Approximate Query Processing in Spatial Databases Using Raster Signatures

Leonardo Guerreiro Azevedo[1], Geraldo Zimbrão[1,2], Jano Moreira de Souza[1,2]

[1]Computer Science Department, Graduate School of Engineering, Federal University of Rio de Janeiro, PO Box 68511, ZIP code: 21945-970, Rio de Janeiro, Brazil

[2]Computer Science Department, Institute of Mathematics, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

{azevedo, zimbrao, jano}@cos.ufrj.br

Abstract. *Traditional query processing provides exact answers to queries. However, in many applications, the response time of exact answers is often longer than what is acceptable. Approximate query processing has emerged as an alternative approach to give to the user an answer in a short time. The goal is to provide an estimated result in one order of magnitude less time than the time to compute the exact answer. There is a large set of techniques for approximate query processing; however, most of them are only suitable for traditional data. This work proposes new algorithms for a set of spatial operations that can be processed approximately using 4CRS (Four-Color Raster Signature).*

Resumo. *Processamento tradicional de consultas visa prover respostas exatas para consultas; todavia, em muitas aplicações, o tempo de uma resposta exata é frequentemente muito maior do que o desejado. Processamento aproximado de consultas tem surgido como uma abordagem alternativa para processar consultas em um curto período de tempo, retornando uma resposta estimada para o usuário. Existem várias técnicas para processamento aproximado de consultas; todavia, muitas delas são aplicáveis apenas a dados tradicionais. Este trabalho propõe novos algoritmos para operações espaciais que podem ser processadas de forma aproximada usando a assinatura 4CRS (Assinatura Raster de Quatro Cores).*

1. Introduction

A main issue in database area is to process queries efficiently so that the user does not have to wait a long time to get an answer. However, there are many cases where it is not easy to accomplish this requirement. In addition, a fast answer could be more important for the user than receiving an accurate exact one. In other words, the precision of the query could be lessened, and an approximate answer could be returned, provided that it is much faster than the exact query processing, and it has an acceptable accuracy.

Approximate query processing has emerged as an alternative for query processing in environments for which providing an exact answer can demand a long time. The goal is to provide an estimated response in orders of magnitude less time than the time to compute an exact answer, by avoiding or minimizing the number of disk accesses to the base data (Gibbons *et al.*, 1997).

There are a large set of techniques for approximate query processing available in different research areas, as presented by Azevedo *et al.* (2004). Good surveys of techniques for approximate query processing are presented by Barbara *et al.* (1997) and Han and Kamber (2001), where several techniques are described and evaluated, w.r.t. data types being reduced and applications where they can be applied. However, most of the techniques are only suitable for relational databases. On the other hand, providing a short time answer to queries becomes a bigger challenge in spatial database area, where the data usually have high complexity and are available in huge amounts. Furthermore, this subject is a hot research issue in spatial-temporal databases as pointed by Roddick *et al.* (2004). Besides the complexity and huge amount of data, Roddick *et al.* (2004) emphasize that several spatial applications focus on retrieving 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"), and not focus on exact information about the qualifying objects (i.e., the car ids), which may be unavailable, or irrelevant.

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 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, such as: 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.

In a traditional SDBMS query processing environment (Figure 1), user queries are sent to the database that processes them and returns to the user an exact answer. In database updates, new data can be inserted or existing data can be updated or deleted from the database.

On the other hand, in a SDBMS set-up for providing approximate query answers, a new component is added, the approximate processing engine (Figure 2). In this new framework, queries are sent directly to the approximate processing engine. It processes the query and returns an approximate answer to the user, along with a confidence interval showing the response accuracy. If the precision is not sufficient for the user to take his decision, the query can be processed by the SDBMS, providing an exact answer to the user.

The approximate processing engine stores reduced representation of real data to perform the approximate processing. Therefore, it is also possible, to execute the query partially over approximate data and partially over real data, when it is not guaranteed that some approximations (or synopses) of objects do not produces the desired accuracy or if the real representation of objects are so simple that it has the same processing costs of computing the exact or the approximate answer. For instance, to compute the area of a polygon of few vertices could be executed in almost the same time as the time to process the same query over a synopsis of the real data.

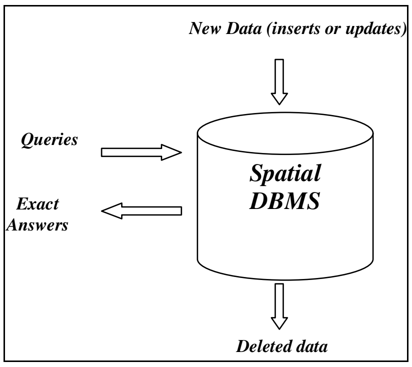

Figure 1. Tradition SDBMS query processing environment.

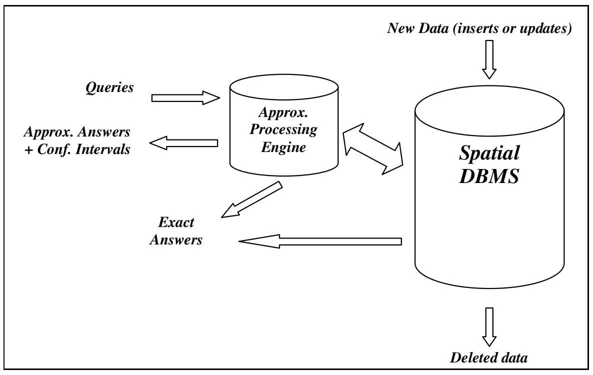
New Data (inserts or updates)
Queries
Approx. Answers + Conf. Intervals
Approx. Processing Engine
Spatial DBMS
Exact Answers
Deleted data

Figure 2. SDBMS set-up for providing approximate query answers.

One important issue regards to the approximate processing engine is related to the maintenances of data. When new data arrives or stored data is updated, it is also required to store the representations of new objects in the engine or update the existing synopses. Therefore, it is also important that synopses can be computed quickly. In the case of deletion of objects, the existing synopses must be deleted as well. Thus all information that comes or leaves the SDBMS must be sent to the approximate engine in order to keep it up-to-date.

This work is concerned with the use of raster signatures in approximate query processing in spatial databases. We extended the proposals of Azevedo *et al.* (2004, 2005) of using Four-Color Raster Signature (4CRS) (Zimbrao and Souza, 1998) for fast and approximate processing of queries over polygon datasets. We propose several new algorithms for a set of spatial operations that can be processed approximately using 4CRS. 4CRS stores the main data characteristics in an approximate and compact representation that can be accessed and processed faster than real data. By doing so, the exact geometries of objects are not processed during query execution, which is the most costly step of a spatial operation, since it requires to search and to transfer large objects from disk to main storage (Brinkhoff *et al.*, 1994; Lo and Ravishankar, 1996). Also, the exact processing needs to use CPU-time intensive algorithms to decide whether the objects match the query condition (Brinkhoff *et al.*, 1993). As a result, the approximate query answer is returned in a shorter time than the time to return an exact answer. On the other hand, the answer is estimated and not exact, so a precision measure is also returned as a confidence interval that allows the user to decide if the accuracy of the response is sufficient. In general, this approximate answer is enough for many applications.

It is important to emphasize that the main target of this work is to present our proposals of algorithms, while the experimental evaluation of them is a current work being developed on Secondo (Güting *et al.*, 2005), which is an extensible DBMS platform for research prototyping and teaching. However, the experimental results of evaluating approximate processing against exact processing presented by Azevedo *et al.* (2004, 2005) demonstrated the efficiency of using 4CRS for approximate query processing. Azevedo *et al.* (2004) evaluated an algorithm for computing polygon approximate area and an algorithm for computing the approximate area of polygon × window intersection (window query). According to those evaluations, the approximate processing is 3.5 times and 14 times faster than the exact processing in response time, while 52 times and 14 times faster relate to number of disk accesses, respectively. The response error is also quite small, an average of -2.62% and 1%, respectively. Azevedo *et al.* (2005) evaluated an algorithm for estimating the overlapping area of polygon join. In this case, the approximate processing varies from 5 to 15 times faster than the exact processing in response time and from 5 to 10 times faster relate to number of disk accesses. Approximate answers have an average error of 0.6%.

This work is divided in sections, as follows: Section 2 presents scenarios and applications where approximate query processing can be applied; Section 3 presents our proposals of algorithms for approximate processing of several spatial operations; and, in Section 4, we present the conclusions and future work.

2. Scenarios and Applications

There are many scenarios and applications where a slow exact answer can be replaced by a fast approximate one, provided that it has the desired accuracy. Hellerstein *et al.* (1997) emphasize that in Decision Support Systems the increasing in business competitiveness is requiring an information-based industry to make more use of its accumulated data, and thus techniques of presenting useful data to decision makers in a timely manner are becoming crucial. They propose also the use of approximate query processing during a drill-down query sequence in ad-hoc data mining, where the earlier queries in the sequence are used solely to determine what the interesting queries are. Papadias *et al.* (2001) proposes the approximate query processing for spatial OLAP.

Gibbons *et al.* (1997) highlight that an approximate answer can provide feedback on how well-posed a query is. Besides, it can be used when the query requests numerical answers, and the full precision of the exact answer is not required, 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 millions, or the nearest percentile of a percentage).

Ioannidis and Poosala (1995) and Gibbons *et al.* (1997) propose the use of approximate query processing to define most efficient execution plan for a given query. Das *et al.* (2004) propose its use in selectivity estimation in Spatial Database Management Systems (SDBMS) in order to return approximate results that come with provable probabilistic quality guarantees.

An approximate answer can also be used as an alternative answer when the data is unavailable in data warehousing environments and in distributed data recording as pointed by Gibbons *et al.* (1997), Faloutsos *et al.* (1997) and Jagadish *et al.* (1995) or in mobile computing as highlighted by Madria *et al.* (1998). In mobile computing, it can be advantageous to relax completeness or precision criteria so that an approximate answer can be sent to the user in cases of slow network link, temporary non-availability, or even low resources of internet connection or available memory for query processing (Madria *et al.*, 1998).

Dobra *et al.* (2002) indicates to use approximate query processing in order to make decisions and infer interesting patterns on-line, such as over continuous data streams. In stream environment usually only limited memory resources are available to query processing related to the volume of data. Thus, we need algorithms that can summarize the data streams involved in a concise and reasonably accurate synopsis (or approximations of the real data) that can be stored in small amount of memory and can be used to provide approximate answers to user's queries along with some reasonable guarantees of answer's accuracy. For example, in applications for trend analysis and fraud/anomaly detection in telecom-network data, where the goal is to identify generic, interesting or "out-of-the-ordinary" patterns rather than provide results that are exact to the last decimal.

3. Approximate query processing using Four-Color Raster Signature

The 4CRS signature was first used to improve the processing of spatial joins of polygon datasets. It was proposed by Zimbrao and Souza (1998) as a filter in the second step of the Multi-Step Query Processor (MSQP) (Brinkhoff *et al.*, 1994), in order to reduce exact geometry test of spatial objects. 4CRS is a polygon raster signature represented by a small bit-map of four colors upon a grid of cells. Each grid cell has a color representing the percentage of the polygon's area within cell: *Empty* (0% of intersection); *Weak* (the cell contains an intersection of 50% or less with the polygon); *Strong* (the cell contains an intersection of more than 50% with the polygon and less than 100%); and, *Full* (the cell is fully occupied by the polygon). The grid can have its scale changed in order to obtain a more accurate representation (higher resolution) or a more compact one (lower resolution). Further details of 4CRS signature can be found in Zimbrao and Souza (1998) and Azevedo *et al.* (2004).

The 4CRS characteristics and the good results obtained using 4CRS as geometric filter in polygon join processing motivated its use on approximate query processing. This new approach has the goal not solely to reduce the number of objects that have their exact geometry processed. Instead, we propose to return to the user an approximate answer that is obtained processing the query over the 4CRS signatures of polygons, without accessing the object's real representation, and not executing the exact geometry step (the most expensive one). Hence, new algorithms must be designed, implemented and evaluated to concern to the requirements of this new of approach.

3.1 Approximate Operations

There are many operations that could benefit from a fast and approximate query processing, so that the user could have an answer in a short time instead of waiting a long time for an exact answer. In this work we present our proposals of approximate query processing algorithms based on the classification proposed by Güting *et al.* (1995) and Güting and Schneider (1995) in the Rose Algebra. They divide the spatial operations into four groups.

- **Spatial operators returning numbers**: area, number of components, distance, diameter, length, perimeter;
- **Spatial predicates**: equal, different, disjoint, inside, area disjoint, edge disjoint, edge inside, vertex inside, intersects, meet, adjacent, border in common;
- **Operators returning spatial data types values**: intersection, plus, minus, common border, vertices, contour, interior;
- **Spatial operators on set of objects**: sum, closest, decompose, overlay, fusion.

In this work, we will show directions for designing and implementing approximate processing algorithms for those operations. In this section we enumerated these operations, and, in the next section (Section 3.2), we present our proposals of algorithms.

3.2 Approximate Operations using 4CRS signature

This section presents directions for researching of approximate processing algorithms for the operations presented in Section 3.1. In order to make the descriptions simpler, we present similar operations in the same section.

3.2.1 Approximate Area of Polygon

The algorithms that return the approximate area of polygon and the approximate area of polygon within cell are proposed by Azevedo *et al.* (2004), while the algorithm that returns the approximate area of polygon join are proposed by Azevedo *et al.* (2005), those works that this work extends. The first two algorithms are based on the expected area of polygon within cell and the last is based on the expected area of intersection of two types of cells. These definitions are presented in details in those works. Therefore, we will present here only short explanations of those definitions, which are used to describe our new proposals of algorithms.

The expected area of polygon within cell corresponds to a sum of estimatives of the area of polygon within cell types. For example, an expected area of polygon within an *Empty* cell is equal to 0% (zero percent), since there is no portion of the polygon inside the cell. For *Weak*, *Strong* and *Full* cells the estimatives are 25%, 75% and 100%, respectively.

The expected areas of intersection of two types of cells are used to estimate the intersection of two polygons, which is approximately answered as the intersection of their 4CRS signatures. For example, the expected area corresponding to a combination of an *Empty* cell with any other type of cell results in an expected area of 0% (zero percent). Similarly, when two *Full* cells overlap, the expected area is 100%. More details about expected area are presented in Azevedo *et al.* (2004, 2005).

3.2.2 Distance

The distance between two polygons can be estimated from their 4CRS signatures, computing the distance among cells corresponding to polygons' borders (*Weak* and *Strong* cells). The result can be estimated as the average of the minimum and maximum distances computed as follow. The minimum distance is the distance between the outer borders of cells (borders adjacent to borders of *Empty* cells), while the maximum distance is computed from the inner borders of these cells (i.e., borders opposite to the outer borders). The minimum and maximum distances can be used to define a confidence interval for the computed approximate distance. Figure 3 presents an example of computing the distance between two polygons using their 4CRS. The polygons are presented in Figure 3.a, while Figure 3.b shows their 4CRS signatures. Figure 3.c is a zoom of a cell combination used to compute the approximate distance, highlighting the minimum and maximum distances between two cells.

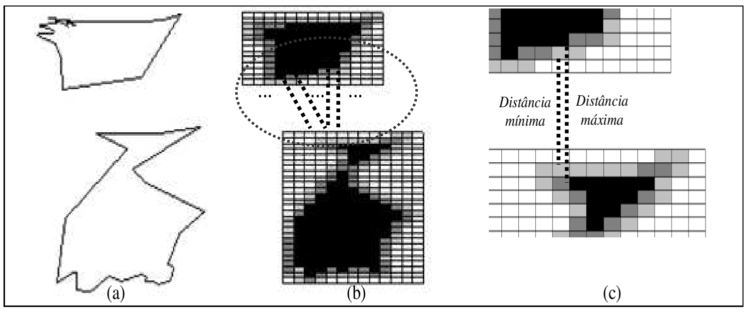

Figure 3. Example of computing the minimum and maximum distance between two polygons from their 4CRS signatures.

3.2.3 Diameter

The diameter of a spatial object is defined as the longest distance between any of its components. Therefore, in the case of polygons, the diameter is the longest distance between the faces that compose the polygon. So, the diameter can be computed using the same algorithm to compute distance between polygons, since each face has a different 4CRS signature, as presented in Figure 4.

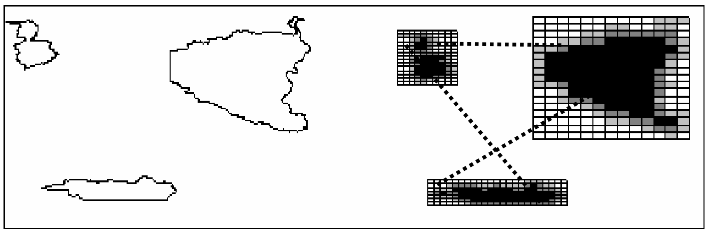

Figure 4. Example of computing the diameter of a polygon made by three faces from its 4CRS signature.

3.2.4 Perimeter and Contour

The perimeter operation calculates the sum of the length of all cycles of a region (or polygon) value. If we intend to compute only the sum of the length of the outer cycles and not consider the holes of a polygon, the contour operator can be used to eliminate holes first. The perimeter of a polygon can be computed from its 4CRS signature. A simple proposal is to compute the perimeter as the average of the outer perimeter and the inner perimeter. Figure 5 presents an example of outer and inner perimeters of a polygon. Figure 5.a and Figure 5.b show the polygon and its 4CRS signature, respectively. Figure 5.c presents the outer perimeter of the 4CRS signature, and Figure 5.d presents the inner perimeter. The outer perimeter can be computed as the sum of the length of edges corresponding to the borders of cells of type different from *Empty*, those are adjacent to *Empty* cells (for example, cell c' presented in Figure 5.b, whose *Top* and *Left* edges would be considered as part of the outer perimeter) or adjacent to the border of the signature's MBR (for example, cell c'' presented in Figure 5.b, whose *Top* and *Left* edges would also be considered as part of the outer perimeter). MBR (Minimum Bounding Rectangle) is the smallest rectangle that encloses a spatial object. It is the most popular geometric key. MBR reduces a spatial object's complexity to four parameters that retain its most important characteristics: position and extension. The signature's MBR is the smallest rectangle that encloses the object's signature, and its coordinates are in power of two (Zimbrao and Souza, 1998). On the other hand, the inner perimeter could be computed as the sum of length of edges of cells adjacent to the cells that were considered when computing the outer perimeter.

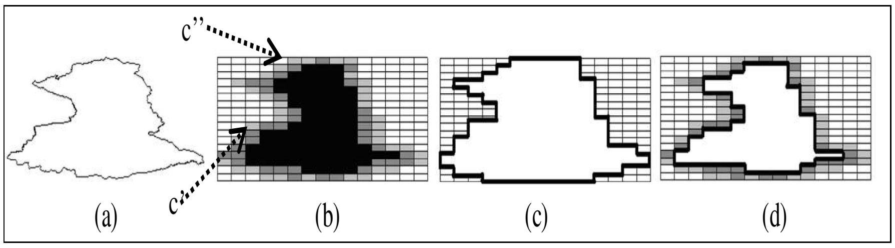

Figure 5. Example of outer and inner perimeter of polygon from its 4CRS signature.

The calculus of the contour of a polygon can be approximately computed from its 4CRS signature similarly to the calculus of the perimeter of a polygon. We can connect the medium points of *Weak* and *Strong* cells in order to compute the contour of the polygon, and we can assume that the maximum contour corresponds to the cells that compose the outer perimeter (Figure 5.c) and the minimum contour corresponds to cells that compose the inner perimeter (Figure 5.d).

3.2.5 Equal and Different

In exact processing, the equal and different operations returns exactly if two objects are equal or not, respectively. On the other hand, in approximate processing using 4CRS is not always possible to state that objects are equal or different with 100% of confidence. In this case, the equality and inequality is defined as a function that returns a value in the interval [0,1] that indicates a true percentage of the equality (or inequality) of objects. We call this value as "affinity degree". By doing so, we can test if two objects are equal (or different) using their 4CRS signatures. For each comparison of pair of cells we compute an affinity degree. The final affinity degree is equal to the sum of the individual degrees divided by the number of comparisons of pair of objects, if no trivial case occurs. So, for instance, when comparing a pair of *Empty* cells, we can state that the polygons are 100% equal, since *Empty* cells do not have any intersection with polygon. Similarly, two polygons are equal when comparing *Full* cells. On the other hand, when comparing *Weak* and *Strong* cells a different reasoning must be used. Our proposal is to use for these cases the concept of expected area employed by the algorithm that computes the approximate area of polygon × polygon intersection (Azevedo *et al.*, 2005). In other words, for exact cases (comparisons of *Empty* ×*Empty* and *Full* × *Full* cells) the equality (or affinity degree) is 1, while for other cases the affinity degree is equal to the expected area. For instance, the overlap of *Weak* × *Weak* cells contributes with 0.0625, while the overlap of *Strong* × *Strong* cells contributes with 0.5625, which represent the intersection of polygon intersection using their 4CRS signatures' cells. The answer is computed as the sum of the affinity degrees divided by the number of comparisons. It is important to highlight that if exists at least one overlap of different cell types we can state that the objects are not equal.

The algorithm that returns if two objects are equal is presented in Figure 6. The algorithm returns 0 if the polygons are not equal, otherwise it returns an affinity degree that shows a measure of equality of the objects.

```
1. real equal(signat4CRS1, signat4CRS2)
2.   if signat4CRS1.lengthOfCellSide ≠ signat4CRS2.lengthOfCellSide then
3.      return 0;
4.   if signat4CRS1.nCells ≠ signat4CRS2.nCells then
5.      return 0;
6.   if signat4CRS1.mbr ≠ signat4CRS2.mbr then
7.      return 0;
8.   affinityDegree = 0;
9.   nRuns=0;
10.  for each c1 cell of signat4CRS1 do
11.    for each c2 cell of signat4CRS2 that overlaps c1 do
12.       if c1.type==c2.type then
13.          if c1.type==Empty or c1.type==Full then
14.             affinityDegree += 1;
15.          else if c1.type==Weak then
16.             affinityDegree += 0.0625;
17.          else
18.             affinityDegree += 0.5625;
19.       else
20.         return 0;
21.       nRuns++;
22.  return affinityDegree / nRuns;
```

Figure 6. Algorithm that returns if two polygons are approximately equal.

In the case of the operation different, we can use a similar reasoning, and we propose an algorithm similar to "equal" algorithm (Figure 6). If the objects' signatures have different cell size or different number of cells or different MBRs then they are different. In other words, we must change the lines 3, 5 and 7 of the algorithm to return 1 instead of zero. The other changes must be done in the loop that evaluate the overlap of pair of cells (line 12 to line 20). The overlap of empty or full cells represents an affinity degree equals to zero (line 14 - "affinityDegree += 0"). The overlap of two *Weak* cells represents that the polygons has affinity degree equal to "1 – 0.0625" to be different (line 16 – "affinityDegree += 1 – 0.0625"). In the case of *Strong* cells overlap, the affinity degree is equal to "1 – 0.5625" (line 18 – "affinityDegree += 1 – 0.5625"). If exists an overlap of different cell types then the objects are different (line 20 – "return 1").

3.2.6 Disjoint, Area Disjoint, Edge Disjoint

Two objects are disjoint if they have no portion in common. In the case of area disjoint, the objects do not have area in common, but they can have overlap of their edges. On the other hand, two objects are edge disjoint if they do not have overlap of edges. 4CRS signatures can be used to estimate if two objects are disjoint, area disjoint or edge disjoint. In some cases it is also possible to return an exact answer.

When comparing the 4CRS signatures of two polygons, if there are only overlap of *Empty* cells × any other type of cells, we can state that the polygons are disjoint, and, consequently, they are also area disjoint and edge disjoint. On the other hand, if there is at least one overlap of a *Full* cell × *Weak* or *Strong* or *Full* cell, we can state that the objects are not disjoint nor area disjoint nor edge disjoint. It is also possible to state that two polygons are edge disjoint if there is no overlap of *Weak* × *Strong* cells, i.e., or one polygon is inside the other, or it is outside the other. Therefore, an approximate answer is returned to the user only when there are intersections of *Weak* × *Strong* cells, otherwise we can return an exact answer.

Our proposal is to use the expected area of overlap of two cells to estimate the answer for these cases, and to return to the user an affinity degree of the answer. Thus, a weight of 100% is assigned to pair of cell comparisons where it is possible to have an exact answer. In the cases that an approximate value is computed, we propose to use the complement value of the expected area of polygon intersection, since we are interested in estimating disjunction of polygons, which is opposite to estimating the intersection area of polygons.

Figure 7 presents an algorithm to compute an affinity degree about disjunction of two polygons. This algorithm can be also used to determine if two objects are area disjoint.

```
1.  real disjoint(signat4CRS1, signat4CRS2)
2.     interMBR = intersectionMBR(signat4CRS1, signat4CRS2);
3.     if interMBR is NULL then /*Does not exist MBR intersection*/
4.        return 1;
5.     if (signat4CRS1.lengthOfCellSide < signat4CRS2.lengthOfCellSide) then
6.        s4CRS = changeScale(signat4CRS1, signat4CRS2.lengthOfCellSide);
7.        b4CRS = signat4CRS2;
8.     else
9.       if (signat4CRS1.lengthOfCellSide > signat4CRS2.lengthOfCellSide) then
10.         b4CRS = signat4CRS1;
11.         s4CRS = changeScale(signat4CRS2,signat4CRS1.lengthOfCellSide);
12.      else /*(signat4CRS1.lengthOfCellSide == signat4CRS2.lengthOfCellSide)*
13.         s4CRS = signat4CRS1;
14.         b4CRS = signat4CRS2;
15.    affinityDegree = 0;
16.    nRuns = 0;
17.    for each b4CRS cell b that is inside interMBR do
18.       for each s4CRS cell s that intersects cell b do
19.          if ( (b.type == Empty) or (s.type == Empty) ) then
20.             affinityDegree += 1;
21.          else
22.            if ( (b.type == Weak) and (s.type == Weak or s.type == Strong) )
23.               ( (b.type == Strong) and (s.type == Weak)) then
24.               affinityDegree += (1 - expectedArea[s.type,b.type]);
25.            else /*Full × Full*/
26.               return 0;
27.          nRuns++;
28.    return affinityDegree / nRuns;
```

Figure 7. Algorithm for returning if two objects are disjoint.

The algorithm to evaluate if two polygons are edge disjoint is very similar to the algorithm proposed in Figure 7. The only difference is in the part that compares *Full* × *Full* cells that correspond to line 22 that must be changed by “affinityDegree += 1”).

3.2.7 Inside (Encloses), Edge Inside, Vertex Inside

From the 4CRS signatures of two polygons is possible to state that a polygon P_1 is inside a polygon P_2 if all cells of 4CRS signature of P_1, different from *Empty*, are overlapped by *Full* cells of the 4CRS signature of P_2. On the other hand, if there is an overlap of at least P_1 cell of type different from *Empty* with an *Empty* cell of P_2, we can state that P_1 is not inside P_2. In the case of overlap of *Weak* × *Strong* cells or *Weak* × *Weak* cells or *Strong* × *Strong* cells, it is not possible to return an exact answer. Hence we need to define an approximate value for these cases of cell overlaps. We propose to use the expected area of polygon intersection for estimating the answer. The algorithm for inside and edge inside operations are the same. However, the vertex inside operation cannot be approximately processed using 4CRS signature, since information about polygon's vertices is not stored by the signature. It is possible to return if a polygon is not vertex inside related to other polygon when they do not intersect, or that the polygon is vertex inside when it is completely inside the other polygon. On the other hand, when the polygons intersect, but one polygon is not inside the other, it is not possible to return an approximate answer for this operation using 4CRS signatures.

```
real inside(signat4CRS1, signat4CRS2)
 if (signat4CRS1.lengthOfCellSide > signat4CRS2.lengthOfCellSide) then
   return 0;
 if (signat4CRS1.lengthOfCellSide < signat4CRS2.lengthOfCellSide) then
     s4CRS = changeScale(signat4CRS1,signat4CRS2.lengthOfCellSide);
     b4CRS = signat4CRS2;
 else
   s4CRS = signat4CRS1;
   b4CRS = signat4CRS2;
 interMBR = intersectionMBR(s4CRS, b4CRS);
 if interMBR is NULL then /*Does not exist MBR intersection*/
    return 0;
 affinityDegree = 0;
 for each b4CRS cell b that is inside interMBR do
    for each s4CRS cell s that intersects cell b do
       if b.type == Empty and s.type ≠ Empty then
          return 0;
       else
       if (b.type == Weak or b.type == Strong) then
          if (s.type == Weak or s.type == Strong) then
             affinityDegree += expectedArea[s.type,b.type];
          else
             if (s.type == Empty) then
                affinityDegree += 1;
             else /*s.type == Full*/
                return 0;
       else /*b.type == Full × any s.type*/
          affinityDegree += 1;
       nRuns++;
  return affinityDegree / nRuns;
```

Figure 8. Algorithm for returning if a polygon P_1 is inside other polygon P_2, according to their 4CRS signature.

Figure 8 proposes an algorithm for returning if a polygon P_1 is inside other polygon P_2, according to their 4CRS signature. It is important to note that, according to the algorithm that computes the grid of cells (Zimbrao and Souza, 1998), if the cell size of signat4CRS1 is greater than the cell size of signat4CRS2, polygon P_1 represented by assinat4CRS1 is bigger than polygon P_2 represented by assinat4CRS2. Hence P_1 is not inside P_2.

3.2.8 Intersects and Intersection

While the intersect operation returns if two polygons intersect, the intersection operation returns the polygon resulting from the intersection.

For the intersect operation an affinity degree is returned. This value is computed using the expected area of polygon intersection. Figure 9 presents an algorithm to evaluate if two polygons intersect. It is important to highlight that in many cases it is possible to return an exact answer.

```
real intersects(signat4CRS1, signat4CRS2)
 interMBR = intersectionMBR(signat4CRS1, signat4CRS2);
 if (signat4CRS1.lengthOfCellSide < signat4CRS2.lengthOfCellSide) then
   s4CRS = changeScale(signat4CRS1, signat4CRS2.lengthOfCellSide);
   b4CRS = signat4CRS2;
 else
   if (signat4CRS1.lengthOfCellSide > signat4CRS2.lengthOfCellSide) then
      b4CRS = signat4CRS1;
      s4CRS = changeScale(signat4CRS2,signat4CRS1.lengthOfCellSide);
   else
      s4CRS = signat4CRS1;
      b4CRS = signat4CRS2;
 affinityDegree = 0;
 for each b4CRS cell b and s4CRS cell s that intersects and are inside interMBR do
      if (b.type == Full) and (s.type ≠ Empty) then
         return 1;
      else
        if (b.type == Strong) and
           ( (s.type==Strong) or (s.type==Full) then
           return 1;
        else
           if (b.type == Weak) and (s.type==Full) then
              return 1;
           else
              if (b.type == Empty) and (s.type==Empty) then
                 affinityDegree += 1;
           else
              affinityDegree += expectedArea[s.type,b.type];
       nRuns++;
    return affinityDegree / nRuns;
```

Figure 9. Algorithm to evaluate if two polygons intersect.

In the case of the algorithm that returns the polygon resulting from the intersection of two polygons evaluating their 4CRS signatures, we propose the following approach: to create a new 4CRS signature from the intersection of the 4CRS signatures of the polygons, and generate a polygon connecting the medium points of the border cells of the new signature (*Weak* and *Strong* cells). The cell types of the new signature can be set according to the following values: if the value resulting from the intersection of the pair of cells is in the interval (50%, 100%) then the type of the new cell is *Strong*; and, if the value is in the interval (0%, 50%] the type of the new cell is *Weak*. 0% and 100% of intersection define cell types equal to *Empty* and *Full*, respectively. Figure 10 presents the algorithm for computing the polygon resulting from the intersection of 4CRS signatures. It is important to highlight that, along with the polygon that represents the intersection, an affinity degree is also returned in order to show a degree of similarity between the approximate polygon and the polygon that would represent the exact intersection.

```
Polygon intersection(signat4CRS1, signat4CRS2, affinityDegree)
  interMBR = intersectionMBR(signat4CRS1, signat4CRS2);
  if (signat4CRS1.lengthOfCellSide < signat4CRS2.lengthOfCellSide) then
    s4CRS = changeScale(signat4CRS1, signat4CRS2.lengthOfCellSide);
    b4CRS = signat4CRS2;
  else
    if (signat4CRS1.lengthOfCellSide > signat4CRS2.lengthOfCellSide) then
      b4CRS = signat4CRS1;
      s4CRS = changeScale(signat4CRS2,signat4CRS1.lengthOfCellSide);
    else
      s4CRS = signat4CRS1;
      b4CRS = signat4CRS2;
  n4CRS = createSignature(interMBR, Empty); /*Create 4CRS signature of Empty cells*/
  affinityDegree = 0;
  for each b4CRS cell b and s4CRS cell s and n4CRS cell n
                     that intersects and are inside interMBR do
       if (b.type == Full) then
         affinityDegree = 1;
         n.type = s.type;
       else if (s.type == Full) then
         affinityDegree = 1;
         n.type = b.type;
       else if (b.type == Empty) or (s.type == Empty) then
         affinityDegree = 1;
         n.type = Empty;
       else if (b.type == Strong) and (s.type==Strong) then
         affinityDegree = expectedArea[s.type,b.type];
         n.type = Strong;
       else /* ( (b.type == Strong) and (s.type==Weak) ) or */
            /* ( (s.type == Strong) and (b.type==Weak) ) or */
            /* ( (b.type == Weak) and (s.type==Weak) )      */
         affinityDegree = expectedArea[s.type,b.type];
         n.type = Weak;
       nRuns++;
       affinityDegree = affinityDegree / nRuns;
  return createPolygon(n4CRS);
```

Figure 10. Algorithm for computing the polygon resulting from the intersection of 4CRS signatures.

3.2.9 Overlay

The overlay operator is defined by Güting and Schneider (1995) as an operation that allows one partition of the plane be superimposed on another, and allows them to be combined into area-disjoint regions. Partitions are given as sets of objects with an attribute of type in regions. The resulting set of objects contains one object for each new region obtained as the intersection of a region of the first partition with a region of the second partition. Note that the regions of a partition do not have to completely cover the plane. Thus, it is possible that a region of the first partition does not intersect any region of the second partition. In this case it will not be part of any new object. An example of the overlay operation is presented in Figure 11. The algorithm for computing the intersection of polygons from their 4CRS signatures presented in Figure 10 can be used to compute the overlay of partitions of plane.

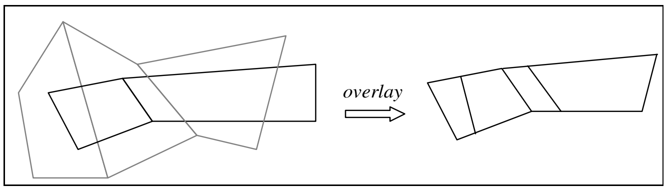

Figure 11. Overlaying two partitions of the plane.

3.2.10 Adjacent, Border in Common, Common border

Two polygons are adjacent if they have at least a portion of their borders in common, which is quite similar to evaluate if two polygons have a border in common. Thus, we are proposing to use the same algorithm that returns an approximate answer if two polygons are border in common and to return if they are adjacent.

One proposal of operation implementation that returns if two polygons are border in common is to employ the expected area, in order to return an affinity degree as the answer. Polygon borders are composed by segments; hence, they do not have area. Therefore, a common border of two polygons can be found only on the overlap of *Weak* or *Strong* cells, which are the cells where the borders are. Thus, it is possible to return an exact answer when there is no overlap of these types of cells. On the other hand, an approximate value is returned. The algorithm is presented in Figure 12.

```
real borderInCommon(signat4CRS1, signat4CRS2)
  interMBR = intersectionMBR(signat4CRS1, signat4CRS2);
  if interMBR is NULL then /*Does not exist MBR intersection*/
    return 0;
  if (signat4CRS1.lengthOfCellSide == signat4CRS2.lengthOfCellSide) then
    s4CRS = signat4CRS1;
    b4CRS = signat4CRS2;
  else
   s4CRS = smallerCellSide(signat4CRS1, signat4CRS2);
    b4CRS = biggerCellSide (signat4CRS1, signat4CRS2);
  affinityDegree = 0;
  nOverlaps = 0;
  for each b4CRS cell b that is inside interMBR do
     for each s4CRS cell s that is inside cell b do
       if b.type == Weak or b.type == Strong then
         nOverlaps += 1;
         if s.type == Weak or s.type == Strong then
            affinityDegree += expectedArea[s.type,b.type];
       else
         if s.type == Weak or s.type == Strong then
            nOverlaps += 1;
            if b.type == Weak or b.type == Strong then
               affinityDegree += expectedArea[s.type,b.type];
  return affinityDegree / nOverlaps;
```

Figure 12. Algorithm to return if two polygons have a border in common.

One proposal of algorithm to return an approximate answer as the border in common of two polygons is to adapt the algorithm presented in Figure 12 in order to create segments connecting the medium points of the overlap of *Weak* × *Weak* cells, *Weak* × *Strong* cells and *Strong* × *Strong* cells. An affinity degree can be returned using the same idea as presented in the algorithm proposed in Figure 12.

3.2.11 Plus and Sum

The plus operator computes the union of two objects, while the sum operator computes the union of a set of objects.

If two polygons do not have MBR intersection, then the polygon that represents the union of these objects is composed by the faces of these two polygons. On the other hand, when the polygons have MBR intersection, a new 4CRS signature is created, according to the algorithm presented in Figure 13. A new polygon is computed from the 4CRS generated signature, connecting the medium points of *Weak* and *Strong* cells (function *computePolygon*).

```
Polygon union(signat4CRS1, signat4CRS2)
  interMBR = intersectionMBR(signat4CRS1, signat4CRS2);
  if interMBR is NULL then /*Does not exist MBR intersection*/
     polygon1 = computePolygon(signat4CRS1);
     polygon2 = computePolygon(signat4CRS2);
     polygon.addFaces(polygon1);
     polygon.addFaces(polygon2);
     return polygon;
  if (signat4CRS1.lengthOfCellSide > signat4CRS2.lengthOfCellSide) then
     b4CRS = signat4CRS1;
     s4CRS = changeScale(signat4CRS2,signat4CRS1.lengthOfCellSide);
  else
     if (signat4CRS2.lengthOfCellSide > signat4CRS1.lengthOfCellSide) then
         b4CRS = signat4CRS2;
         s4CRS = changeScale(signat4CRS1,signat4CRS2.lengthOfCellSide);
  unionMBR = computeUnionMBR(s4CRS.mbr,b4CRS.mbr)
  /*Create 4CRS signature with only Empty cells*/
  n4CRS = createSignature(unionMBR, b4CRS.lengthOfCellSide, Empty);
  for each b4CRS cell b that intersects n4CRS cell n do
     n.type = b.type;
     for each s4CRS cell s that intersects n4CRS cell n do
        if n.type == Empty or s.type == Full then
           n.type = s.type;
        else if n.type == Weak and s.type == Strong then
           n.type = s.type;
     polygon = computePolygon(n4CRS);
  return polygon;
```

Figure 13. Algorithm to compute the union of two polygons using their 4CRS signatures.

3.2.12 Minus

The minus operator applied on polygon P1 related to polygon P2 is composed by the portion of P_1 that does not have intersection with P_2. One proposal for this operation using their 4CRS is to set to *Empty* the cells of P_1 signature that is overlapped by cells of types *Strong* and *Full* of P_2 signature. In order to compute the polygon from the resulting 4CRS signature we must consider that *Full* cells can be part of the new polygon's border, besides *Weak* and *Strong* cells. The algorithm is presented in Figure 14.

3.2.13 Fusion

The fusion operator is defined by Güting and Schneider (1995) as an operation that merges the values of a specified (set of) spatial attribute(s) based on the equality of the values of another (set of) non-spatial attribute(s). For each group of equal non-spatial attribute values a (set of) new spatial value(s) is created as the geometric union of a set of spatial values of the group. In Figure 15, a partition of districts with their land use is given. The task is to compute the regions with the same land use. Neighbor districts with the same land use are replaced by a single region (i.e., their common boundary line is erased). In Figure 15, each of the hatched areas on the left is part of an object describing a district. On the right, after the application of the fusion operator, all areas belonging to the same group form a single regions value and are hatched in the same way. The algorithm presented in Figure 13 can be used to compute the union of the regions resulting from the fusion operation.

```
Polygon minus(signat4CRS1, signat4CRS2)
 interMBR = intersectionMBR(s4CRS, b4CRS);
 if interMBR is NULL then /*Does not exist MBR intersection*/
   return createPolygon(signat4CRS1);
if (signat4CRS1.lengthOfCellSide > signat4CRS2.lengthOfCellSide) then
 b4CRS = signat4CRS1;
   s4CRS = changeScale(signat4CRS2,signat4CRS1.lengthOfCellSide);
 else
   if (signat4CRS2.lengthOfCellSide > signat4CRS1.lengthOfCellSide) th
     b4CRS = changeScale(signat4CRS1,signat4CRS2.lengthOfCellSide);
    s4CRS = signat4CRS2;
   else
     b4CRS = signat4CRS1;
     s4CRS = signat4CRS2;
 for each b4CRS cell b that is inside interMBR do
    for each s4CRS cell s that intersects cell b do
      if s.type == Strong or s.type == Full then
        b.type = Empty;
 polygon = createPolygon(s4CRS1, ConsiderAlsoFullCellsAsBorder);
 return polygon;
```

Figure 14. Algorithm to compute the minus operation of two polygons using their 4CRS signatures.

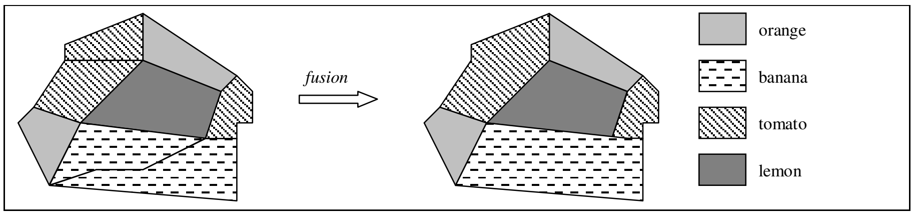

Figure 15. Merging a partition of districts concerning the same land use.

3.2.15 Closest

The closest operator yields that object of an object set whose spatial value is nearest to a spatial reference value. In approximate processing using 4CRS signatures, we can employ the algorithm for computing the distance, presented in Section 3.2.2, to find the nearest object. A proposal for affinity degree calculus is to use the following formula "1 – (maximum distance – minimum distance) / distance". For instance, if the minimum distance from this object to the reference value is equal to 80 and maximum distance is equal to 110, we can estimate the distance between objects as equal to (80 + 110) / 2 = 95, and an affinity degree equal to 68%, since 1 - (110-80)/95 = 1 – 30 / 95 = 0.68.

3.2.16 Decompose

The decompose operator takes a collection of objects with a spatial attribute. It produces a new collection of objects as follows: For each object in the operand set its attribute value is decomposed into its components (a component is a point, a block, or a face). If there are *n* components, then *n* copies of the original object are produced each of which has one component as the value of a new attribute.

In this work we are concerned with polygons represented by their 4CRS signatures. The components of polygons are faces, and the components of a signature of a polygon are signatures for each face. Hence the decompose operator applied on a 4CRS signature of a polygon produce new objects where each object has an attribute with value equal to a signature of a face of the original polygon.

4. Conclusions

This work proposed new algorithms for approximate query processing in spatial databases using raster signatures. The target is to provide an estimated result in orders of magnitude less time than the time to compute an exact answer, along with a confidence interval for the response. We extended the proposals of Azevedo *et al.* (2004) and Azevedo *et al.* (2005) of using Four-Color Raster Signature (4CRS) (Zimbrao and Souza, 1998) for fast and approximate processing of queries on polygon datasets. By doing so, the exact geometries of objects are not processed during the query execution, which is the most costly step of the spatial query processing since it requires to search and to transfer large objects from disk to the main storage (Brinkhoff *et al.*, 1994; Lo and Ravishankar, 1996). Also, the exact processing algorithm needs to use complex CPU-time intensive algorithms for deciding whether the objects match the query condition (Brinkhoff *et al.*, 1993). There are many scenarios and applications where a slow exact answer can be replaced by a fast approximate one, provided that it has the desired accuracy, as presented in Section 2.

We proposed to use 4CRS for approximate processing of many spatial operations. These operations were enumerated in Section 3.1 according to the classification proposed by Güting *et al.* (1995) and Güting and Schneider (1995) in the Rose Algebra. In Section 3.2 we presented proposals of algorithms for those operations, which are the main contributions of this work. The experimental evaluation is not addressed in this work; it is on going work developed on Secondo (Güting *et al.*, 2005), which is an extensible DBMS platform for research prototyping and teaching.

As future work, we plain to implement and to evaluate algorithms involving other kinds of datasets, for example, points and polylines, and combinations of them, point × polyline, polyline × polygon and polygon × polyline. Azevedo *et al.* (2003) proposed a raster signature for polylines named Five-Colors Directional Raster Signature (5CDRS). In that work, 5CDRS signature was employed as a geometric filter for processing spatial joins in multiple steps of datasets made by polylines. The results obtained were quite good. Our proposal is to evaluate 5CDRS signature for approximate query processing of polyline datasets. Besides, we also expect to evaluate the 5CDRS and 4CRS signatures together, testing the approximate query processing of queries involving polylines and polygons. The results obtained employing both signatures as geometric filter in multi-step spatial join were also quite good, as presented by Monteiro *et al.* (2004).

5. References

Aronoff, S. (1989), Geographic Information Systems, WDL Publications, 1st edition.

Azevedo, L. G., Monteiro, R. S., Zimbrao, G., Souza, J. M. (2003), "Polyline Spatial Join Evaluation Using Raster Approximation". In: *GeoInformatica, Kluwer Academic Publishers* , vol. 7, n. 4, pp. 315-336.

Azevedo, L. G., Monteiro, R. S., Zimbrao, G., Souza, J. M., 2004, "Approximate Spatial Query Processing Using Raster Signatures". In: *Proceedings of VI Brazilian Symposium on GeoInformatics*, pp. 403-421, Campos do Jordao, Brazil, Nov.

Azevedo, L. G., Zimbrao, G., Souza, J. M., Güting, R. H. (2005), "Estimating the overlapping area of polygon join". In: *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, to be published, Angra dos Reis, Brazil, Aug.

Barbara, D., Dumouchel, W., C. Faloutsos, *et al*. (1997), "The New Jersey data reduction", *Bulletin of the Technical Committee on Data Engineering*, v. 20, n. 4 (Dec), pp. 3-45.

Brinkhoff, T., Kriegel, H. P., Schneider, R (1993), "Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Spatial Database Systems". In: *Proceedings of the Ninth International Conference on Data Engineering*, pp. 40-49, Vienna, Austria, Apr.

Brinkhoff, T., Kriegel, H. P., Schneider, R., Seeger, B. (1994), "Multi-step Processing of Spatial Joins", *ACM SIGMOD Record*, v. 23, n.2 (Jun), pp. 197-208.

Das, A., Gehrke, J., Riedwald, M. (2004), "Approximation Techniques for Spatial Data". In: *Proceedings of the 2004 ACM-SIGMOD International Conference on Management of Data*", pp. 695-706, Paris, France, Jun.

Dobra, A., Garofalakis, M., Gehrke, J. E., Rastogi, R. (2002), "Processing complex aggregate queries over data streams". In: *Proceedings of the 2002 ACM-SIGMOD International Conference on Management of Data*, pp. 61-72, Madison, Wisconsin, USA, Jun.

Faloutsos, C., Jagadish, H. V., Sidiropoulos, N. D. (1997), "Recovering information from summary data". In: *Proceedings of 23rd International Confeference on Very Large Data Bases*, pp. 36-45, Athens, Greece, Aug.

Gibbons, P. B., Matias, Y., Poosala, V. (1997), *Aqua project white paper.* Technical Report, Bell Laboratories, Murray Hill, New Jersey, USA.

Güting, R. H. (1994), "An Introduction to Spatial Database Systems", *The International Journal on Very Large Data Bases*, v. 3, n. 4 (Oct), pp. 357-399.

Güting, R.H., Almeida, V., Ansorge, D., Behr, T., Ding, Z., Höse, T., Hoffmann, F., Spiekermann, M., and Telle, U. (2005), "Secondo: An Extensible DBMS Platform for Research Prototyping and Teaching", In *Proceedings of 21st International Conference. on Data Engineering (ICDE'05)* (Tokyo, Japan, April 5-8, 2005), IEEE Computer Society, Washington, DC, USA, 1115-1116.

Güting, R.H., Schneider, M. (1995), "Realm-Based Spatial Data Types: The ROSE Algebra", *The International Journal on Very Large Data Bases*, v. 4, n. 2 (Apr), pp. 243 - 286.

Güting, R. H., De Ridder, T., Schneider, M. (1995), "Implementation of the ROSE Algebra: Efficient Algorithms for Realm-Based Spatial Data Types". In: *Proceedings of the 4th International. Symposium on Large Spatial Databases Systems*, pp. 216-239, Portland, USA, Aug.

Han, J., Kamber, M. (2001), Data Mining: concepts and techniques, Morgan Kaufmann publishers, 1st edition.

Hellerstein, J. M., Haas, P. J., Wang, H. J. (1997), "Online aggregation". In: *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 171-182, Tucson, Arizona, USA, May.

Ioannidis, Y. E., Poosala, V. (1995), "Balancing histogram optimality and practicality for query result size estimation". In: *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pp. 233-244, San Jose, California, USA, May.

Jagadish, H. V., Mumick, I. S., Silberschatz, A. (1995), "View maintenance issues in the chronicle data model". In: *Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 113-124, San Jose, California, USA, May.

Lo, M. L., Ravishankar, C. V. (1996), "Spatial Hash-Joins". In: *Proceedings of the 1996 ACM-SIGMOD International Conference on Management of Data*, pp. 247-258, Montreal, Quebec, Canada, Jun.

Madria, S. K., Mohania, M. K., Roddick, J. F. (1998), "A Query Processing Model for Mobile Computing using Concept Hierarchies and Summary Databases". In: *The 5th International Conference of Foundations of Data Organization,* pp. 147-157, Kobe, Japan, Nov.

Monteiro, R. S., Azevedo, L.G., Zimbrao, G., Souza, J. M. (2004), "Polygon and Polyline Join Using Raster Filters". In: *Proceedings of the 9th International Conference on Database Systems for Advances Applications*, pp. 255-261, Jeju Island, Korea, Mar.

Papadias, D., Kalnis, P., Zhang, J. *et al.* (2001), "Efficient OLAP Operations in Spatial Data Warehouses". In: *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, pp. 443-459, Redondo Beach, CA, USA, Jul.

Roddick, J., Egenhofer, M., Hoel, E., *et al.* (2004). "Spatial, Temporal and Spatiotemporal Databases Hot Issues and Directions for PhD Research", *SIGMOD Record*, v. 33, n. 2 (Jun), pp. 126-131.

Samet, H. (1990), The Design and Analysis of Spatial Data Structure, Addison-Wesley Publishing Company, 1st edition.

Tao, Y., Sun, J., Papadias, D. (2003), "Selectivity estimation for predictive spatio-temporal queries". In: *Proceedings of the 19th International Conference on Data Engineering*, pp. 417-428, Bangalore, India.

Zimbrao, G., Souza, J. M. (1998), "A Raster Approximation for Processing of Spatial Joins". In: *Proceedings of the 24rd International Conference on Very Large Data Bases*, pp. 558-569, New York City, New York, USA, Aug.