

APPROXIMATE SPATIAL QUERY PROCESSING USING RASTER SIGNATURES

Leonardo Guerreiro Azevedo¹, Rodrigo Salvador Monteiro¹, Geraldo Zimbrão^{1,2} and Jano Moreira de Souza^{1,2}

¹Computer Science Department, Graduate School of Engineering, Federal University of Rio de Janeiro; ²Computer Science Department, Institute of Mathematics, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

Abstract: Nowadays, the database characteristics, such as the huge volume of data, the complexity of the queries, and even the data availability, can demand minutes or hours to process a query. On the other hand, in many cases it may be enough to the user to get a fast approximate answer, since it has the desired precision. The challenge to give to the user an exact query answer within a reasonable time becomes even bigger in the spatial database field. This work proposes the use of the Four Color Raster Signature (4CRS) for approximate query processing. The main goal is to reduce the time required to process a query executing it on approximate data (4CRS signature) instead of accessing the real datasets. The experimental tests demonstrated the good results of our proposal. Considering the test of the most important algorithm, the time required to process an approximate query answer has average of 7.22% of the time to get an exact answer, the disk accesses have average of 7.04% and the average error is 1% related to exact processing. Besides the 4CRS storage requirements are also quite small, which has an average of only 3.57% of the space required to store the real datasets.

Key words: Spatial Queries; Approximate Query Answer; Raster Signature.

1. INTRODUCTION

The increase of storage capacity and the decrease of hardware costs have made possible for applications to deal with large amount of data, involving Gigabytes, Terabytes and even Petabytes of information. Such data are often stored in hard disks or tapes. Thus, accessing a simple piece of information may take a very long time, because disk access time is something between

10^4 and 10^7 times slower than memory access time. On the other hand, it is important that applications answer to user requests in a short period of time. The challenge in answering such requests becomes even bigger within spatial data environments, where the data have a high complexity.

Spatial data consists of spatial objects made up of points, lines, regions, rectangles, surfaces, volumes, and even data of higher dimension which includes time (Samet, 1990). Examples of spatial data include cities, rivers, roads, counties, states, crop coverage, mountain ranges, parts in a CAD system, etc. It is often desirable to attach spatial with non-spatial attribute information. Examples of non-spatial data are road names, addresses, telephone numbers, city names, etc. Since spatial and non-spatial data are so intimately connected, it is not surprising that many of the issues that need to be addressed are in fact database issues.

Spatial DBMS (Database Management Systems) provides the underlying database technology for Geographic Information Systems (GIS) and other applications (Güting, 1994). There are numerous applications in spatial database systems area: traffic supervision, flight control, weather forecast, urban planning, route optimization, cartography, agriculture, natural resources administration, coastal monitoring, fire and epidemics control (Aronoff, 1989; Tao et al., 2003). Each type of application deals with different features, scales and spatiotemporal properties.

Query processing optimization in spatial databases is not a trivial problem. It is, actually, in the databases field, a big challenge to give the user a response within a reasonable time. Faloutsos et al. (1997) point different situations, besides the large amount of data that make it harder to meet the users' needs. There are, for instance, cases in which data is stored in third part devices or in remote bases and are temporarily not available, increasing reasonably the response time. Old data can be disposed in order to make room for new ones. Therefore, it becomes impossible to answer queries on deleted information.

Traditional query processing has focused on providing exact answer to queries, in such a way that seeks to minimize response time and maximize throughput (Gibbons et al., 1997). However, the huge volume of data, the complexity of the query, and the availability of the data can demand minutes or hours to process the query. In many cases, it is enough for the user to get an approximate and fast answer for the query instead of the exact and slower answer. For instance, when the query requests numerical answers, and the full precision of the exact answer is not needed, e.g., a total, average, or percentage for which only the first few digits of precision are of interest (such as the leading few digits of a total in the millions, or the nearest percentile of a percentage) (Gibbons et al., 1997).

Environments where providing an exact answer results demands an undesirable response times motivate the study of techniques for providing approximate answers to queries. The goal is to provide estimated answers in orders of magnitude less time than the time to compute an exact answer, by avoiding or minimizing the number of accesses to base data (Gibbons et al., 1997). An important application of these techniques is query optimization: to estimate plan costs it is necessary very fast response times but not exact answers (Ioannidis and Poosala, 1995). Also, as an emerging application we can cite spatial OLAP (Papadias et al., 2001).

This work presents a new approach for approximate query processing on spatial data. We propose the use of the Four Color Raster Signature (4CRS) (Zimbrao and Souza, 1998) for computing fast and approximate, but with acceptable precision, answers for queries on polygon datasets. Instead of accessing the real data, the query is executed on the 4CRS polygon signatures. The 4CRS stores the main data characteristics in an approximate and compact representation that can be accessed and processed faster than the real data. As a result, the approximate query answer will be available faster than the exact answer. A precision measure is also returned in order to provide a confidence interval. In general, this approximate answer will be enough to the user to make a decision, taking a much shorter execution time than it would be if the exact answer were processed.

This paper is divided in sections, as follows. Section 1 is this introduction. Section 2 surveys the related literature. In Section 3, we present some 4CRS characteristics and the approach of its use for approximate query processing on spatial data. Section 4 is dedicated to the experimental test. Finally, in the Section 5, we present our conclusions.

2. RELATED WORK

There are many approaches on the approximate query processing area. The works presented by Barbará et al. (1997) and Han and Kamber (2001) are good surveys of these techniques. They show the efficiency of the techniques, w.r.t. the data types being reduced. In both works, classifications are proposed. However, the one suggested by Han and Kamber is wider and thereby is going to be used to list some works in this research field:

- Data cube aggregation: involves the use of aggregation operation in the data cube construction process. Sarawagi and Stonebraker (1994), Agarwal et al. (1996), Ross and Srivastava (1997) present algorithms for data cube construction processes and their pre-computation. Cannataro et al. (2002) propose data cube aggregations for summarizing XML documents;

- Dimensionality Reduction: reduces the data set size by removing irrelevant or redundant attributes. Kohavi and John (1997) present a wrapper approach for selecting the best attributes to be used in a particular algorithm and in a specific domain. Dash et al. (1997) present a method based on entropy measurement for selecting attributes on unsupervised data.
- Data compression: data encoding or transformations are applied so as to obtain a reduced or “compressed” representation of the original data. One of most known techniques is the wavelet. The use of this data approximation technique was studied at first by Matias et al. (1998). Algorithms for query processing on wavelet coefficients of relational data are presented in Chakrabarti et al. (2000). Another techniques in this data compression subject are Qanticubes (Furtado and Madeira, 2000a) and FCompress (Furtado and Madeira, 2000b). Qanticubes is a technique for data compression on data cubes of data warehouses and FCompress is for compression of fact tables of data warehouses.
- Numerosity Reduction: the data is replaced or approximated by “smaller” forms of data representation. Han and Kamber (2001) divide these techniques in parametric and nonparametric. For parametric methods, a model is used to estimate the data, so that only the data parameters need be stored, instead of the actual data. Regression and log-linear models (Johnson and Wichern, 1992) are examples of parametric methods. Nonparametric methods store reduced representations of the data. Histograms (Poosala et al., 1996), cluster and index structures, (Aoki, 1998) and sampling (Poosala et al., 1996; Furtado and Madeira, 1999) are examples of nonparametric methods;
- Discretization and Concept Hierarchy Generation: the data is reduced by collecting and replacing low-level concepts by higher-level ones or dividing the range of the attribute into intervals. Many techniques are presented in Han and Kamber (2001).

In spite the existences of many approaches on the approximate query processing area, most of them are for relational data. Roddick et al. (2004) stand out that in several spatio-temporal applications, the size of the data and the high frequency of updates impose the use of approximate processing. For instance, for processing data streams which are potentially unbounded in size. Furthermore, even if all data were stored, the size of the index would render exact query processing very expensive. Finally, in several applications the main focus of query processing is retrieval of approximate summarized information about objects that satisfy some spatio-temporal predicate (e.g, ‘the number of cars in the city center 10 minutes from now’), as opposed to exact information about the qualifying objects (i.e., the car ids), which may be unavailable, or irrelevant. Therefore, researching new techniques that support the uniqueness of spatial data became a major issue

in the database field. In this work, a new approach for approximate query processing in Spatial Database subject will be present.

3. FOUR COLOR RASTER SIGNATURE AND APPROXIMATE QUERY PROCESSING

This section aims to presenting the use of the Four Color Raster Signature (4CRS) (Zimbrao and Souza, 1998) in approximate query processing. The main idea is to execute the query on the 4CRS representation of the polygons, instead of the real dataset. The 4CRS stores the main characteristics of the data in an approximate and compact representation that can be accessed and processed faster than the real data. As a result, the required time to compute the approximate query answer will be much shorter than the time to get the exact answer. On the other hand, the answer is estimated and not exact. However, a confidence interval is also returned in order to show the distance between the approximate and the exact answer. In general, the approximate answer will be enough for the user to make his decision since it has a short execution time and the desired accuracy.

The 4CRS signature is a raster approximation. A raster approximation is an object representation upon a grid of cells. Each cell stores relevant information using few bits. The grid resolution can be changed in order to obtain a higher resolution representation or a more compact one. Using many cells the approximation will be more precise, but it will require more storage space. On the other hand, using few cells, the approximation will be compact, but query answers will be less accurate. This section is divided in sub-sections as follow. The Sub-section 3.1 is dedicated to 4CRS signature. In the Sub-section 3.2 the algorithm for computing the grid of cells is presented. Finally, in the Sub-section 3.3 the use of 4CRS in approximate query processing is presented in more details.

3.1 Four Color Raster Signature (4CRS)

The 4CRS (Zimbrao and Souza, 1998) is used for representing polygons, and it is a small bit-map using four colors. Each color represents an intersection type between the object and the cell (Table 1). In Figure 1, an example of 4CRS is presented.

Table 1. Types of 4CRS cells

Bit value	Cell type	Description
00	Empty	The cell is not intersected by the polygon
01	Weak	The cell contains an intersection of 50% or less with the polygon
10	Strong	The cell contains an intersection of more than 50% and less than 100% with the polygon
11	Full	The cell is fully occupied by the polygon

In order to represent the bit-map compactly, displacement vectors representing the boundary of the polygon are stored in disk. In addition, the cell type for each cell traversed by the displacement vectors is also stored. The only cell types stored are Weak and Strong, because no cell traversed by the border line can be Full or Empty. Therefore, this information can be represented using only one bit. The bit-map can be reconstructed applying a simple algorithm on the stored information discovering which cells are Empty (outside the polygon) and which are Full (inside the polygon). More details can be found in (Zimbrao and Souza, 1998; Monteiro et al., 2004).

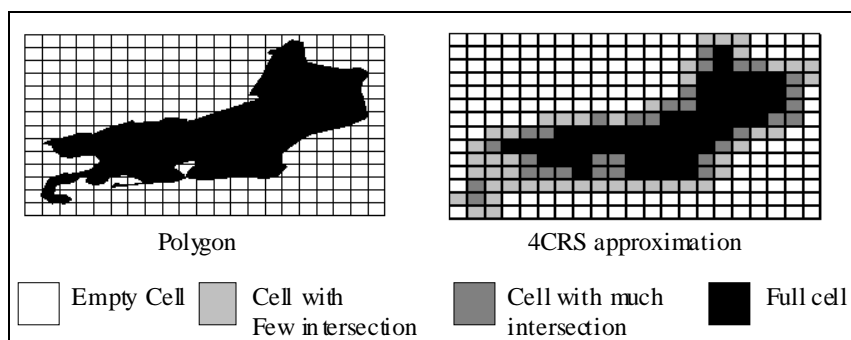


Figure 1. Example of 4CRS signature

3.2 Space Division in Cells

Raster approximations are constructed upon grids of cells. When testing the cells of two objects, the overlapping cells of their approximations must be compared. However, only cells of same size and that overlap perfectly can be compared. In order to conform to these requirements, the generation of grids must follow some pattern. If such requirements are not met, it becomes impossible to compare two approximations, as presented in Figure 2-a. Therefore, the space must be divided into cells independently of the object position. There will be a universal grid, that is, the coordinate system determines the grid. An algorithm specifying a pattern for computing raster approximations is presented in (Zimbrao and Souza, 1998; Azevedo et al., 2003) which we explain in this sub-section.

The requirements can be achieved if we constrain that the length of each cell side be a power of two (2^n), and that the vertices of each cell be a multiple of the same power of two ($a \times 2^n$) in the coordinate system. By doing so, we ensure that, if two cells of the same size overlaps each other, then they are perfectly superimposed to one another (Figure 2-b).

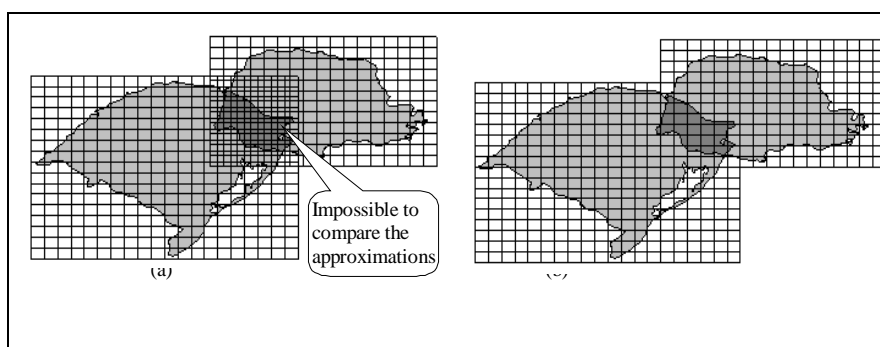


Figure 2. (a) Grids of same size not overlapping perfectly. (b) Perfect overlap permitting comparison.

The space is decomposed in a $p \times q$ cells grid with a 2^n size, which corresponds to the MBR- 2^n of the object. The MBR- 2^n vertices are $(2^n a_0, 2^n b_0)$ and $(2^n a_p, 2^n b_q)$, where a_0, a_p, b_0, b_q and n are integers. Besides, n is chosen so that $(a_p - a_0)(b_q - b_0) \leq N$, where $(a_p - a_0)$ is the number of cells in the axis x , $(b_q - b_0)$ is the number of cells in the axis y and N is the maximum number of grid cells. N is chosen so that the average size of the approximations results in a tree with good performance results. A good choice is try to keep the approximation size close to 3 or 4 times the MBR size. For example, if each entry will use an average of 80 Bytes, a 16 KB page will accommodate 100 to 200 entries and a huge dataset (1000 K objects) leads to an R-Tree of just 3 or 4 levels.

As shown in Figure 3, the MBR- 2^n is computed based on the object MBR, truncating its coordinates to powers of 2. The grid of cells is represented by the points $2^n a_0, 2^n a_1, \dots, 2^n a_p$, that determine a set of parallel lines to the vertical axis, and the points $2^n b_0, 2^n b_1, \dots, 2^n b_q$, that determine a set of parallel lines to the horizontal axis.

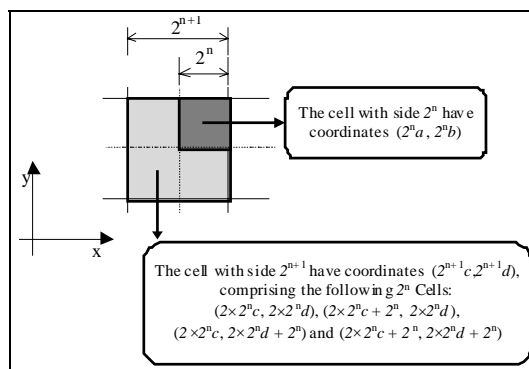


Figure 3. - Line up of Cell corners.

3.3 Approximate query answering using 4CRS

The 4CRS signature was first used to improve the processing of spatial joins of polygon datasets, reducing the need for examining the exact geometry of spatial objects to find the intersecting ones (Zimbrao and Souza, 1998). It was employed as a filter in the second step of the Multi-Step Query Processor (MSQP) (Brinkhoff *et al.*, 1993) and the results showed that 4CRS, when compared to other approaches, reduced the inconclusive answers by a factor of more than two. As a result, the need for retrieving the representation of polygons and carrying out exact geometry tests was reduced by a factor of more than two. The 4CRS was also used for intersection tests between polylines and polygons datasets (Monteiro *et al.*, 2004) and the results were also very good. The experiments performed with real data sets resulted in performance gains validating approach effectiveness. The number of exact intersection tests was reduced by 59%. The overall execution time and number of disk accesses were both reduced by 48%.

The 4CRS characteristics and the good results obtained using 4CRS for approximating polygons motivated the employment of it on approximate query processing. Consequently, new algorithms must be designed and implemented to concern the requirements of this area. Gibbons *et al.* (1997) present five metrics to evaluate approximate query engines:

- Coverage: the range of queries for which approximate answers can be provided.
- Response time: the time to provide an approximate answer for a query.
- Accuracy: the accuracy of the answers provided, and the confidence in that accuracy.
- Update time: the overheads in keeping its synopses up-to-date.
- Footprint: the storage requirements for its synopses.

Selection and join queries are fundamental operations in any SDBMS. A spatial selection retrieves from a dataset the entries that satisfy some spatial predicate with respect to a reference object q . The most common type of spatial selections is window queries, where the predicate is overlap and q defines a window in the workspace (e.g., ‘find all lakes that are intercepted by a city’). A spatial join operation selects from two object sets, the pairs that satisfy some spatial predicate, usually intersect (e.g., ‘find all cities that are crossed by a river’) (Papadias *et al.*, 1999). However, usually the user is interested in some properties of the objects that intercept each other, and not only to know what objects have intersection. Besides, these properties must be processed so fast as the intersect predicate. In this work, we present algorithms for computing the approximate area property. Instead of accessing the object real representation, the 4CRS signature is used to return an approximate answer faster.

In this work we developed and tested algorithms using 4CRS signature for approximate query processing for the following queries:

- Polygon approximate area (Sub-section 3.4)
- Approximate area of polygon x window intersection (Sub-section 3.5)

However, the algorithms can be extended or new algorithms may be developed in order to answer other kinds of approximate query processing, such as:

- Approximate area of polygon x polygon intersection
- Distance
- Buffer
- Perimeter
- Topological queries: there are eight topological relationships among pairs of regions, based on the intersections of their topological interiors, boundaries, and exteriors (Papadimitriou *et al.*, 1996). These mutually exclusive relations are: overlaps, disjoint, equal, meets, contains, covers, not contains, not covers.

3.4 Algorithm for polygon approximate area

Given a raster signature, the algorithm for computing polygon approximate area is as follows. We must sum the estimated polygon area inside each cell of the grid. It is easy to see that empty cells and full cells have 0% and 100% of its area intersecting the polygon, respectively. Weak and strong cells need a different approach. The polygon area inside a weak cell seems to be uniformly distributed in the open interval (0%, 50%). In fact, it is not unrealistic to assume it since the cell corners is completely independent of polygons coordinates. So, the average polygon area inside a weak cell is assumed to be 25% - as confirmed by empirical tests. For the

same reason, the average polygon area inside a strong cell is 75%. One should note that, in this case, to calculate a confidence interval it is enough to know the average and standard deviation of the dataset. In our tests the both the average and the standard deviation were very close to the uniform distribution ones.

To estimate the polygon area we just have to count the number of each cell type in the polygon 4CRS signature, and multiply them by the average area stated above. The algorithm in C-like language is presented in Figure 4.

```
void computeApproximateArea(signature4CRS)
  nWeakCells = nStrongCells = nFullCells = 0;
  cellArea = signature4CRS.edgeSize * signature4CRS.edgeSize;
  For each cell in signature4CRS.cells Do
    If (cell.type == Weak) Then
      nWeakCells++;
    Else If (cell.type == Strong) Then
      nStrongCells++;
    Else If (cell.type == Full) Then
      nFullCells++;
  return (nWeakCells * weakWeight + nStrongCells * strongWeight +
         nFullCells * fullWeight) * cellArea;
```

Figure 4. - Algorithm for computing polygon approximate area.

A precision measure is also defined in order to estimate the minimum and maximum areas. The formula is similar to the polygon approximate area formula. The difference is only on the weights used to estimate the polygon area intersecting each cell. While for computing the minimum area the weights are the intersection minimum percentage between the each cell types and the object, the maximum area formula uses the intersection maximum percentage.

The weights for estimating the minimum area are: 0 for empty cells and weak cells; 0.50 for strong cells; and, 1 for full cells.

The weights for estimating the maximum area are: 0 for empty cells; 0.50 for weak cells; and, 1 for strong and full cells.

3.5 Algorithm for approximate area of polygon x window intersection (window query)

The algorithm for computing the approximate area of polygon x window intersection is similar to the algorithm for polygon approximate area (Figure 4). The reason is that we can consider the window as a big full cell, and the intersection of a full cell with any cell type is equal to the area corresponding to cell type. The main difference is that the polygon may be whole contained in the window or may be partially contained in the window. In the former, we only have to execute the polygon approximate area algorithm. In the

latter, we must take care of the cells that are crossed by the window boundaries. In this case, when counting the number of these cells we must consider only the part of the cell that is contained inside the window and not their whole area, i.e., for this sort of cell we must consider the value corresponding to the cell intersection area with the window divided by the cell area, which is less than one. After counting the number of cells of each type, we only have to apply the approximate area formula. The weights used in the formula are the same as the polygon approximate area algorithm. The algorithm in C-like language is presented in Figure 5. For computing the precision of the approximate area (minimum and maximum areas), we use this same algorithm replacing the weights by the same weights used for the polygon approximate area precision calculus.

```

void computeApproximateIntersectionArea(signature4CRS, window)
nWeakCells = nStrongCells = nFullCells = 0;
cellArea = signature4CRS.edgeSize * signature4CRS.edgeSize;
For each signature4CRS cell that is inside the window Do
    If (cell.type == Weak) Then nWeakCells++;
    Else If (cell.type == Strong) Then nStrongCells++;
    Else If (cell.type == Full) Then nFullCells++;
For each cell of signature4CRS1.cells that
    is crossed by the window Do
    intersectionArea = computeIntersectionArea(cell, window)
If (cell.type == Weak) Then
    nWeakCells += intersectionArea / cellArea;
Else If (cell.type == Strong) Then
    nStrongCells += intersectionArea / cellArea;
Else If (cell.type == Full) Then
    nFullCells += intersectionArea / cellArea;
return (nWeakCells * weakWeight + nStrongCells * strongWeight +
        nFullCells * fullWeight) * cellArea;

```

Figure 5. - Algorithm for computing approximate area of polygon \times window intersection.

4. EXPERIMENTAL RESULTS

In this section, the experimental results will be presented corresponding to the execution of the approximate area algorithms presented in Sub-section 3.3. In order to evaluate the 4CRS efficiency the approximate area results are compared with the results obtained computing the object exact areas. The results demonstrate the 4CRS efficiency.

4.1 Test environment

The tests were executed on a PC Pentium IV 1.8 GHz with 512 MB of RAM. The page size used in the experiments was 2,048 bytes. The main goals were compare the response time, the storage requirements and

accuracy of the approximate processing against the processing of the exact representation of the polygons.

The tests were divided into two parts: polygon approximate area, and polygon x window intersection approximate area. In the first test, the approximate area and the exact area were computed over all of the 4CRS signatures and the polygons, respectively. The purpose of this test is just to show a measure of approximations quality – of course a better answer can be obtained by keeping an area attribute for each polygon. In the second tests, in order to take account only the objects that at least have intersection and not all of them, the R*-tree (BECKMANN *et al.*, 1990) was chosen as a spatial access method which is meant to reduce the search space. This choice is due to the wide use of this structure, as well as, to the successful results found in the literature. The access methods traditionally used make use of the object's Minimum Bounding Rectangle (MBR) and this step returns what is called a set of candidates, since it contains all the pairs of polygons that belong to the answer plus other pairs that have only MBR intersection. Therefore, the approximate and exact answers were computed over the resulting data after the access method execution. The approximate query processing was done using the algorithms presented in the Sub-section 3.4 and Sub-section 3.5, while the exact query processing was performed using the General Polygon Clipping library that is available on the web at <http://www.cs.man.ac.uk/aig/staff/alan/software/#gpc>.

4.2 Experimental datasets, approximations and R*-trees characteristics

The polygon real data sets used in the experiments consist of township boundaries, census block-group, topography, geologic map and hydrographic map from Iowa (USA), available on-line in “<http://www.igsb.uiowa.edu/nrgis/gishome.htm>”, and Brazilian municipalities (IBGE, 1996). Some data characteristics are presented in Table 2. Some original datasets were replicated in order to have more representative data. We randomly generated 500 windows for computing approximate area of polygon x window intersection for each original dataset. We omitted the characteristics of all windows data because of their simplicity.

Table 2. Test datasets.

Dataset	Size (Kb)	# pol.	# segments	Aver. # segm.	4CRS size (KB)	4CRS/dataset size -%
Census block-group	29,105	17,844	1,764,588	98	1,006	3.46
Topography	123,367	40,140	7,561,104	188	2,487	2.02
Hydro. map	7,753	2,670	475,812	178	153	1.97
Township boundaries	17,508	12,216	1,059,438	86	722	4.12
Geologic map	10,703	9,984	640,428	64	583	5.45
Municipalities	6,382	4,645	399,002	85	282	4.42
Average						3.57

In order to generate the 4CRS signatures, we have to choose the maximum number of cells (Sub-section 3.1). We performed tests using 500, 1000, 1500 for a maximum number of cells in the grid. The best performance (regarding the trade off between precision and approximation size) was achieved with 500 cells. Due to space limitations we show only the 500 cells grid results. The 4CRS overall sizes are presented in the last column of Table 2. It is important to note that the 4CRS compacting rate varies according to the complexity (average number of segments). The approximation is more compact when the object is more complex, as expected for raster approximations. The 4CRS signature generation time was not showed because Zimbrao and Souza (1998) evaluate its efficiency and shows good results.

In order to evaluate the 4CRS efficiency the approximate area results are compared with the results obtained computing the object exact areas. Therefore, two kinds of R*-tree must be generated: one R*-tree storing the 4CRS signatures and another one without storing them. The former is used in the approximate query processing, and in spite of indexing the real objects they will not be accessed. Table 3 shows the R*-tree characteristics. The column 'R*-Tree type' indicates if the characteristics are for R*-Tree that stores 4CRS signature or R*-tree that not stores them.

The algorithm for approximate area executes until the level of the leaf nodes where the 4CRS signatures are stored. On the other hand, in order to compute the exact area, it is not necessary to access the 4CRS signatures. Then, the R*-tree is generated without storing them, consequently the R*-tree size is smaller, but the spatial objects must be accessed to compute the exact area.

Table 3. – R*-tree characteristics.

Base	R*-Tree type	size (Kb)	Time (sec)	Node average use (%)	Tree Height	# leafs
Iowa Census block-group	4CRS	1,604	23.664	67.73	3	896
	-	822	23.394	72.40	3	403
Iowa Topography	4CRS	8,712	168.032	31.55	3	4,270
	-	2,388	115.586	56.20	3	1,170
Iowa Hydrologic map	4CRS	248	2.674	68.88	3	120
	-	124	2.514	72.77	2	60
Iowa Township boundaries	4CRS	1,168	19.829	68.11	3	573
	-	582	16.674	70.01	3	285
Iowa Geologic map	4CRS	946	12.268	67.96	3	464
	-	480	12.168	69.48	3	235
Brazil Municipalities	4CRS	434	6.720	72.05	3	211
	-	214	5.468	73.74	3	103

4.3 Results of approximate query processing

The experimental results were divided into two parts. The results of the algorithm for polygon approximate area (Sub-section 3.4) are presented in Table 4. The experimental results of the algorithm for polygon x window intersection approximate area (Sub-section 3.5) are presented in Table 5. The results presented in both tables are: the accuracy (approximate area, minimum approximate area, and maximum approximate area percentages); the total execution time (the time needed to compute the approximate answer, the time needed to compute the exact answer; and the proportion between approximate and exact processing, that represents how fast is the approximate area processing related to the exact area processing); finally, the last three columns present the number of disk accesses required to compute the approximate area; the number of disk accesses required to compute the exact area; and the proportion between them. The total execution time is not a good measure of performance gain as it is totally dependent on the algorithm used. Instead, the total number of disk accesses is a reliable performance gain measure, as the objects to be processed have to be, at least, read from disk.

In the tests of the algorithm for polygon x window intersection approximate area, we assume that an area attribute is present in each MBR polygon, so if the polygon is completely inside the window query its area attribute is used in both algorithms.

Table 4. – Experimental results of the algorithm for polygon approximate area

Dataset	Approximate area accuracy (%)			Execution time (secs)			# Disk Access		
	Avg.	Min.	Max.	Appr	Exact	Appr/Exact (%)	Appr	Exact	Appr / Exact (%)
Census block-group	-3.92	-11.46	3.62	1.69	7.82	21.65	259	14,552	1.78
Topograp.	-0.13	-26.70	27.05	3.26	25.51	12.80	694	61,683	1.13
Hydrologic	-3.75	-12.34	4.85	0.32	2.03	15.79	39	3876	1.01
Township boundaries	-4.00	-8.26	0.26	1.55	3.13	49.52	193	8753	2.20
Geologic map	-2.87	-17.33	11.60	0.95	2.52	37.72	154	5351	2.88
Municip.	-1.07	-16.79	14.65	0.39	1.18	32.99	77	3190	2.41
Average	-2.62					28.41			1.90

Table 5. – Experimental results of the algorithm for polygon × window area

Dataset	Approximate area accuracy (%)			Execution time (secs)			# Disk Access		
	Avg	Min.	Max.	Appr	Exact	Appr/Exact (%)	Appr	Exact	Appr / Exact (%)
Census block-group	0.47	-3.75	3.46	8.95	103.48	8.65	4,873	63,182	7.71
Topograp.	0.50	-7.96	11.46	7.58	348.02	2.18	7,576	141,583	5.35
Hydrologic	0.39	-2.17	4.89	0.94	25.807	3.65	885	17,168	5.15
Township boundaries	0.27	-1.29	4.82	5.29	53.768	9.83	3,250	32,251	10.08
Geologic map	1.45	-6.86	14.62	4.23	54.769	7.72	2,811	30,727	9.15
Municip.	0.96	-0.19	1.99	2.78	24.665	11.29	556	11,629	4.78
Average	1.00					7.22			7.04

The results were good. The approximate query processing has a quite small error, a short execution time and a small number of disk accesses comparing with the exact query processing. In the case of polygon approximate area the average error for each dataset varies from -4.00% to -0.13% (average of -2.62%). The execution time is between 12.80% and 49.52% (average of 28.41%) of the time need to process the exact answer, and the number of disk accesses is between 1.01% and 2.88% (average of 1.90%). As stated before, this query is to illustrate the method accuracy on approximating one polygon area alone.

The results were better for the polygon x window intersection approximate area algorithm: approximate area error average for each dataset between 0.27% and 1.45% (average of 1%); execution time between 2.18%

and 11.29% (average of 7.22%); and, number of disk accesses between 4.78% and 10.08% (average of 7.04%).

One can note that there were some bad results: on Geologic Map data set the max error for a query was 14.62%. In fact, queries where few polygons (and consequently few cells) are involved produces not so good results. This can be explained by the Central Limit Theorem: the distribution of the sum of cell area will tend to be Normal as the number of cells increase, since the polygon area inside each cell is supposed to be uniformly distributed. With few cells, the distribution does not tend to be Normal. Nonetheless, queries with few polygons can be executed accessing the polygons instead of the approximations – we will spend just a little more time.

Finally, we now show that it is easy to compute a confidence interval for the result of a query, so the user can decide if the precision is enough. Assuming the uniform distribution, the variance of % area of weak cells is $(0.5-0)^2/12 = 1/48$. The strong cell has the same variance. Using the Central Limit Theorem, the sum of a great number of cells will tend to be Normal, and its variance will be $N/48$, where N is the number of cells. So, consulting any statistical table, for a 95% confidence interval we have a range of $N \times (0.25 \pm 1.96 \times (N/48)^{1/2})$, and for a 99% confidence interval we have $N \times (0.25 \pm 2.576 \times (N/48)^{1/2})$. To get the numbers in area units we have to multiply these limits by the cell area. Also, we have to consider the strong and full cells. For example, if a window query produces 100 weak cells, 120 strong cells and 400 full cells we compute the 95% confidence interval as follows (for simplicity we assume that each cell has the same area, equals to 1):

- Weak cells: $100 \times (0.25 \pm 1.96 \times (100/48)^{1/2}) = 25 \pm 2.83$
- Strong cells: $120 \times (0.75 \pm 1.96 \times (120/48)^{1/2}) = 90 \pm 3.10$
- Full cells: 400 (full cells have the exact area!)
- Total: 515 ± 5.93

So, the confidence interval has a range of $\pm 1.15\%$, that is, 95% of the approximate answers with these number of cells will have an error of at most $\pm 1.15\%$, a result with enough precision for most applications. Conversely, lets look at a query involving few cells, say 10 weak, 12 strong and 10 full:

- Weak cells: $10 \times (0.25 \pm 1.96 \times (10/48)^{1/2}) = 2.5 \pm 0.89$
- Strong cells: $12 \times (0.75 \pm 1.96 \times (12/48)^{1/2}) = 9 \pm 0.98$
- Full cells: 10 (full cells have the exact area!)
- Total: 21.5 ± 1.87 , that is, $\pm 8.9\%$.

5. CONCLUSION

The experimental results demonstrated the efficiency of the 4CRS use for approximate query processing. In our tests, the approximate answers have a quite small error (average of -2.62% and 1% for approximate area and window \times polygon approximate area), the execution time is much shorter than the time required to process the exact answers (average of 28.41% and 7.22% related to the exact processing, respectively), and the number of disk accesses were also quite small (1.90% and 7.04% of the exact query processing disk accesses, respectively) (Table 4 and Table 5). Moreover, the space required to store 4CRS approximations are much smaller than the space needed to store the real datasets (Table 2), which is in average 3.57%.

We can credit to the Central Limit Theorem the good results obtained in our approach. In fact, this Theorem ensures that good results would be obtained even if the polygon area inside each cell was not uniformly distributed. Also, more precise answers will be obtained as the number of cells involved in a query increase, that is, larger window queries involving a great number of polygons. This is a great result since the exact answers of these queries are the more time consuming. On the other hand, our approach will obtain less precise answers when few polygons are in the query – in these cases, if the estimated precision was not enough, the exact query could be performed in acceptable time.

ACKNOWLEDGEMENTS

Azevedo and Monteiro are supported by CNPq. This work was partially developed at FernUniversität in Hagen and University of Stuttgart (Germany) where Azevedo and Monteiro are currently visitor students, respectively.

REFERENCES

- Agarwal, S., R., Deshpande, P. M., Gupta, A., Naughton, J. F., Ramakrishnan, R., Sarawagi S., 1999, *On the computation of multidimensional aggregates*, *SIGMOD* (1999), 193-204.
- Aoki, P. M., 1998, *Generalizing “search” in generalized search trees*, *IEEE Conf. Data Engineering* (1998), 380-389.
- Aronoff, S., 1989, *Geographic Information Systems*, WDL Publications, 1989.
- Azevedo, L. G., Monteiro, R. S., Zimbrao, G., Souza, J. M., 2003, *Polyline Spatial Join Evaluation Using Raster Approximation*, *GeoInformatica, Kluwer Academic Publishers* (2003), 7 (4): 315-336.
- Beckmann, N., Kriegel, H. P., Schneider, R., et al., 1999, *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles*, *ACM SIGMOD* (1999), pp. 322-331.

- Brinkhoff, T., Kriegel, H.-P., Seeger, B., *Comparison of approximations of complex objects used for approximation-based query processing in spatial database systems*, *Conf. on Data Engineering* (1993), 40-49.
- Cannataro, M., Guzzo, A., Pugliese, A., 2002, *Knowledge Management and XML: Derivation of Synthetic Views over Semistructured Data*. *ACM SIGAPP* 10 (2002), 33-36.
- Chackrabarti, K., Garofalakis, M., Rastogi, R., Shim, K., 2000, *Approximate Query Processing Using Wavelets*, *VLDB '2000*.
- Dash, M., Liu, H., Yao, 1997, J., *Dimensionality reduction of unsupervised data*, *IEEE International Conference on Tools with AI 9* (1997), 532-539.
- Faloutsos, C., Barbara, D., DuMouchel, W., Haas, Hellerstein, P. J., J. M., Ioannidis, Y. E., Jagadish, H. V., Johnson, T., Ng, R. T., Poosala, V., Ross, K. A. and Sevcik, K. C., 1997, *The New Jersey Data Reduction Report*, *IEEE Data Engineering Bulletin* (1997), 20(4):3-45.
- Furtado, P., Madeira H., 1999, *Summary Grids: Building Accurate Multidimensional Histograms*, *DASFAA 6* (1999).
- Furtado, P., Madeira, H., 2000a, *Data Cube Compression with Quanticubes*, *Data Warehousing and Knowledge Discovery* (2000), 162-167.
- Furtado, P., Madeira, H., 2000b, *FCompress: A New Technique for Queriable Compression of Fact and Datacubes*, *IDEAS* (2000).
- Gibbons, P. B., Matias, Y., Poosala, V., 1997, *Aqua project white paper*, *Technical report*, *Bell Laboratories* (1997).
- Gütting, H., 1994, *An Introduction to Spatial Database Systems*. *VLDB Journal* 3 (1994), 357-399.
- Han, J., Kamber, M., 2001, *Data Mining: concepts and techniques*, Academic Press.
- IBGE (Brazilian Institute of Geography and Statistics), 1996, *Malha Municipal Digital do Brasil - 1994*, IBGE .
- Ioannidis, Y. E., Poosala, V., 1995, *Balancing histogram optimality and practicality for query result size estimation*. *ACM SIGMOD* (1995), 233-244.
- Johnson, G. H. and Wichern D. A., 1992, *Applied Multivariate Statistical Analysis*, Prentice Hall.
- Kohavi, R., John, G. H., 1997, *Wrappers for feature subset selection*, *Artificial Intelligence* (1997), 273-324.
- Matias, Y., Vitter, J. S., Wang, M., 1998, *Wavelet-based histograms for selectivity estimation*, *ACM SIGMOD* (1998).
- Monteiro, R. S., Azevedo, L.G., Zimbrão, G., Souza, J. M., 2004, *Polygon and Polyline Join Using Raster Filters*, *DASFAA* (2004), 255-261.
- Papadias, D., Kalnis, P., Zhang, J., Tao, Y., 2001, "Efficient OLAP Operations in Spatial Data Warehouses", *SSTD*.
- Papadias, D., Mamoulis, N., Theodoridis, Y., 1999, *Processing and optimization of multiway spatial joins using R-Trees*, *ACM PODS* , 189-200.
- Papadimitriou, C. H., Suciu, D. and Vianu, V., 1996: *Topological queries in spatial databases*, *PODS* , 81-92.
- Poosala, V., Ioannidis, Y. E., Haas, P. J., Shekita, E. J., 1996, *Improved Histograms for Selectivity Estimation of Range Predicates*, *ACM SIGMOD* (1996), 294-305.
- Roddick, J., Egenhofer, M., Hoelpapadias, E. D., Salzberg, B., 2004, *Spatial, Temporal and Spatiotemporal Databases - Hot Issues and Directions for PhD Research*, *SIGMOD Record* (2004), 33(2).
- Ross, K., Srivastava, D., 1997, *Fast computation of sparse datacubes*, *VLDB '1997*, 116-125.
- Samet H., 1990, *The Design and Analysis of Spatial Data Structure*, Addison-Wesley Publishing Company.

- Sarawagi, S., Stonebraker, M., 1994, *Efficient organization of large multidimensional arrays*, *ICDE* (1994), 328-336.
- Tao, Y., Sun, J., Papadias, D., 2003, *Selectivity estimation for predictive spatio-temporal queries*, *ICDE* (2003).
- Zimbrão, G. and Souza, J. M., 1998, *A Raster Approximation For Processing of Spatial Joins*, *VLDB'1998*.