Feb* are reused, since their measure values are identical. The *root* node represents the *apex cuboid*, where the most aggregated measure values are stored.

After a single scan of relation R, S-Cubing starts generating the remaining aggregations. For this, it starts generating aggregations from leaf nodes to root node, in a top-down fashion. Figure 4 illustrates S-Cubing trees after some aggregations. The first S-Cubing tree in Figure 4 represents the cube after R scan. The last tree is a full data cube, identical to S-Cubing tree in Figure 3. The first path computed after a base relation scan is *root→Chagas→M→Jan*, which produces path *root→Chagas→all→Jan*. Then, the path *root→Chagas→F→Feb* is computed, producing *root→Chagas→all→Feb* path, as Figure 4 (B) illustrates. Next, the paths *root→all→F→Feb* and *root→all→M→Jan* are created, reusing descendant nodes of *Chagas* node. From such aggregated paths, S-Cubing can generate paths *root→all→all→Feb* and *root→all→all→Jan*, also reusing *Jan* and *Feb* leaf nodes. Figure 4 (C) illustrates these four aggregated paths in S-Cubing tree. Next, the path *root→Dengue→M→Jan* must be traversed to produce new aggregated paths. This way, in Figure 4 (D) there is the first aggregated path: *root→Dengue→all→Jan*. The second aggregated path is *root→all→M→Jan*, but it exists in S-Cubing tree, so the algorithm checks if it can reuse the nodes. In the example, nodes *M* and *Jan* cannot be reused, so new nodes must be created. The path *root→all→all→Feb* is the same created before, but the path *root→all→all→Jan* is different from Figure 4 (C), since node *Jan* has new measures values and cannot be reused. The result is illustrated in Figure 4 (E).

The shared dimensions idea is a fundamental optimization, since it produces intermediate aggregate nodes from a single base relation scan. A partial cube can compute with only S-Cubing tree illustrated in Figure 4(A), and generate the remaining paths according to user queries. In general, S-Cubing can compute full or partial cubes, including iceberg cubes with frequent nodes, i.e. nodes measures above a certain user threshold (Ex.DO > 25,GEO\_IDs with distance > 100km from Amazonas River with GEO\_ID47 for instance,and so on).

FIG4.

Formally defined, a S-Cubing SC generates a new relation composed by spatial tuples sts of the form st*=* {D1, D2, D3, … Dn, m1, m2, …my}, where D represents a spatial or non-spatial dimension, *n* the cube dimensionality and *y* the number of measures, including spatial ones. Spatial dimension is defined as sd*=*{Gid1, Gid3, … Gidz, Sop}, where Gid represents a GEO ID and Sop represents a spatial operator, such as UNION, INTERSECTION, TOUCH, EUCLIDIAN DISTANCE and many others. Each node in S-Cubing tree requires a spatial library call. S-Cubing adopts GEOTools as a third part spatial library (Garnett, 2003) (Turton, 2008). Successive GEO Tools calls produce Gids for both spatial dimensions and measures. Each measure *m* can be a spatial or non-spatial measure. A non-spatial measure *m* can be defined as *m=*{mv1, mv3, … mvk, MF}, where *mvs* are double-integer values used by a statistical function MF, being MF*=* {MIN *|* MAX *|* AVG *|* SUM *|* VARIANCE *…*}. Some statistical functions use one *mv*, like SUM, COUNT, etc., but several, like VARIANCE and MODA for instance, use many *mvs*. S-Cubing adopts Apache Math Library (Apache, 2014) for non-spatial measures. A spatial measure *sm* can be defined as sm*=*{Gid3, Gid45, … Gidp, Sop}, identical to a spatial dimension. A *sm* can be extended, producing a complex spatial measure *csm* defined as csm*=*{sm, sm2, … m1, m2, … my}. Both sm and m were defined previously.

Algorithm 1 presents how S-Cubing generates all nodes from a base S-Cubing tree, similar to Figure 4 (A). The algorithm output is a spatial cube like the one illustrated in Figure 4 (E). The algorithm starts scanning the S-Cubing tree from root to leaf nodes (line 8). When a leaf node is reached a backtracking process starts and there are several copies of descendant nodes to ancestor nodes (lines 13-17). If a node N does not have a descendant node D, all D descendants are also descendants of N (line 11). If N has a descendant D, we must check if N descendant D is a reused node, i.e. a node referenced before. If it is a referenced node, a cloned node must be created (Lines 14 and 15) to avoid wrong measure values in the lattice. Otherwise, just an update is necessary (Lines 16 and 17). In summary, each set of descendant nodes, from a level of S-Cubing tree, has its measure values updated (lines 14-15 or 16-17) using an asynchronous bag-of-tasks (Kwan, 2010). We must join all started threads before begin a new level of S-Cubing tree, since ancestral nodes are computed from descendants nodes. This way, all descendant measures values must be computed before an ancestral node is processed. Line 21 guarantees such a requirement.

ALG1.

“Fields describe physical phenomena that change continuously in time and/or space, like temperature, land elevation, land use and population density, frequently used in human geography. They are perceived as having a value at each point in a continuous N-dimensional spatial and/or spatiotemporal domain. In real-world practice, scientists and practitioners register the values of a field by taking samples at (generally) fixed locations, and inferring the values at other points in space using some interpolation method. Thus, fields can be described by a function that indicates the distribution of the phenomena or feature of interest. The most popular discrete representation for fields is the raster model, where the 2D space is divided into regular squares.”(Gómez, 2012)

Fields can be continuous spatial dimensions or measures. They impose hierarchies and must support classic OLAP operators, like Roll-up, Drill-down, and/or Drill-across. S-Cubing approach addresses a solution to spatial fields, as Figure 5 illustrates. A spatial dimension city, with name and GEO ID attributes, has several temperatures, each of them described as a temperature double value and a GEO ID spatial location. In S-Cubing tree the temperature values and locations are one tree level, no matter if temperature is obtained from a raster representation. Aggregations are performed normally in a top-down/bottom-up strategy and classic OLAP operations are possible. Temperatures can be classified as hot, cold and normal, for instance. This way, the attribute temperature can be associated to these labels, forming a new attribute, named, for instance, Temperature Classification, in the base relation and a new level in S-Cubing Tree. This kind of information, i.e., hot, cold and normal temperatures, can also be obtained from regular multidimensional queries on the fly, using RANGE filters like: “Temperature BETWEEN 18-25o Celsius is normal, LESS THAN 18o Celsius and HIEHER THAN 25o Celsius is hot. HIGHER THAN 35o Celsius can be considered very hot”.

FIG5.

In Figure 5, Belo Horizonte city has four locations to collect the temperature of day 10 at 1pm and six locations on the same day at 8pm. S-Cubing approach deals with such particularity with no impact to the end user. The presence of nulls is also supported by S-Cubing approach. Figure 6 illustrates a new tuple, where there is a collection of temperature values from São Paulo city without hour information. One of the locations responsible to collect temperature also fails to collect data, producing a null value. The aggregations and OLAP operations are performed normally without considering dimension hour, but considering all aggregations of higher levels of the hierarchy, i.e., day and city dimensions, respectively. Figures 5 and 6 consider fields as continuous spatial measures, but they can be continuous spatial dimensions. Similar to city spatial dimension, temperature spatial measure can become a spatial dimension if we introduce a new level (Temperature Classification, for instance) in the S-Cubing tree and several rows in the base relation R, one for each temperature location and value. In general, S-Cubing approach is useful for many variations of the same problem.

FIG6.

Scale hierarchies and their challenges are detailed in (Bimonte, 2012). The S-Cubing approach can implement two different tree-based solutions for multi-scale data. The first considers an explicit hierarchy, where different spatial attributes of a base relation share a common hierarchy. For instance, the city of São Paulo in scale 1:10.000 and 1:100.000 and neighbors of São Paulo in scale 1:10.000. In the first solution, we consider an explicit hierarchy between both scales and São Paulo in scale 1:10.000 is used to produce São Paulo in scale 1:100.000, i.e., the spatial and statistical measures in scale 1:100.000 can be calculated from scale 1:10.000. The base relation illustrated in Figure 7 is an explicit scale hierarchy example, where a data cube similar to Figure 6 is computed using both shared dimensions and top-down/bottom-up computations.

FIG7.

The second S-Cubing multi-scale solution is illustrated in Figure 8. Instead of an explicit scale hierarchy, we implement one full data cube per scale and common GEO IDs are detected on the fly by S-Cubing tree traversal operations. In Figure 8, GEO IDs of temperature sensors 11, 14, and 51 from São Paulo city are commons in scales 1:10.000 and 1:100.000. In Belo Horizonte city no matching was found between scales 1:10.000 and 1:100.000, so no drill-down/roll-up operations are possible. We omit temperature values in Figure 8 to reduce the tree size, but each GEO ID on each node has a temperature value. The sub-tree “all” started from nodes 1:10.000 and 1:100.000 are also omitted to reduce the tree size. Drill-down and roll-up scale operations are performed by traversing multiple S-Cubing trees, but only equal and different lower hierarchy levels GEO IDs are highlighted in S-Cubing. In general, both solutions perform multi-scale queries in a transparent way to the end user.

FIG8.

Multiple complex spatial measures are implemented by S-Cubing approach. Each SC node or SC cell implement one or many measures, as illustrated in Figure 1 and explained in the beginning of Section 3. The permute operator can be easily implemented. Let´s use the temperature of Belo Horizonte and São Paulo cities example. The spatial measure temperature can become a dimension temperature if in a query we postpone the dimension temperature filter. A query “select the AVG(temperature) from city equal São Paulo, day equal 10, any time and sensor ID equal 111” has a path in S-Cubing tree. The path: São Paulo, GEO ID 01, day 10 and finally a temperature sensor 111. The temperature of sensor 111 is AVG((21.3+22.2) /2). In S-Cubing approach dimensions and measures are simply SC Nodes or SC Cells.

There are two strategies to compute a cube in S-Cubing: (i) based on shared dimensions calculus and (ii) based on neighborhood relationships calculus. The second strategy is a matrix based solution, where the concept of cells is introduced. A cell has dimension and measure attributes, similar to nodes. A cell does not have descendants, so it is correct to state that a SC Node is a subtype of SC Cell, where descendants are introduced. The S-Cubing based on neighborhood relationships calculus are used for unique scale maps. It receives a fine grained grid of cells with hundreds of thousands cells and a set of neighborhood relationships calculus. It is possible to apply the same neighborhood relationship calculus in all levels of the hierarchy or one different neighborhood relationship calculus per level of the lattice or any combination of previous scenarios. Nulls, continuous fields in dimensions, multiple complex measures, permute operator are also enable in S-Cubing based on neighborhood relationships calculus.

Figure 9 illustrates an example where a state of Brazil, named Minas Gerais (MG), is represented as regular grid cell with 9x9 cells. Figure 9-A represents the base cuboid and Figure 9-D the apex cuboid. This kind of representation is very useful in scenarios where the geo-object Minas Gerais state must be measured according to its position in the globe and not according to a geopolitical or any other feature (Ex. Cities, Streets, Neighbors, Salaries, telecom/internet infrastructure and many others). In general, both views are used together, i.e., a regular cell space representing irregular portions of the globe together with geo-objects representing many features or dimensions of a decision making process. The S-Cubing approach enables drill-down, roll-up, slice and dice operations in a data cube based on neighborhood relationship calculus.

FIG9.

A regular cell space can be implemented as a base relation, similar to the one illustrated in Figure 10. Each row is a grid cell, composed of a cell identification, cell location, cell size, and several dimensions and measures, as any other multidimensional database. In Figure 10-A, each cell is associated with a city, with a neighbor, has a time to collect temperatures, occurs on a specific day and has several measure values collected by temperature sensors in the globe. From a base relation, illustrated by Figure 10-A, S-Cubing generates the remaining data, illustrated by new relations (Figures 10-A and B).

The fine grained regular cell space, composed by 81 cells, has many cells associated with one city, but as the coarsely increase the number of cities associated with a unique regular cell also increases, as Figure 10 illustrates. In the example, Betim, Cataguases, Barbacena and Juiz de Fora are cities from Minas Gerais state. Shared dimension tree based solution and the neighborhood calculus matrix based solution are complementary. The end user are interested in evolving the temperature of geo-objects, like cities, over time, but sometimes is also useful to analyze a specific region, composed by several fragments of cities, for instance. A regular matrix of cells is an alternative to enable irregular region measuring. S-Cubing approach executes multiple neighborhood calculus, producing new thematic maps. End user can select several cells of any level of a neighbor hierarchy. In Figure 10, there is a single neighbor hierarchy [9x9, 3x3, 2x2, 1x1]. There are several ways to build a regular cell space from Minas Gerais state geo-object. “In real-world practice, scientists and practitioners register the values of a field by taking samples at (generally) fixed locations, and inferring the values at other points in space using some interpolation method. Thus, fields can be described by a function that indicates the distribution of the phenomena or feature of interest. The most popular discrete representation for fields is the raster model, where the 2D space is divided into regular squares.”(Gómez, 2012)

Fig.10

Algorithm 2 implements the cube lattice generation from a fine grained regular cell space, as Figure 9-A illustrates. The algorithm produces the remaining thematic maps with aggregated measures, similar to the shared dimension strategy explained before in this work. The difference is the automatic hierarchy generation using neighborhood relationship calculus, which produce aggregated results until a unique cell of the entire space is indexed.

%%%%%

Continuous dimensions, like time and location, can produce huge amount of data. They are stored as leaves in SPATIAL-Cubing tree. This way, it is possible to store all values individually, perform interpolation to provide user-friendly OLAP operations (drill-down, roll-up and others), and compute range queries like “*Select all GIDs AVG(temperature) from Brazil layer, between date/time X and Y, with humidity between W and K, from scales 1:10000 and 1:50000 and with distance less than 100km from Amazon river* ”.

Multiple scales are fundamental to perform realistic spatial analysis and are implemented on each SPATIAL-Cubing node, being used as a flag to avoid incorrect spatial operations. Multiple scales can also be stored as a SPATIAL-Cubing tree level, producing n different sub-trees, one for each scale. Finally, multiple scales can be implemented as multiple SPATIAL-Cubing tree levels, one tree level for each scale and the relationships among *GIDs* from different scales. Multiple spatial object association and any other *NxM* association are implemented in SPATIAL-Cubing tree, since it is a prefixed tree with suffix copies of the same data. Interpolation based on neighborhood relationships can be used to establish *GIDs* associations, where two sets of *GIDs* have two different scales and are stored on two different levels of SPATIAL-Cubing tree.

Spatial dimensions and measures have identical representation, but it is CPU costly to exchange dimensions with measures in SPATIAL-Cubing, therefore it is not implemented. An on-the-fly permute operation will require a complete scan of the SPATIAL-Cubing tree and several tree fusions and vice-versa.

SPATIAL-Cubing approach implements automatic spatial hierarchies based on neighborhood relationships. The space can be partitioned into regular cells of any scale. Each cell normally has its location and several other properties, like humidity, temperature and many others. Normally, each cell property is a measure of type *m*, defined previously in this paper. A SPATIAL Matrix is used to compute such cubes instead of a traditional SPATIAL-Cubing tree. Each SPATIAL Matrix cell has the same SPATIAL-Cubing tree node properties to turn the indexing phase independent from node or cell types.

SPATIAL-Cubing receives a matrix of cells Ma as input and performs a 3x3, 4x4, nxn sub-matrix scan of *Ma*. Figure 3 illustrates how to generate higher levels in the hierarchy using only the immediate lower level matrix. A 3x3 cell aggregation is selected, so first a 3x3 square is scanned from cell position (1,1) to (3,3) in the matrix Ma, as Figure 3-A illustrates. The next valid position to be processed is (1, 4), so a new 3x3 square of cells is scanned from matrix *Ma*, as Figure 3-B illustrates. The process continues until the entire matrix *Ma* is processed. A partial output is a new SPATIAL Matrix with 16 cells (4x4), as Figure 3-C illustrates.

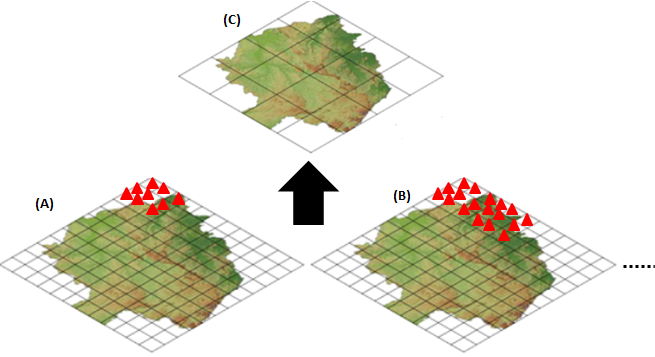
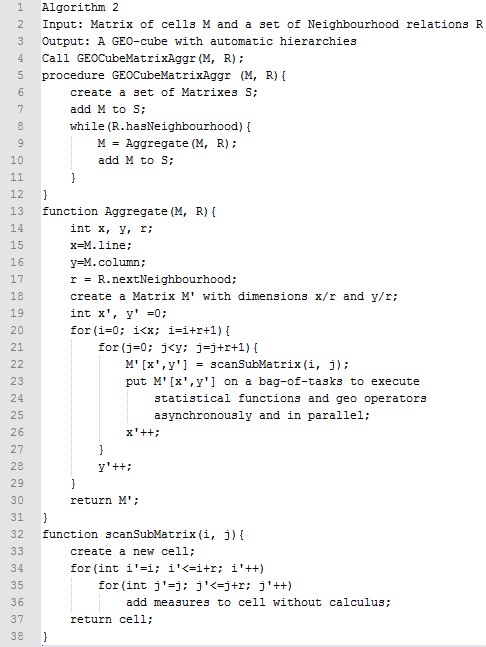


Figure 3: SPATIAL-Cubing aggregations in a regular cell space.

Algorithm 2 presents how SPATIAL-Cubing generates several SPATIAL Matrixes, one per hierarchy level. A set of neighborhood relationships is used to define the aggregation level and the number of hierarchies (line 10). Each matrix generation uses a fine grained matrix as input, scanning its cells (lines 20 and 21) using the neighborhood relation r as a parameter. Each sub-matrix with dimensions rxr is scanned to generate an aggregate cell (line 22). The aggregation step is done in two phases: first conventional and spatial measures are put together to be calculated further (line 22) and then an asynchronous call is made to perform statistical and spatial calculus over numbers and GIDs, respectively (lines 23, 24 and 25).

The aggregation level can vary in SPATIAL-Cubing. The first level adopts a 3x3 aggregation, but the second level uses a 2x2 cell aggregation. In algorithm 2, R stores {3, 2} values, representing both levels. Such flexibility is very useful for end users and it is implemented in SPATIAL-Cubing. All non-spatial measures are computed for each new SPATIAL Matrix cell. The set of SPATIAL Matrixes are managed by SPATIAL-Cubing, so any aggregated value of matrix level can be quickly retrieved by the kernel. Normally, each SPATIAL Matrix is a new layer to facilitate end users visualizations. SPATIAL-Cubing prototype adopts new layer per SPATIAL Matrix.

Algorithm 2: A SPATIAL-Cubing Matrix Aggregation.

A SPATIAL-Cubing tree node creation or a SPATIAL Matrix cell creation are CPU bound, thus very costly to be done sequentially. In this sense, for each created node all non-spatial measures must be updated using several numbers and a statistical function. Furthermore, a spatial operator must be applied to several sets of SPATIAL IDs. By far, spatial operators, like Union, Intersection, Touch and many others, demand more CPU than any other task, thus SPATIAL-Cubing implements both spatial operations and statistical calculus asynchronously and in parallel.

There is a bag-of-tasks (Kwan, 2010) in SPATIAL-Cubing responsible for notifying a pool of threads used to perform both spatial and statistical operations in parallel. Each thread receives the new node or cell and several numbers or sets of SPATIAL IDs from lower levels of the data cube lattice. At each thread, a spatial operation is performed using Spatial Tools (Turton, 2008) (Garnett, 2003) and the result is stored back into the new node or cell asynchronously. The same occur to statistical calculus using Apache library (Apache, 2013). Since there is no order in spatial operations or statistical calculus, some nodes or cells without lower level properties can be processed first. In this case, a thread removes a new task from the bag and puts the node or cell without lower level properties back into the bag. This chaotic environment scales well as our experiments demonstrate. The bag-of-tasks is thread-safe to avoid data corruptions. The bag-of-tasks calls are performed by both SPATIAL-Cubing algorithms, illustrated by lines 17, 18 of algorithm 1 and lines 23, 24 and 25 of algorithm 2.

# 4 Experiments

A comprehensive performance study was conducted to check the efficiency and the scalability of the proposed approach and prototype. We tested SPATIAL-Cubing prototype against a PostGis solution using materialized views. The SPATIAL-Cubing prototype and algorithms were coded in Java 64 bits (version 7.0). In all experiments, the relation can fit in main memory. Cube computation tests include both I/O and CPU times. I/O times are considered to load input relations from external memory to main memory. PostGis has only a sequential version, thus we set SPATIAL-Cubing with one thread to make comparisons.

We ran the prototypes in two Intel Xeon four-core processors with 2.4GHz each core, 10MB cache and 16GB of RAM DDR2 667MHz. The system runs Windows Server 2008 64 bits version. All tests were executed five times, we removed the lowest/highest runtimes and an average is calculated for the three remaining runtimes.

In the first experiment, we evaluated SPATIAL-Cubing scalability. A base relation R’ is composed by five spatial dimensions, one spatial hierarchy and two non-spatial measures. All possible queries were computed by SPATIAL-Cubing and PostGis solutions. R´ is defined as {county dimension, micro-regions dimension, macro-regions dimension, states dimension and regions dimension, number of dengue disease measure and number of person measure}. All spatial dimensions adopt the Union spatial operator. All non-spatial measures use SUM statistical function. The spatial hierarchy from the lowest dimension to the higher one is *sh={county => micro-regions => macro-regions => state => regions}*. We tested SPATIAL Cube with one, two, four and eight threads, as Figure 6 illustrates. SPATIAL-Cubing achieved a speedup of five in a machine with eight cores and eight SPATIAL-Cubing threads. Its efficiency is close to 100% with one thread and decreases up to 60% using eight threads. Efficiency is a measure of the fraction of time for which the cores are usefully employed.

Efficiency is defined as *E=speedup/cores*. In general, SPATIAL-Cubing speedup and efficiency results are promising. As the number of threads increases the efficiency decreases, since there is contention in memory system and there are always Operating Systems and some applications tasks concurrently using eight cores with SPATIAL-Cubing eight threads, turning memory exchanges at register level, cache level and RAM level a bottleneck. SPATIAL-Cubing becomes even more inefficient as the number of threads bypass eight threads.

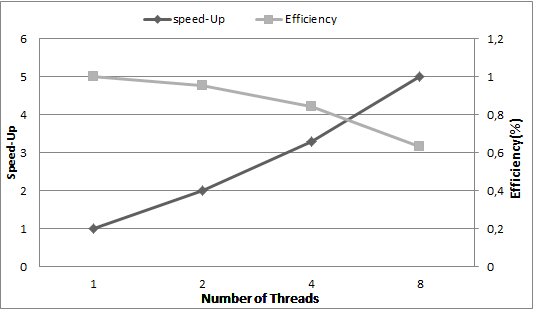
****

Figure 6: SPATIAL-Cubing scalability.

In the second experiment, we compared one thread SPATIAL-Cubing with a PostGis solution using materialized views stored only on RAM. Five simple base relations with two spatial measures and two non-spatial measures were used in the experiment. We tested how fast both solutions perform all x all SPATIAL IDs of Brazil counties, micro and macro regions, states and regions, using touch spatial operator together with union spatial operator, SUM of populations and SUM of number of Dengue diseases. The results are illustrated in Figure 7. In general, a sequential SPATIAL-Cubing prototype is 12-18% faster than a standard PostGis solution in generating spatial group-by operations. Cube C3 is the largest one, combining about 22 thousand counties. As the number of SPATIAL IDs becomes smaller, SPATIAL-Cubing x PostGis runtimes become similar. Cube C7 has similar runtimes, since both solutions group all regions of Brazil, which has less than 10 regions. Most related prototypes are built on top of a PostGis server, thus an experiment with PostGis is very useful.

A Map Cube (Shekhar, 2001) executable prototype was not obtained with the authors, thus no comparative studies were conducted.

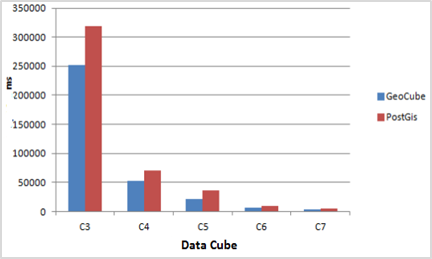


Figure 7: Comparative experiment with SPATIAL-Cubing and PostGis.

# Conclusions

In this paper we describe SPATIAL-Cubing approach and prototype. The SPATIAL-Cubing approach innovates in implementing a non-relational SOLAP approach that attempts most of novel SOLAP data model requirements, including multiple complex spatial measures, continuous dimension, scale hierarchy and missing values. SPATIAL-Cubing also innovates in implementing a spatial cube based on any neighborhood relationships, avoiding complex hierarchy definitions. This type of data cube proves to be very useful to decision makers and simulation tools. A solution using bag-of-tasks and several threads makes the SPATIAL-Cubing prototype execution parallel and asynchronous. A SPATIAL-Cubing prototype was implemented using World Wind (Bell, 2007), a very sophisticated 3D viewer for multiplatform. A comparison of SPATIAL-Cubing and a PostGis solution showed that SPATIAL-Cubing non-relational SOLAP solution becomes faster as the relation becomes bigger and sparser.

SPATIAL-Cubing future works include its integration with qCube approach (Silva, 2013), a high dimensional cube approach capable to answer aggregated range queries over 20, 50, 100 dimensions and several measures. A SPATIAL-Cubing for multicomputer architectures, similar to (Zhang, 2012), is a promising research direction. SPATIAL-Cubing using external memory and top-k spatial queries is part of SPATIAL-Cubing future plans. Updates strategies can become easier after qCube integration. The qCube approach pre-computes only 1D cuboids and generates the remaining aggregations on-the-fly. This way, various types of updates on dimensions, measures and hierarchies are supported by qCube.

**ACKNOWLEDGES**

This work was partially supported by UFOP, Capes and FAPEMIG.

**References**

Ahmed, T. O. and Miquel, M., 2005. *Multidimensional Structures Dedicated to Continuous Spatiotemporal Phenomena*. Proceedings of the 22nd British National Conference on Databases: Enterprise, Skills and Innovation. pp 29-40, 2005.

Alzate, J. C., Moreno, F. J., and Echeverri, J., 2012. *A spatio-temporal extension to the map cube operator.* In AIP Conference Proceedings(Vol. 1479, p. 2310).

Apache, 2014. *Apache Commons Math*, Accessed: 2nd October 2014. <http://commons.apache.org/proper/commons-math/>.

Bell, D. G., [Kuehnel, F.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Kuehnel,%20F..QT.&searchWithin=p_Author_Ids:37295459700&newsearch=true), [Maxwell, C.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Maxwell,%20C..QT.&searchWithin=p_Author_Ids:37295461300&newsearch=true), [Kim, R.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Kim,%20R..QT.&searchWithin=p_Author_Ids:37299791400&newsearch=true), [Kasraie, K.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Kasraie,%20K..QT.&searchWithin=p_Author_Ids:37295426600&newsearch=true), [Gaskins, T.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Gaskins,%20T..QT.&searchWithin=p_Author_Ids:37295426000&newsearch=true), [Hogan, P.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Hogan,%20P..QT.&searchWithin=p_Author_Ids:37295425600&newsearch=true), [Coughlan, J.](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Coughlan,%20J..QT.&searchWithin=p_Author_Ids:37299111600&newsearch=true), 2007. *NASA World Wind: Open Source GIS for Mission Operations.* Proc. of IEEE Aerospace Conference. Big Sky.

Bédard, Y. L. (1997). Étude de l'état actuel et des besoins de R&D relativement aux architectures et technologies des data warehouses appliquées aux données spatiales. Research Report for National Defence Canada Centre for Research in Geomatics.

Bédard, Y. M. (2001). Fundamentals of spatial data warehousing for geographic knowledge discovery. Miller, H., Han, J. (Eds.), Geographic Data Mining and Knowledge Discovery, pp. 53-73.

Bimonte, S., Bertolotto, M., Gensel, J., Boussaid, O., 2012. *Spatial OLAP and Map Generalization: Model and Algebra*. IJDWM, pp. 24-51.

Bimonte, S., Miquel, M., 2010. *When Spatial Analysis Meets OLAP: Multidimensional Model and Operators*. IJDWM, pp 33-60.

Bimonte, S., Wehrle P., Tchounikine, A., and Miquel, M., 2006. *GeWOlap: A Web Based Spatial OLAP Proposal*. In Second International Workshop on Semantic-based Spatialgraphical Information Systems. Springer-Verlag.

Bringay, S. B. (2011). Towards an on-line analysis of tweets processing. . *In Database and Expert Systems Applications. Springer Berlin Heidelberg.*, pp. 154-161.

Codd, E. F., 1968. *Cellular Automata*, Academic Press, Inc. New York and London.

Ferraz, V. R. T. and Santos, M. T. P., 2010. *Globeolap - improving the geospatial realism in multidimensional analysis environment*. In ICEIS (5)’10, pages 99-107.

Gómez, L.I., Gómez, S.A., Vaisman, A.A., 2012. *A generic data model and query language for spatiotemporal OLAP cube analysis*. EDBT 2012, pp. 300-311.

Gonzalez, H. H. (2006). Flowcube: constructing RFID flowcubes for multi-dimensional analysis of commodity flows. . *In Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment.*, pp. 834-845.

Gray, J., Bosworth A., Layman A. and Pirahesh H., 1996. *Data Cube: A Relational Operator Generalizing Group-By, Cross-Tab and Sub-Totals*. ICDE.

Garnett, J., 2003. *GeoTools Data Store Performance and Recommendations,* Program Manager GeoConnections.

Han, J. S. (1998). Selective materialization: An efficient method for spatial data cube construction. In Research and Development in Knowledge Discovery and Data Mining. Springer Berlin Heidelberg., pp. 144-158.

Han, J. , Pei, J., Dong, G. , Wang, K., 2001. Efficient computation of iceberg cubes with complex measures, In ACM SIGMOD International Conference on Management of Data, pages 1–12, 2001.

Jin, X. H. (2010). Visual cube and on-line analytical processing of images. *In Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 849-858.

Lima, J. D. C., Hirata, C. M., 2011. *Multidimensional cyclic graph approach: Representing a data cube without common subgraphs*, Information Sciences Journal. 181: 2626-2655.

Lin, C. X. (2008). Text cube: Computing ir measures for multidimensional text database analysis. . *IEEE ICDM'08.*, pp. 905-910.

Kaczmarski, K., 2011. *Comparing gpu and cpu in olap cubes creation, Proceedings of the 37th International Conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM'11, Springer-Verlag, Berlin, Heidelberg, pp. 308-319

[Kwan, S. K., Jogesh, K. M. J., 2010. *Bag-of-Tasks applications scheduling on volunteer desktop grids with adaptive information dissemination*, Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks.](http://dl.acm.org/citation.cfm?id=1990471&CFID=374829141&CFTOKEN=66244875)

Mondrian, 2014. *Pentaho Analysis Services: Mondrian.*  Accessed: 2nd October 2014. <http://mondrian.pentaho.org/>.

Moreira, A. A. , Lima, J. D. C. 2012. *Full and partial data cube computation and representation*

*over commodity PCs*., 'IRI' , IEEE, , pp. 672-679

Moreno F., Arango F., and Fileto R., 2009. *Extending the map cube operator with multiple spatial aggregate functions and map overlay*, Proc. of Geoinformatics, Fairfax.

Nienartowicz, Ł. (2013). OLAP Models for Sequential Data–Current State of Research and Open Problems. *In New Trends in Databases and Information Systems. Springer Berlin Heidelberg.*, pp. 415-423.

PostGIS, 2014. Spatial and Geographic objects for PostgreSQL. Accessed: 2nd October 2014. <http://postgis.net/>.

Sandro, B. and Anne Tchounikine, M. M., 2006. *Geocube, a multidimensional model and navigation operators handling complex measures: Application in spatial olap.* Advances in Information Systems, 4243/2006:100-109.

Shekhar, S., Lu, C., Tan, X., and Chawla, S., 2001. *Map Cube: A Visualization Tool for Spatial Data Warehouses*, pages 74-109. Harvey j. miller and jiawei han (eds.) edition.

Scotch, M. and Parmanto, B., 2005. Sovat: *Spatial olap visualization and analysis tool*. Hawaii International Conference on System Sciences, 6:142b.

Sboui, T., Bédard, Y., Brodeur, J., Badard, T., 2007. *A conceptual framework to suport semantic interoperability of geospatial datacubes*, Proceedings of ER/2007 SeCoGis Workshop, November5-9, Auckland, New Zealand, Lecture Notes in Computer Sciences, Springer, pp. 378-387. Springer Berlin/Heidelbert.

Silva, J. T. C., Rezende, J. F. V., Fidêncio, É., Melo, T. R., Neves, B., Lima, J. D. C. and Carneiro, T. G. S., 2012. *SIGHabitar – Business Intelligence Based Approach for the Development of Land Information Systems: The Multipurpose Technical Cadastre of Ouro Preto- Brazil,* Computational Science and Its Applications – ICCSA, V. 7334, pp. 302-315.

Silva, R. R., Lima, J. D. C. and Hirata, C. M., 2013. *qCube: Efficient integration of range query operators over a high dimension data cube.* Journal of Information and Data Management, Vol. 4, No. 3, October 2013, Pages 469–482.

Technologies, K., 2005. *JMAP Spatial OLAP: On-Line Analytical Processing for Spatial Databases*. Innovative technology to support intuitive and interactive exploration and analysis of spatio-temporal multidimensional data. 2005.

Turton, I., 2008.  *Open Source Approaches in Spatial Data Handling.* Springer Berlin Heidelberg. (pp. 153-169)

Wang, Y., Song, A. & Luo, J., 2010. *A mapreducemerge-based data cube construction method*, Proceedings of the 2010 Ninth International Conference on Grid and Cloud Computing, GCC '10, IEEE Computer Society, Washington, DC, USA, pp. 1-6.

Xin, D., Han, J., Li, X., Shao, Z. and Wah, B. W., 2007. *Computing Iceberg Cubes by Top-Down and Bottom-Up Integration: The Star-Cubing Approach*, IEEE Transactions on Knowledge and Data Engineering, 19(1): 111-126.

Yu, Y. L. (2009). iNextCube: Information network-enhanced text cube. . *Proceedings of the VLDB Endowment.*, pp. 1622-1625.

Zaamoune, M., Bimonte, S., Pinet, F., Beaune, F., 2013. *A New Relational Spatial OLAP Approach for Multi-resolution and Spatio-multidimensional Analysis of Incomplete Field Data*, ICEIS 2013, pp 145-152.

Zhang, J.,You, S. and Gruenwald, L., 2012. *High-performance Online Spatial and Temporal Aggregations on Multi-core CPUs and Many-core GPUs*. Proceedings of the Fifteenth International Workshop on Data Warehousing and OLAP, DOLAP 2012, pp 89-96.

Zhao, P. L. (2011). Graph cube on warehousing and OLAP multidimensional networks. *In SIGMOD Conference.*, pp. 853-864.