Benchmarking Spatial Data Warehouses

Thiago Luís Lopes Siqueira^{1,2}, Ricardo Rodrigues Ciferri²,
Valéria Cesário Times³, Cristina Dutra de Aguiar Ciferri⁴

¹ São Paulo Federal Institute of Education, Science and Technology, IFSP,
Salto Campus, 13.320-271, Salto, SP, Brazil

² Computer Science Department, Federal University of São Carlos, UFSCar,
13.565-905, São Carlos, SP, Brazil

³ Informatics Center, Federal University of Pernambuco, UFPE,
50.670-901, Recife, PE, Brazil

⁴ Computer Science Department, University of São Paulo at São Carlos, USP
13.560-970, São Carlos, SP, Brazil

prof.thiago@cefetsp.br, ricardo@dc.ufscar.br, vct@cin.ufpe.br, cdac@icmc.usp.br

Abstract. Spatial data warehouses (SDW) enable analytical multidimensional queries together with spatial analysis. Mainly, three operations are related to SDW query processing performance: (i) joining large fact tables and large spatial and non-spatial dimension tables; (ii) computing one or more costly spatial predicates based on spatial ad hoc query windows; and (iii) aggregating data according to different spatial granularity levels. Several techniques to improve the query processing performance over SDW have been proposed in the literature. However, we identified the lack of a benchmark to carry out a controlled experimental evaluation of such techniques and, principally, to effectively measure the costs of the aforementioned three complex operations. In this paper, we propose a novel spatial data warehouse benchmark, called Spadawan, to provide performance evaluation environments for SDW and enable a further investigation on spatial data redundancy. The Spadawan benchmark is available at <http://gbd.dc.ufscar.br/spadawan>.

Keywords: spatial data warehouse, benchmarking, performance evaluation, drill-down and roll-up operations.

1 Introduction

Spatial data warehouses (SDW) enable analytical multidimensional queries together with spatial analysis. A relational SDW inherits several components of conventional data warehouses, such as fact and dimension tables, numeric measures and hierarchies that aggregate these measures according to distinct granularity levels [1]. Additionally, the SDW has spatial attributes that store vector geometries and define spatially-enabled components, such as spatial dimension tables, spatial measures and spatial hierarchies [2][3][4]. Typically, a spatial hierarchy is a *predefined* 1:N association among higher and lower granularity spatial attributes that is determined by

a spatial relationship, e.g. containment, such as $(city) \preceq (address)$. As a result, spatial OLAP (SOLAP) operations are common roll-up and drill-down extended to hold spatial predicates [5]. Also, the well-known star and snowflake schemas may be adequately adapted to support the inclusion of spatial attributes, which introduce new storage costs and might impair query processing performance [3][6].

Mainly, three operations are related to SDW query processing performance: (i) joining large fact tables and large spatial and non-spatial dimension tables; (ii) computing one or more costly spatial predicates based on spatial ad hoc query windows; and (iii) aggregating data according to different spatial granularity levels. An example of a spatial and multidimensional query is “find out the total revenue earned by suppliers whose addresses are inside a rectangular window”. This query mentions a topological relationship and a spatial *ad hoc* query window that was not previously stored in dimension tables. Another query may be issued to roll-up to the city granularity level by using a larger window that intersects the cities where the suppliers are located, for instance.

Indices and materialized views are used to provide efficient query processing over SDW, and the requirements to evaluate their efficiency are datasets with different characteristics of data volume, data distribution and data types, as well as diverse types of query concerning their selectivity. The literature mentions benchmarks for decision support and data warehouses [7][8][9], and for spatial databases [10][11], synthetic spatial datasets generators [12] and real spatial datasets (e.g. Tiger/Line, see <http://www.census.gov/geo/www/tiger/>). However, using them to evaluate SDW query processing requires several adaptations to comprise spatial roll-up and drill-down operations, for instance. Therefore, there is a lack of a SDW benchmark to carry out a controlled experimental evaluation and, principally, to effectively assess the costs of the aforementioned operations.

In this paper, we propose a novel **spatial data warehouse benchmark**, called Spadawan, to address the query processing performance on spatial roll-up and drill-down operations using *predefined* spatial hierarchies over SDW. As spatial predicates, the Spadawan benchmark focuses on intersection, containment and enclosure range queries. Furthermore, it comprises redundant and non-redundant SDW schemas based on the Star Schema Benchmark (SSB) [8]. Consequently, the Spadawan benchmark provides a further spatial data redundancy investigation and comparison with a non-redundant SDW schema.

This paper is organized as follows. Section 2 surveys related work. Section 3 describes the SDW schemas of the Spadawan benchmark, while Section 4 describes data loading operations according to each schema. Section 5 presents the queries of the Spadawan benchmark and their particularities. Section 6 briefly describes a case study and Section 7 concludes the paper.

2 Related Work

Benchmarks for spatial databases [10][11] are not aimed at assessing the efficiency of SOLAP operations, although they focus on the spatial predicate computation. Regarding data warehouses, TPC-D is an obsolete benchmark for decision support

databases that does not support indices nor materialized views [7]. This fact motivated the proposal of the TPC-H [7], which provides individual queries that are not known in advance. However, its schema differs from the traditional star schema. The TPC-DS [9] suppresses this issue with a snowflake schema, but is aimed at data refreshing and its project is still under development. The SSB [8] extends the TPC-H to enable the analysis of historical trends and provides a set of predefined queries to run over its star schema. The SSB's queries refer to descriptive locations of suppliers and customers, since there is a predefined conventional hierarchy among attributes, i.e., $(region) \preceq (nation) \preceq (city) \preceq (address)$. However, the SSB does not hold spatial attributes nor stores maps that would enable multidimensional queries with spatial predicates, which is the focus of the Spadawan benchmark.

We argue that the SSB can be adapted to maintain spatial data and therefore provide spatial roll-up and drill-down operations evaluation, by reusing synthetic or real spatial datasets. This adaption requires maintaining the queries' semantics by adding spatial predicates and providing spatial predefined hierarchies based on the conventional existing ones. In this paper, we propose the Spadawan benchmark by extending the SSB to store a real spatial dataset and by altering the SSB's queries aiming at enabling spatial roll-up and drill-down operations evaluation.

3 The Spadawan Benchmark Schemas

We considered existing conceptual and logical models for SDW [2][3][4] in order to propose our SDW schemas, which extend the SSB schema by introducing spatial attributes that store geometries in spatial dimension tables, as shown in Fig. 1. The spatial attributes have the suffix *_geo* and are based on the SSB's conventional attributes that describe suppliers and customers locations, concerning their addresses, cities, nations and regions. We designed the *redundant* (Fig. 1a) and the *hybrid* (Fig. 1b) SDW schemas aiming at different purposes as follows.

According to Stefanovic et al. [3], *Customer* and *Supplier* should be considered as spatial-to-spatial dimension tables and must store all spatial attributes, as shown in Fig. 1a. Clearly, these spatial dimension tables maintain spatial data redundancy. For instance, the map for Europe is stored in every row whose supplier is located in Europe. Therefore, the *redundant* schema aims at investigating to what extent SOLAP queries performance is affected by spatial data redundancy.

On the other hand, Fidalgo et al. [4] state that, in SDW, spatial data must not be redundant and should be shared whenever is possible. Considering that the SSB's customers and suppliers share city, nation and region locations, but have individual addresses, we designed the *hybrid* schema (Fig. 1b) to comply with these characteristics that are not treated by the *redundant* schema. For instance, the *hybrid* schema's City spatial dimension table maintains distinct maps of cities where customers and suppliers reside. Therefore, the *hybrid* schema aims at evaluating the overhead of introducing additional joins costs to the query processing performance, as these joins are required to avoid spatial data redundancy.

The spatial data redundancy may also increase the number of tables to be scanned. Suppose that a spatial *ad hoc* query window intersects both customers and suppliers

cities geometries. Then, in a SDW with a redundant schema (as shown in Figure 1a), two tables would be scanned, while in a hybrid schema SDW (as given in Figure 1b), a single table storing all geometries for cities would be searched.

Finally, our extensions preserved descriptive data as well as created two spatial hierarchies based on the SSB's original conventional hierarchies. They are valid for both the *redundant* and the *hybrid* schemas: (i) $(region_geo) \preceq (nation_geo) \preceq (city_geo) \preceq (c_address_geo)$; and (ii) $(region_geo) \preceq (nation_geo) \preceq (city_geo) \preceq (s_address_geo)$. According to Malinowski and Zimányi [13], these hierarchies can be classified as simple symmetric spatial hierarchies with the containment spatial relationship. We emphasize that the *hybrid* schema is not a snowflake schema, since the latter normalizes hierarchies.

Fig. 1. The Spadawan benchmark schemas

[16] © 2009 Association for Computing Machinery, Inc. Reprinted by permission.

4 Data Generation and Loading

Loading data into the SDW schemas described in Section 3 requires running the SSB data generator as well as performing other tasks depending on the selected schema. Fig. 1 shows the data cardinality of each table according to the scale factor S chosen to generate the SSB dataset. Regarding suppliers and customers locations, there are always 5 distinct regions, 5 nations per region and 10 cities per nation. We determined that the spatial attributes that represent cities, nations and regions are polygons, which were reused from the Tiger/Line real dataset. On the other hand, customer and supplier descriptive addresses cardinalities depend on S , as well as the number of customers and suppliers per city. As for geographic addresses, they are

synthetic points randomly distributed inside each city polygon. We implemented a software to generate and distribute these points such that customers and suppliers have unique and distinct addresses. As a result, the spatial data volume of addresses varies according to S , as well as the quantity of customer and supplier addresses inside each city. Data sets that have already been used for populating the SDW redundant and hybrid schemas are available at <http://gbd.dc.ufscar.br/spadawan>.

The Spadawan benchmark's geometries do not suffer modifications after the data loading. Obviously, the same scale factor S and the same spatial dataset used for the *redundant* schema must be used for the *hybrid* schema in order to enable spatial data redundancy investigation. Section 4.1 and 4.2 describe, respectively, the data loading for the *redundant* and *hybrid* schemas, Section 4.3 discusses how to extend these schemas to increase spatial data volume and to decrease spatial predicate selectivity.

4.1 Loading the Redundant Schema

The following five steps must be performed to load the *redundant* SDW schema.

1. Load the geometries for cities, nations and regions into temporary tables.
2. Execute the SSB data generator with scale factor S and load its tables.
3. Run our generator of addresses, which also loads customer and supplier addresses into temporary tables.
4. Alter and update the tables *Customer* and *Supplier* to include the geometries of addresses, cities, nations and regions. Define all the constraints.
5. Discard all the temporary tables and build spatial indices supported by the DBMS (e.g. R-tree [14] or GiST [15]) on the spatial attributes.

4.2 Loading the Hybrid Schema

Loading the *hybrid* schema also requires five steps. Steps 1, 2 and 3 are similar to those described for the *redundant* schema. The remaining steps are defined as follows.

4. Alter and update the tables *Customer* and *Supplier* to include foreign keys referencing the spatial dimension tables, which are the altered temporary tables of steps 1 and 3.
5. Build spatial indices supported by the DBMS on the spatial attributes.

4.3 Increasing Data Volumes

The spatial data volumes for City, Nation and Region levels are fixed in the SSB. We argue that a fixed data volume for spatial data is unrealistic and should impose a severe drawback to Spadawan benchmark. In order to overcome this drawback, we describe the algorithm *IncreaseVolume* to enable increasing the spatial data volume and decreasing the spatial predicate selectivity. A high selectivity determines that most of the spatial objects are processed in the spatial predicate computation, while a low selectivity ensures that only a few of them is processed. The algorithm

IncreaseVolume consists of an intermediate step between the steps 2 and 3 presented in Sections 4.1 and 4.2, and can be used to load both *redundant* and *hybrid* schemas.

The algorithm *IncreaseVolume* generates a spatial data volume n times larger than that built with a given scale factor S . Translation (line 3) is an operation that shifts a given geometry to another location, according to chosen offsets. As a result, a translation modifies all coordinates of the geometry. Specifically, the translation used in the *IncreaseVolume* algorithm must assure that: (i) geometries of the same granularity level do not overlap; and (ii) the spatial hierarchy must not be deteriorated. For instance, if *city1* is a city and was replicated and translated, the copy of *city1* must not overlap other cities. Also, if *city1* is inside *nation1*, the copy of *city1* must be inside the copy of *nation1*.

Consider that: (i) $|X|$ is the cardinality of the spatial attribute X , i.e., the number of distinct objects that X can assume; (ii) *subj* is a spatial object for the attribute X ; (iii) *subj.id* is the identifier for the spatial object *subj*; and (iv) *subj'* is a copy of the spatial object *subj*. Then, the strategy to generate an identifier for *subj'* is to do: *subj'.id* $\leftarrow$ *subj.id* + $|X|$ (line 4). Analogously, the primary key values for replicated suppliers and customers can be determined (line 6). Regarding the spatial predicate selectivity, the commented lines (lines 7 and 8) must be executed when constant selectivity is desired. Otherwise, the original selectivity will be divided by n . We further discuss this issue in Section 5.1.

Algorithm IncreaseVolume	
1	For $i \leftarrow 1$ To $n-1$
2	Replicate the initial set of geometries
3	Translate the replicated geometries to new coordinates
4	Generate new identifiers for these geometries
5	Replicate the initial dataset of the dimension tables Customer and Supplier
6	Generate new primary key values for these customers and suppliers
7	/* Replicate the initial set of spatial query windows */
8	/* Translate these windows together with the replicated geometries */
9	End-For

5 Queries

5.1 Ad hoc Spatial Query Windows

Regarding the spatial query windows, they are quadratic, correlated with the spatial data, and considered *ad hoc* because their rectangles are not stored in any spatial dimension table. A spatial roll-up operation requires a set of four windows, each one associated to a granularity level (Address, City, Nation or Region) and has a specific size (as lower the granularity, smaller is the window). We defined two separate types of sets for the spatial query windows: disjoint and overlapping.

Regarding the type disjoint, consider a set of windows dI . Every window of dI has one centroid that is an address. To create dI 's windows, initially, one arbitrary address is chosen to be the centroid of the address window. Then, city, nation and region windows are produced subsequently by reusing the centroid of the address

window, as shown in Fig. 2a. Note that the query window size is proportional to the granularity level. In order to create another set of windows $d2$, the centroid for its windows is another address, specifically chosen to assure that the windows of $d2$ do not overlap any window of $d1$. As a result, all windows of different sets are disjoint, and the user can query distinct locations as previously fetched objects are not reused.

Concerning the type overlapping, consider a set of windows $o1$ whose windows were created similarly to $d1$. In order to create another set of windows $o2$, any point inside the address window of $o1$ is chosen to be the centroid of the new address, city, nation and region windows. As a result, all windows of different sets overlap, and the user can retrieve data related to a specific neighborhood, as shown in Fig. 2b. In fact, continuous-line windows were built using an address as centroid, while dashed-line windows had centroids obtained from any point inside the previous address window. The query window size is also proportional to the granularity level. Overlapping query windows were designed to evaluate the reuse of previously fetched objects, which is a task aided by system cache and buffers.

The Spadawan benchmark performs five roll-up/drill-down operations based on five fixed sets of disjoint query windows, as well as performs ten roll-up/drill-down operations based on ten fixed sets of overlapping query windows. Since the quantity of windows is fixed and they also have a fixed place, the number of spatial objects that satisfies the spatial predicate associated to a given window is also fixed. Therefore, replicating a set of windows together with spatial data, as described by the *IncreaseVolume* algorithm, maintains the spatial predicate selectivity constant. On the other hand, increasing only the spatial data volume by n times, divides the spatial predicate selectivity by n .

Fig. 2. Spatial ad hoc query windows.

5.2 Query Types 1, 2 and 3

Queries of type 1, 2 and 3 were based on query Q2.3 of the SSB and aim at evaluating the performance of: (i) at least three joins among tables, depending on the selected

SDW schema; (ii) four spatial predicates computation based on *ad hoc* spatial query windows; and (iii) data aggregation according to four spatial granularity levels.

Figure 3 illustrates how a single query was transformed into a spatial roll-up operation. We replaced conventional predicates that formerly referred to nominative locations by spatial predicates involving *ad hoc* spatial query windows. Instead of asking for a single descriptive granularity level, four queries of distinct spatial granularity levels are issued subsequently, considering that: R_A is a spatial relationship to evaluate supplier addresses against the spatial query window QW_A , R_C is a spatial relationship to evaluate cities against the spatial query window QW_C , R_N is a spatial relationship to evaluate nations against the spatial query window QW_N , and R_R is a spatial relationship to evaluate regions against the spatial query window QW_R . The size of the spatial query windows QW_A , QW_C , QW_N and QW_R are distinct and decreases as the granularity level decreases. This ensures a control of the selectivity factor of the queries in different granularity levels.

As a result, Query Types 1, 2 and 3 enable data aggregation according to the four aforementioned spatial granularity levels. Query Type 1 focuses mainly on the intersection relationship (i.e. IRQ: intersection range query on the spatial predicate), while Query Type 2 focuses mainly on the containment relationship (i.e. CRQ: containment range query on the spatial predicate) and Query Type 3 focuses mainly on the enclosure relationship (i.e. ERQ: enclosure range query on the spatial predicate).

Query Type 1 is detailed in Table 1. It uses the containment spatial predicate at the Address level and the intersection predicate at City, Nation and Region levels. The QW/Extent column shows the fraction of the extent occupied by the spatial query window. For instance, the query window for Address level represents 0.001% of the extent. Table 1 lists the average number of objects that are returned per query, considering 5 roll-up operations with the sets of spatial disjoint query windows and 10 roll-up operations with the sets of spatial overlapping query windows.

Table 1 shows the selectivity factor (SF), which consists of the conventional SF multiplied by the spatial SF . The former is fixed and defined by the SSB as 1/1000. The later is calculated by dividing the number of returned spatial objects by the spatial attribute cardinality. For instance, at City granularity level, the spatial SF is 3.6/250 and therefore the query SF is $1/1000 * 3.6/250$ (value of 0.0000144). Only one spatial SF was defined at Nation level to assess the efficiency when no spatial objects are returned as query answer (Table 2). This represents an extreme situation on query processing.

It is not possible to estimate the number of addresses that satisfies the spatial predicate, since the address data volume and the number of addresses inside each city depend on the scale factor S used to generate the SSB dataset. Therefore, we estimated the number of objects retrieved by the query as well as the SF for the Address level using the data generation scale factor of 1.

Query Types 2 and 3 are detailed in Tables 2 and 3, respectively, and evaluate other spatial predicates using different sizes of query windows. We emphasize that all buffers and cache must be flushed at the end of each spatial roll-up operation that utilize spatial disjoint query windows. On the other hand, they must not be flushed when utilizing overlapping spatial query windows.

Fig. 3. The template for Query Types 1, 2 and 3.

[16] © 2009 Association for Computing Machinery, Inc. Reprinted by permission.

Table 1. Additional information for Query Type 1.

Level	Predicate R	QW/Extent	Disjoint Query Windows		Overlapping Query Windows	
			Objects/query	<i>SF</i>	Objects/query	<i>SF</i>
Address	$R_A = \text{CRQ}$	0.001%	2.2	0.00000022	5.4	0.00000054
City	$R_C = \text{IRQ}$	0.05%	3.6	0.0000144	4.0	0.000016
Nation	$R_N = \text{IRQ}$	0.1%	1.6	0.000064	3.0	0.00012
Region	$R_R = \text{IRQ}$	1%	1.2	0.00024	2.0	0.0004

Table 2. Additional information for Query Type 2.

Level	Predicate R	QW/Extent	Disjoint Query Windows		Overlapping Query Windows	
			Objects/query	<i>SF</i>	Objects/query	<i>SF</i>
Address	$R_A = \text{CRQ}$	0.01%	19.0	0.0000019	37.0	0.0000037
City	$R_C = \text{CRQ}$	0.1%	1.4	0.0000056	3.0	0.000012
Nation	$R_N = \text{CRQ}$	10%	1.2	0.000048	0.0	0.0
Region	$R_R = \text{CRQ}$	25%	0.4	0.00008	1.0	0.0002

Table 3. Additional information for Query Type 3.

Level	Predicate R	QW/Extent	Disjoint Query Windows		Overlapping Query Windows	
			Objects/query	<i>SF</i>	Objects/query	<i>SF</i>
Address	$R_A = \text{CRQ}$	0.00001%	1.0	0.0000001	1.0	0.0000001
City	$R_C = \text{ERQ}$	0.0005%	0.8	0.0000032	1.0	0.000004
Nation	$R_N = \text{ERQ}$	0.001%	0.8	0.000032	1.0	0.00004
Region	$R_R = \text{ERQ}$	0.01%	1.0	0.0002	1.0	0.0002

5.3 Query Type 4

Query type 4, shown in Fig. 4, was based on the SSB's query Q3.3 and consists of a spatial roll-up and spatial drill-down operations with two ad hoc spatial query windows, which add an extra high join cost. Basically, this query retrieves “the revenue per year per brand for suppliers of an area x to the customers of an area y ”. The same granularity level is used for both customers and suppliers simultaneously. The containment spatial predicate is verified at Address level while the intersection

predicate is verified at City, Nation and Region levels. Table 4 shows additional details.

Fig. 4 Query Type 4.

Table 4. Additional information for Query Type 4.

Level	Predicate	QW/Extent	Disjoint Query Windows		Overlapping Query Windows	
			Objects/query	SF	Objects/query	SF
Address	CRQ	0.001%	9.1	0.00000091	11.3	0.00000114
City	IRQ	0.05%	7.2	0.0000288	9.0	0.000036
Nation	IRQ	0.1%	3.2	0.000128	5.0	0.0002
Region	IRQ	1%	2.4	0.00048	3.0	0.0006

6 Case Study

We have already used the Spadawan benchmark to investigate the impact of spatial data redundancy over SDW [6]. We loaded the following datasets: D1: the redundant schema using the scale factor $S = 10$, which occupied 150 GB; D2: the hybrid schema with $S = 10$, which occupied 15 GB; D3: the hybrid schema with $S = 6$; and D4: the hybrid schema with $S = 2$. Regarding City, Nation and Region levels, the spatial data volume remained fixed as well as the spatial predicate selectivity. The Address level data volume varied according to S .

We performed five spatial roll-up operations, using the five sets of disjoint query windows, and collected the average elapsed time at each granularity level. The GiST index was defined over the spatial attributes to enhance the spatial predicate computation. Experiments were conducted on a computer with a 2.8 GHz Pentium D processor, 2 GB of main memory, a 7200 RPM SATA 320 GB hard disk, Linux CentOS 5.2, PostgreSQL 8.2.5 and PostGIS 1.3.3.

Table 5 shows the results obtained for the datasets D1, D2, D3 and D4 for Query Type 1. It is important to observe that: (i) the spatial data redundancy drastically impaired query processing performance especially at Nation and Region levels whose cardinalities are lower; and (ii) the smaller the conventional data volume, the shorter the elapsed time to process the queries over the hybrid schema. Spatial data redundancy impaired not only the query processing performance, but also the storage requirements, since D1 occupied ten times more space than D2.

Another interesting issue was raised by evaluating Query Type 4 against the dataset D1. At Region and Nation granularity levels, we aborted the query processing

after 4 days of execution, since this elapsed time was prohibitive. At City level, the query took 172,900.15 seconds (approximately 48 hours). On the other hand, the same query issued against the dataset D2 took only 130.34 seconds, i.e., the spatial data redundancy provided an unacceptable increase of 132,900.00%.

We have developed the Spatial Bitmap Index (SB-index) [16] in order to decrease the query response time in SDW. The SB-index was also validated using the Spadawan benchmark. For further details about the performance evaluation, see [16].

Table 5. Elapsed times in seconds for Query Type 1.

	D1	D2	D3	D4
Address	2831.23	2853.85	1803.62	594.31
City	2773.10	2758.70	1686.61	562.08
Nation	3449.76	2765.61	1694.00	545.59
Region	6200.44	2790.29	1703.31	552.94

7 Conclusions and Future Work

This paper proposed a novel benchmark for spatial data warehouses, called Spadawan, whose main characteristics are: (i) it generates SDW datasets composed of points and polygons in spatial attributes; (ii) it is composed of different types of SOLAP queries that enable the performance evaluation of intersection range queries, containment range queries and enclosure range queries in the spatial predicate; (iii) it enables the evaluation of spatial roll-up and drill-down operations; (iv) it provides a means of investigating spatial data redundancy in SDW by designing two distinct data schemas with spatial hierarchies and spatial dimensions; (v) it permits the adjustment of the SDW data volume and the spatial predicate selectivity; and (vi) it uses spatial query windows that may overlap each other or may be disjoint from each other. We validated the Spadawan benchmark using it to generate a performance evaluation environment to assess the impact of spatial data redundancy over SOLAP queries [6] and the efficiency of the SB-index data structure [16].

As future work, we intend to propose additional SOLAP query types to analyze drill-across operations on extended SDW schemas and to compute aggregations of geometries of spatial objects. We also plan to incorporate different spatial data, such as lines, polygons with holes and with islands, on the spatial data generation and SOLAP query processing. Another future work would be to extend the current benchmark by covering all types of classification hierarchies in addition to the predefined 1:N. The use of the Spadawan benchmark with different techniques, such as indices and materialized views, is another future work.

Acknowledgements. This work has been supported by the following Brazilian research agencies: FAPESP, CNPq, CAPES, INEP and FINEP. The first two authors thank the support of the Web-PIDE Project in the context of the Observatory of the Education of the Brazilian Government. The work carried by the third author was supported by funds from the CNPq under the Grant 479018/2009-0. The last author's work has been founded by FAPESP under the Grant 2009/06052-7.

References

1. Kimball, R., Ross, M.: *The data warehouse toolkit: the complete guide to dimensional modeling*, John Wiley & Sons, Inc., 2002.
2. Malinowski, E., Zimányi, E.: *Advanced data warehouse design: from conventional to spatial and temporal applications (data-centric systems and applications)*, Springer, 2008.
3. Stefanovic, N., Han, J., Koperski, K.: "Object-based selective materialization for efficient implementation of spatial data cubes", *IEEE Trans. Knowl. Data Eng.*, 12(6): 938-958, 2000.
4. Fidalgo, R., Times, V.C., Silva, J., Souza, F.F.: "GeoDWFrame: a framework for guiding the design of geographical dimensional schemas", In *DaWaK*, pages 26-37, 2004.
5. Rivest, S., Bédard, Y., Proulx, M., Nadeau, M., Hubert, F., Pastor, J.: "SOLAP technology: merging business intelligence with geospatial technology for interactive spatio-temporal exploration and analysis of data", *J. of Photogrammetry and Remote Sensing*, 60:17-33, 2005.
6. Siqueira, T.L.L., Ciferri, C.D.A., Times, V.C., Oliveira, A.G., Ciferri, R.R.: "The impact of spatial data redundancy on SOLAP query performance", *J. Braz. Comp. Soc.*, 15(2):19-34, 2009.
7. Poess, M., Floyd, C.: "New TPC benchmarks for decision support and web commerce", *SIGMOD Record*, 29(4):64-71, 2000.
8. O'Neil, P., O'Neil, E., Chen, X., Revilak, S.: "The star schema benchmark and augmented fact table indexing", In *TPCTC*, pages 237-252, 2009.
9. Poess, M., Smith, B., Kollar, L., Larson, P.: "TPC-DS, taking decision support benchmarking to the next level", In *SIGMOD*, pages 582-587, 2002.
10. Paton, N.W., Williams, M.H., Dietrich, K., Liew, O., Dinn, A., Patrick, A.: "VESPA: a benchmark for vector spatial databases", In *BNCOD*, pages 81-101, 2000.
11. Günther, O., Oria, V., Picouet, P., Saglio, J., Scholl, M.: "Benchmarking spatial joins à la carte", In *SSDBM*, pages 32-41, 1998.
12. Theodoridis, Y., Silva, J.R., Nascimento, M.A.: "On the generation of spatiotemporal datasets", In *SSD*, pages 147-164, 1999.
13. Malinowski, E., Zimányi, E.: "Spatial hierarchies and topological relationships in the spatial MultiDimER model", In *BNCOD*, pages 17-28, 2005.
14. Guttman, A.: "R-trees: a dynamic index structure for spatial searching", In *SIGMOD*, pages 47-57, 1984.
15. Aoki, P.M.: "Generalizing "search" in generalized search trees", In *ICDE*, pages 380-389, 1998.
16. Siqueira, T.L.L., Ciferri, R.R., Times, V.C., Ciferri, C.D.A.: "A spatial Bitmap-based index for geographical data warehouses", In *ACM SAC*, pages 1336-1342, © 2009 ACM, Inc. <http://doi.acm.org/10.1145/1529282.1529582>