

Indexing Field Values in Field Oriented Systems : Interval Quadtree

Myoung-Ah KANG
Ph.D. Student
LISI, INSA de Lyon
F- 69 621, France
Tel.+33 4 72 43 88 98
makang@if.insa-lyon.fr

Sylvie SERVIGNE
Associate Professor
LISI, INSA de Lyon
F- 69 621, France
Tel. +33 4 72 43 88 98
servigne@if.insa-lyon.fr

Ki-Joune LI
Associate Professor
CS Dept., Pusan National
University, South Korea
Tel.+82 51 510 2299
lik@hyowon.cc.pusan.ac.kr

Robert LAURINI
Professor
LISI, INSA de Lyon
F- 69 621, France
Tel. +33 4 72 43 88 98
laurini@if.insa-lyon.fr

ABSTRACT

With the extension of spatial database applications, field oriented systems emerge as an important research issue in order to deal with continuous natural phenomena during the last years. It however has a large volume of data and efficient indexing methods for field data are necessary to overcome the performance obstacle. In special, we introduce indexing methods for field value queries (i.e. searching some regions where the temperature is more 20 degrees). We introduce the concept of *subfield* and show how we make use of this concept to index field values in field oriented systems. We present two implementation methods based on Quadtree space subdivision. We modify traditional linear quadtree implementation method for field value query processing using *subfields*. We analyze the performance of our methods. Experimentation with real terrain data shows that proposed indexing methods improve the query processing time of field value queries in comparison with the case of no indexing method.

Keywords

field oriented systems, indexing method, field values, subfield.

1. INTRODUCTION

The concept of field has been widely discussed for dealing with natural and environmental continuous phenomenon including the earth sciences, such as oceanography, atmospheric physics, and geology, which involve large data set and their analysis [3, 4, 5, 18]. The technology of field databases serves as an important bridge for extending spatial database to scientific databases.

A field can be defined as a function over space and time from a mathematical point of view as follows,

$F : S \times T \rightarrow R^n$, where S and T are spatial and temporal domains, respectively.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. CIKM '99 11/99 Kansas City, MO, USA
© 1999 ACM 1-58113-075-9/99/0010...\$5.00

Field data is either scalar or vectorial depending on the result of the corresponding function. When the result is of single value, such as the temperature at a point, the field becomes scalar. On the other hand, the field is vectorial, when the result is of multiple values. And the most requested queries for analyzing field phenomena can be roughly divided into two categories as follows,

1. Queries based on a given position: e.g. what is the value at a given point p ?
2. Queries based on a given field value: e.g. what is the region where the rainfall is more than 2000 mm per year ?

The first type of queries is to find the field values at a given position. On the other hand, the second type, which is the inverse of the first one, inquires the region, where the field has a given value. Although the two queries are related with field, their processing methods are different and the processing cost is expensive due to the large volume of data of field.

Even though it might be possible to represent a field in a multi-dimensional space and a conventional spatial indexing and query processing method may be applied, but it will give a poor performance, since the properties of field data are quite different from those of spatial data. Suppose that we have a field of temperature on a certain region. The field can be represented as a surface on the region. And we may apply for example R^* -tree as indexing method. However, it gives a poor performance due to the high degree of overlapping of field along the axis of elevation, although it works well on 2-dimensional or normal 3-dimensional spaces. We therefore need a new approach for processing field data, considering its characteristics.

Despite the importance, no significant attention has been paid and no remarkable work has been done on the indexing and query processing for field data. Although a certain amount of researches are found, they still focus on the representation or modeling issues for field. Most of the researches deal with the issues of the representation of continuous fields and appropriate data models [9, 10, 11]. In [14] an object oriented model to implement a new estimation method for field data has been specified and the refined object oriented model which permit to change dynamically the estimation method has been studied [7].

In this paper, we will propose new indexing schemes, which are the most fundamental base for developing and implementing query processing methods of field data. The proposed indexing

schemes exploit the continuity of field, which is an important property of field. And our methods are based on the division of a field into several *subfields* in the context of the homogeneity of field values. The notion of subfield allows an approximate search in the level of subfields instead of the exhaustive search on the entire field, by discarding non-qualified subfields for a given search condition.

The organization of this paper is as follows. In section 2, we will introduce the nature of field and some notations for field representation. We will show the overall procedures of query processing in field-oriented systems and particularly the difficulties of query processing based on field values in that section. In section 3, we will propose a new concept *subfield* and show how to use it for indexing field values. In section 4, we will describe two indexing methods based on Quadtree structures in order to exploit the subfields. We present results of some experiments, which show the improvement of the performance by using our indexing methods in section 5. And we conclude our paper in section 6.

2. Field and Query Processing in Field Databases

2.1 Field representation

While a conventional database consists of a set of entities, a field represents a continuous phenomenon. In other words, certain types of continuous values are assigned at a point in a field. Field is scalar or vector according to the number of values at a point. While a single value is defined at a given point in scalar field, multiple scalar values are assigned at a given point in vector field. Temperature is a typical example of scalar field and wind is an example of vector field since it contains two scalar fields, such as two types of speed along *x*-axis and *y*-axis respectively. It is therefore possible to represent a vector field with multiple scalar fields. For this reason, we will focus on the scalar field in this paper.

In many cases, the phenomenon under consideration can not be sampled in every point belonging to the study area: for example groundwater, temperature. Thus in general the data set is measured in some points or in some zones and the spatial interpolation or extrapolation methods are used in order to estimate the value of properties at not sampled locations.

The estimation method is therefore an important basis of the low level representation of field. In general a low level representation of a field consists of a set of sample field objects and an estimation method. Depending on how to select objects for sampling and on the estimation method, the low level representation methods of a field are classified as follows,

1. The methods belonging to the first category are based on the regular partitioning of the field space. They consist of field objects of square or triangle, which partition the entire space with regular subspaces. Each object has a representative field value and all the points within an object have the same field value. DEM (Digital Elevation Model) is a typical example of this type. In order to compute the representative value of an object, a certain number of estimation methods, e.g., kriging, B-spline, FDM

(Finite Differences Method), work on such discretized field space. [12, 14]

2. An example of the second category is TIN (Triangulated Irregular Network). The sampled field objects become triangles in TIN and we estimate the value of field in a triangle by interpolation with the values of the three vertices of the triangle. This method is based on the irregular decomposition of a space. Contour line is another example of the second category. A space is partitioned to nested bands and an appropriate estimation method is employed to find the value of field between contour lines.

For the reason of simplicity, we only consider DEM and TIN as the low level representation of field in this paper, which are the most typical examples of the first and the second methods respectively.

The low level representations of field require a large volume of data. The methods listed above allow to represent field data with relatively small volume. In order to improve the accuracy of field data, we are obliged to increase the number of sampled objects and this result in a large volume of data. The large volume of field data obviously degrades the performance of systems. We need therefore invent methods to improve the performance in spite of the large volume.

The performance of field database systems mainly relies on the indexing and query processing mechanism. Despite of the importance, little attention has been paid on these issues. Efficient indexing and query processing methods for field data are expected. In the next section, we will therefore investigate on the field indexing and query processing.

2.2 Field Queries

In order to process field data, we need a set of basic field operators for search and retrieval on field databases. Suppose that we wish to find a region where temperature is higher than a given value. In this case, we need a field operator, which is an extension of spatial operator. While we could not give the complete list of field operators, since they depend on the types of application, we give a list of fundamental field query types, which are common in almost cases.

A field query consists of two variables, namely spatial variable *S*, and value variable *V* concerning field value. Since a field can be defined by an equation, $f(V, S) = 0$, a field query can be represented by a function of one specified variable for calculating the other variable. Each variable can be a value or a set of values. For example, *S* can be a point, a line or a region. And *V* can be also a value or an interval. We may consider the temporal variable on field query, but in this paper we exclude the issue related with temporal aspect.

The response of the query depends on the type of *S* and *V*. Now we can classify the types of queries as follows,

$$Q1 : f^1(S) = V \text{ (Find field value on a given area } S\text{.)}$$

$$Q2 : f^1(V) = S \text{ (Find region with a given field value } V\text{.)}$$

When the query region S in Q1 is a point, the query result may be field value or gradient at the point. However, it can be differently interpreted, when the query region is an area. For example, it may be the minimum, maximum, integral or average of field values on a region S . It means that Q1 includes query such as “Find the average field value on region S .” Anyway retrieving field values at a point or on a region is demanded for processing these queries.”

Q2 is to find the region with a given field value. It is however possible to extend this query type to find the region where the field value is given as an interval $[V_1, V_2]$. For example, “Find the region where temperature is between $[20^\circ C, 25^\circ C]$ ” belongs to this type of query.

We can also extend the scope of field value to gradient or integral in Q2. For example, the query such as “Find the region where the slope is greater than 45° .” The gradient and integral are very helpful concept to analyze field phenomenon in various domains, such as meteorology. The velocity of wind can be derived from the gradient of air pressure.

2.3 Query Processing in Field Databases

The query processing methods in field database systems depend on the low level representation and query types. The first type of query is relatively easy to process, since it is sufficient to retrieve field object corresponding to the specified area. For example in DEM, we only need to find the meshes contained or overlapped with the query region. And we simply retrieve the field values of the meshes and compute the maximum or average if necessary.

In TIN, we need additional computations to process the queries of Q1. It includes the estimation of the field value from the triangles, which is however a trivial task. And it would be necessary to index triangles, in order to search the corresponding triangles with the query region due to the amount of the triangle objects in a field. Conventional spatial indexing methods, such as R-tree or its variants can be used for this purpose [2, 8, 16, 17].

It is relatively more difficult to process the queries of Q2 than Q1. For example, suppose that a query is given such like “Find the zone where the rainfall is more than 200mm and less than 300mm?”. Without indexing, we should scan all the database to find the region. A proper indexing for field database is a crucial function for improving the performance of field query processing. The property of field data is not fully considered by the conventional spatial indexing methods. It means that they are based on the assumption that the objects are separated, which is no longer true in field. A field is in fact a large object that covers the entire domain and no longer separable. In order to reuse conventional spatial indexing methods, we should therefore find a method to divide a field into a number of subfields. In subsequent sections, we will present the concept of subfield and field indexing methods by using subfield.

3. Subfield and Field Query Processing

3.1 Subfield

In this section, we present the concept *subfield* for the indexing field values. Subfield is a subspace, in which field values are close together. In other words, the difference between maximum field

value and minimum field value in a subfield is less than a given threshold. The i -th subfield of F is defined as follows,

$$F_i = (A_i, V_{min,i}, V_{max,i}),$$

where A_i represents the area that F_i covers and $V_{min,i}$ and $V_{max,i}$ mean the minimum and maximum of F_i respectively.

We may append other kinds of values to $V_{min,i}$ and $V_{max,i}$, if necessary, for example, the average of field values of subfield. Figure 1 shows an example of subfields, where a field is divided into several subfields. In this figure, subfields are represented by rectangles. For the reason of simplicity, we assume that the shape of area A_i be rectangle as Figure 1. The height of rectangles implies the difference between minimum and maximum values in a subfield, which are limited to a given threshold.

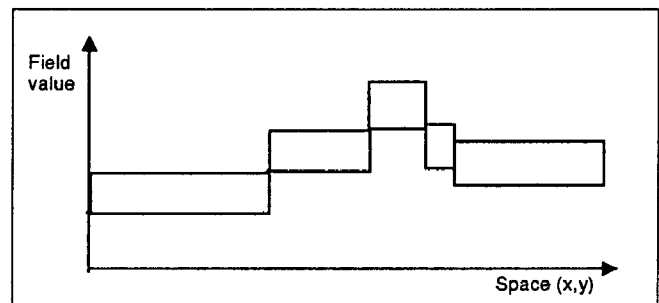


Figure 1. Subfields in a 2-d field

We may divide a field in various ways to obtain subfields, but an efficient way of division is region quadtree [12, 15, 16]. Quadtree is originally used for representing bitmap images or spatial objects in raster model. We subdivide a bitmap image into four quadrants in recursive way until each quadrant satisfies a certain condition.

By the same way, we divide the entire field into four subfields until each subfield satisfies the condition that the difference between the maximum and minimum field values of each subfield be less than the threshold. Figure 2 is an example of division by region quadtree where the threshold is defined as 10 in two dimensional space. The numbers in each subfield mean the interval of field values, which is given as maximum and minimum field values respectively. And Figure 3 shows the corresponding quadtree structure with the number and the interval of each node. The leaf nodes represent the subfields shown in Figure 3.

The concept of subfield is useful in processing field queries presented in section 2, and we will explain it in the next section.

60, 66	65, 75	73, 80	
58, 69	64, 73		
64, 70		68, 78	64, 74
		59, 68	54, 62, 55, 65 49, 59, 46, 56

Figure 2. Subfields with threshold 10

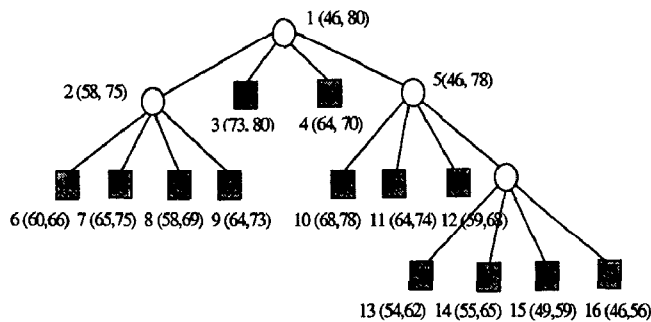


Figure 3. Quadtree of Figure 2

3.2 A Field Query Processing Strategy with Subfield

As we have discussed in section 2, the field queries belonging to Q1 are simple to process, since a spatial indexing and corresponding query processing procedure are enough to find the field value on a given region. The queries of Q2 are however more difficult to evaluate than those of Q1. In this section, we focus on the query processing for Q2, for this reason.

In fact, the subfield can be used as MBR (Minimum Bounding Rectangle) in spatial indexing method such as R*-tree. It means that we can make use of *filtering and refinement strategy* used in R*-tree [2]. Suppose a field query “find the region where the temperature is between [20°C, 25°C]”. Then the procedure of field query processing indexing is divided into three steps as follows,

- Step 1 (filtering step) : find all subfields whose interval overlaps with [20°C, 25°C].
- Step 2 (refinement step) : retrieve all field objects in the selected subfields by step 1.
- Step 3 (estimation step) : estimate the region where temperature is between [20°C, 25°C] based on the low level representation method of field.

Since the above procedure slightly differs from the filtering and refinement strategy proposed by [2, 17], we call it *filtering, refinement and estimation strategy*. The filtering step allows to avoid checking all objects in field database by selecting the candidate subfields which probably satisfy the query condition. During the refinement step, we retrieve the field objects from field databases, which are squares or triangles depending on the low level representation, contained by the candidate subfields.

While the spatial query processing is completed by the second step, the query processing for fields needs an additional step of estimation. The estimated region is computed during the third step by using the field objects depending on the low level representation method of field. For example, the region is computed by interpolation using the triangles retrieved by the second step, if the field is represented by TIN.

Figure 4 illustrates the field value query processing procedure. Note that the XY plan is simplified to (x,y) axis in this figure. We also suppose that TIN is used for the low level representation of field and that a query such as “Find the zone where temperature is more than 20°C and less than 25°C” is given. For the filtering step, we select three subfields, 1, 2, and 3, which overlap with query interval. And we retrieve the triangles contained or intersected by the selected subfields during refinement step. Finally, we compute the region where the temperature is between [20°C, 25°C] with the retrieved triangles.

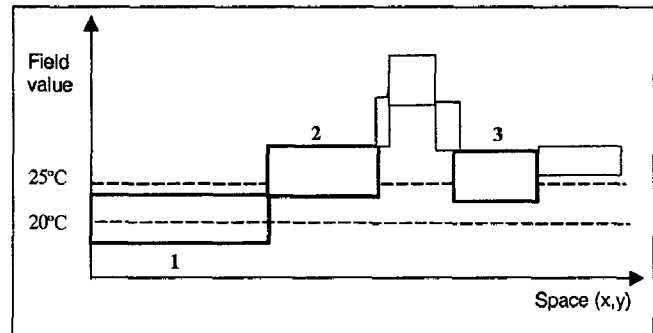


Figure 4. Indexing for field value queries with subfields

Among the three steps, the performance of query processing is mainly influenced by the first and second steps, since the number of disk accesses is determined by them. In particular, field indexing plays an important role in reducing the number of disk accesses during the first and second steps. We will therefore propose several field indexing methods in the next section.

4. Field Value Indexing

4.1 Field Value Indexing by Pointer-based Quadtree

The most obvious implementation method of quadtree is Pointer-based Quadtree, where each node in quadtree has four pointers to its child nodes. Each intermediate node in quadtree has the interval of field, that is maximum and minimum field values as Figure 3 and four pointers to its child nodes. By contrast, the leaf node has the interval of field values and disk address where

the field objects within subfield covered or intersected by the leaf quadrant are stored instead of pointers to its child nodes. It means that we can retrieve the field objects within a subfield by accessing the corresponding leaf node.

With this structure of Pointer-based Qudtree, we can process field query of Q2 as described in the previous section by filtering, refinement and estimation strategy. Suppose that we would like to find region where the temperature is between [20°C, 25°C]. In this case, the interval becomes a query field value. At first, we traverse Pointer-based Quadtree by examining if the interval of each node overlaps with the query field value at each level. The nodes which do not intersect with the query field value are excluded and the candidate subfields are selected. It means that unnecessary subfields very little influence on the performance of query processing. And we finally retrieve the field objects contained by the candidate subfields and estimate the region if necessary. The Pointer-based Quadtree helps for searching the candidate field objects as well. The algorithm is explained by Algorithm 1.

```

Procedure Field value indexing by Pointer Quadtree
Input :  $P$  (root node of quadtree),  $I$  (given field values)
Output :  $B$  (set of field objects)
 $A \leftarrow \{P\}$ .
 $B \leftarrow \{\}$ .
while  $A$  is not empty,
     $m \leftarrow$  delete a node in  $A$  .
    if  $m$  is non leaf node and  $interval(m)$  overlaps with  $I$ ,
        then insert four child nodes into  $A$ .

    if  $m$  is leaf node and  $interval(m)$  overlaps with  $I$ ,
        then insert the field objects of  $m$  into  $B$ .
end While
end Procedure

```

Algorithm 1. Field value indexing by Pointer-based Quadtree

Suppose that a field value query is given as "Find the zone where the field value is 76 ?" in Figure 3. We can obtain filtered subfield 3 and 10 by the search paths {1, 3}, {1, 5, 10} respectively by Algorithm 1.

Although this algorithm works well, it has several weak points in implementation as follows,

1. The first is a traditional problem of the Pointer-based Quadtree implementation. The data structure of Pointer-based Quadtree needs relatively a large amount of storage space and it leads to frequent disk accesses when following pointers.
2. When the regions satisfying the query condition of field interval, for example [20°C, 25°C], are scattered on several branches of quadtree, we must traverse many paths in the quadtree. It means that it is difficult to conserve the proximity with regard to field values in Pointer-based

Qudtree. It results in a degradation of performance. For example, when we search the subfields overlapping the query field value 59 in Figure 2 and 3, we find that the subfields satisfying query condition are {8, 12, 14, 15, 16}. However these subfields are scattered on the many branches although the intervals of subfields are very similar such as (58, 69), (59,68), (54, 62), etc.. The reason is why Pointer-based Quadtree is not totally hierachical with respect to the interval of field values of subfields.

In the next subsection, we will propose an improvement of Pointer-based Quadtree to overcome these problems and enhance the performance.

4.2 Interval Quadtree

In order to avoid the disadvantages of Pointer-based Quadtree implementation, linear quadtree represented by a list of values of leaf nodes are generally used [15, 16]. To speed up the search, the elements of the list are usually reorganized using a tree structure in the form of a B-tree or one of its variations. In general the key served for the tree structure is the value of locational code.

However this method is not available for the field value query processing. Because the search condition is not the position of nodes but the interval of field values. Thus we will make a hierarchic tree structure by using the intervals of field values of subfields as the key. In this paper, we call this method *Interval Quadtree*. For this tree struture with the intervals field values, we exploit the transformation method [6].

A subfield element of the list of leaf nodes for Interval Quadtree is the form

$$(S_{id}, N_{id}, Size, V_{min}, V_{max}, D),$$

where S_{id} is a subfield identifier. N_{id} is the identifier of the corresponding leaf node such as peano key and $Size$ is the length of the node. V_{min} and V_{max} are the interval data of subfield; the minimum and maximum field value, and D is disk address where the field objects within subfield are stored.

As Figure 5 illustrates, each interval of subfields becomes a point in the transformed space and we utilise an existing spatial method for indexing the points. In this paper we make use of R*-tree known as one of the most efficient spatial methods.

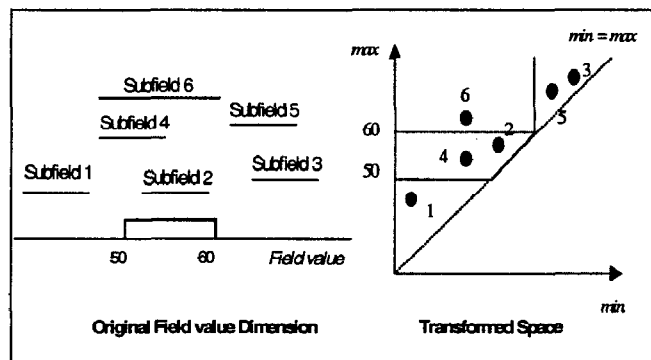


Figure 5. Indexing by Transformation method

In Figure 5, suppose that the lines represent the intervals of field values of each subfields. For example, when a query such as “*find the region where field value is between 50 and 60*” is given, the subfields 2, 4, 6 in the shaded query region will be found and we will check the field objects in the subfields 2, 4, 6.

5. Implementaion and Performance

For the performance measurements of the field value queries indexing methods proposed in this paper, we utilise the real field data of a region in America whose the physical representation is DEM and whose resolution is 2048 * 2048.

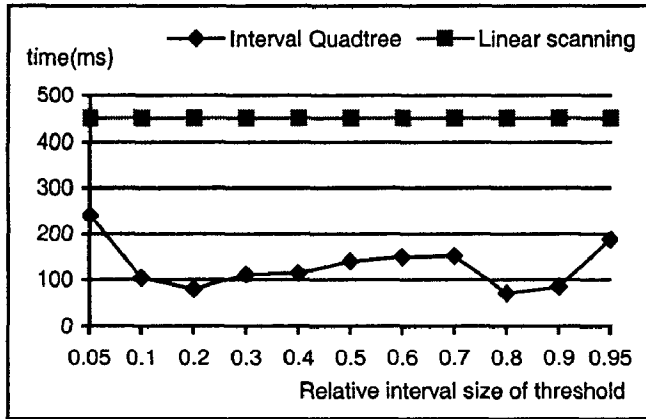


Figure 6. Comparison between Interval Quadtree and Linear scanning

In Figure 6 the Interval Quadtree with R*-tree is compared with linear scanning seach by means of the average query processing time for 50 randomly generated query field values. The page size for scanning the disk and for the construction of R*-tree is 1024 bytes. We varied the relative interval of threshold for subfields over the interval between the maximum and the minimum field value in the total field values from 0.05 to 0.95. The result shows that the Interval Quadtree improves the performance of field value queries.

While we can see in the graph of Interval Quadtree in Figure 6, in the proposed indexing method the interval size of threshold which decides the number of subfields is an important issue to be considered for performance. In other words, the performance of this indexing method depends on the different relative interval of threshold. We find out that the performance is better when the relative interval size of threshold is round 0.2 and 0.8.

When the relative interval size of threshold is 1 which means that there is only a subfield, the field value query processing time with Interval Quadtree indexing method is almost similar to the linear scanning time. When it becomes 0, the performance with Interval Quadtree is also very poor since each cell of test DEM data becomes a subfield and there is almost no filtering effect for query processing.

As mentioned before, the threshold is related with the number of obtained subfields. Table 1 shows the number of subfields obtained for Interval Quadtree, the height of R*-tree, the time for filtering step and that of the refinement step according to the relative interval size of threshold.

A very small threshold leads to too many subfields which demande more time for filtering and unnecessary disk accesses because it spreads field objects which can be read with only one disk access time over several subfields. By contrast, a very large threshold generates many field obejcts to be refined with a poor filtering rate causing the poor performance. And we also think that there may be a correlation between the page size and the size of field objects in a subfield. To find the optimal threshold value is important and interesting. We leave this as one of our future works.

Relative interval size of threshold	Number of subfields	Height of R*-tree	Filtering time(ms)	Refineme-nt time(ms)
0.05	52204	4	145	96
0.1	15724	3	36	69
0.2	3889	3	7	76
0.3	2077	3	3	109
0.4	1423	3	2	113
0.5	898	2	1	139
0.6	406	2	1	149
0.7	37	2	0.1	162
0.8	19	1	0.01	72
0.9	13	1	0.01	72
0.95	4	1	0.01	170

Table 1. Some variables according to the relative interval size of threshold

We compared the two implementation methods proposed in the previous section; Pointer-based Quadtree and Interval Quadtree. The R*-tree indexing method is also used in the comparison since it is a popular bench mark. Thus we constructed R*-tree with 3-d rectangles of subfields obtained, 1-d for the interval of field values and 2-d for the spatial information.

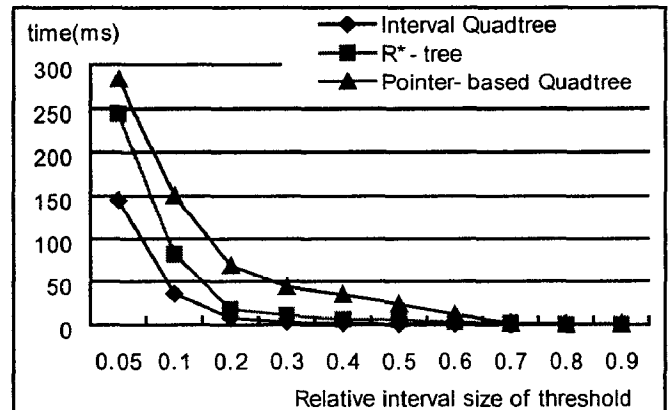


Figure 7. Comparison of proposed indexing structures

We measured the average time required for retrieving satisfied subfields with 50 field value queries since the refinement processing time is the same for all the proposed methods. Figure 7 shows that Interval Quadtree is more efficient than the others. Due to the overlap on the axis of field value between subfields rectangles, the performance of R*-tree is worse than that of Interval Quadtree but is better than that of Pointer-based Quadtree.

In Figure 7 and Table 1, when we calculate the time of the query processing for filtering and refinement steps, we can conclude that indexing methods proposed in this paper, even if it is Pointer-based Quadtree implementation, improve the performance for the field values queries in comparison with linear scanning.

6. Conclusion

To process the field value queries in field-oriented systems is very expensive due to the large volume of data. In this paper, we have proposed new indexing structures which index on the field value domain in field-oriented systems. The main idea is to divide a field into several *subfields* whose the difference between the maximum field value and the minimum field value in the area becomes less than a given threshold. In this paper, we make use of the way of Quadtree space subdivision. The goal is to avoid to search all field objects by approximate search as regards subfields. This approach is derived from filtering and refinement spatial query processing strategy.

We proposed two indexing structures in order to implement the concept of *subfield* and showed how to use these structures to process field value queries. The first indexing structure is Pointer-based Quadtree, while the second indexing structure: Interval Quadtree, is the linear quadtree with the hierarchic structure with the interval of field values of subfields used as the search key. For the hierarchic structure, R*-tree is used by the transformation method which transform an interval into a point in this paper.

We evaluated the two structures and pure R*-tree indexing method by considering subfields as rectangles. Interval Quadtree structure is more efficient than the others. The performance measurements showed that the indexing structures proposed in this paper improve the field value query processing time, compared to linear scanning.

The intervals of field values of subfields can be considered as the time intervals on temporal database. It means to make use of the indexing method for time interval query such as interval B-tree instead of R*-tree using transformation method proposed in this paper for Interval Quadtree. The experiment to compare the performance of Interval Quadtree using R*-tree with that using the interval B-tree [1] would be an interesting project in the future. As mentioned above, it is also worthwhile to find the optimal threshold of the interval of subfields.

7. REFERENCES

- [1] ANG C., TAN K. (1995) "The Interval B-tree", Information Processing Letters, 53(2), pp.85-89.
- [2] BECKMANN ?, KREIGEL R., SCHNEIDER R., SEEGER B. (1990) "The R*-tree: An efficient and robust access method for points and rectangles", Proceedings of 1990 SIGMOD, pp. 322-331.
- [3] BURROUGH PA, FRANK AU (1996) "Geographic Objects with Indeterminate Boundaries". Taylor and Francis. 345p.
- [4] COUCLELIS, H. (1992). "People Manipulate Objects (but Cultivate Fields): Beyond the Raster-vector Debate in GIS", in: Frank, A. U., Campari, I. and Formentini, U. (Eds), "Theories and Methods of Spatio-temporal Reasoning in Geographic Space", Lecture Notes in Computer Science 639, pp. 65-77, Berlin: Springer.
- [5] COUCLELIS, H. (1996) "Towards an Operational Typology of Geography Entities with Ill-defined Boundaries", in "Geographic Objects with Indeterminate Boundaries", edited by Peter A. Burrough and Andrew U. Frank, Taylor and Francis, pp. 45-55.
- [6] FALOUTSOS C. (1996) "Searching Multimedia Databases by Contents". Kluwer Academic Press. 155p.
- [7] GORDILLO S., BALAGUER F. (1998) "Refining an Object-oriented GIS design Model: Topologies and Field Data". Proceedings of the 6th International Symposium on Advances in Geographic Information Systems (ACM-GIS), pp.76-81, Washington, D.C., USA.
- [8] GUTTMAN A. (1984) "R-trees – a Dynamic Index Structure for Spatial Searching", Proceedings of the ACM SIGMOD Conference, pp.47-57, Boston MA.
- [9] KEMP, K. (1993). "Environmental Modelling with GIS: A Strategy for Dealing with Spatial Continuity", Technical Report No. 93-3, National Center for Geographical Information and Analysis, University of California, Santa Barbara.
- [10] KEMP K. (1997) "Fields as a Framework for Integrating GIS and Environmental Process Models. Part 1: Representing Spatial Continuity; Part 2: Specifying field variables". Transactions in GIS, Vol 1, n 3, pp 219-234 and pp. 235-246.
- [11] LAURINI R, PARIENTE D. (1996). "Towards a Field-oriented Language: First Specifications". In "Geographic Objects with Indeterminate Boundaries" Edited by Burrough PA, Frank AU. Taylor and Francis. pp. 225-235.
- [12] LAURINI R., THOMPSON D. (1992) "Fundamentals of Spatial Information Systems" Academic Press.
- [13] NASCIMENTO MA, SILVA JRO (1998) "Towards Historical R-trees", Proceedings of ACM Symposium on Applied Computing (ACM-SAC).

- [14] PARIENTE D.(1994) "*Estimation, Modélisation et Langage de Déclaration et de Manipulation de Champs Spatiaux Continus*". PhD, Institut National des Sciences Appliquées de Lyon, 6 December, 192p.
- [15] SAMET H. (1990) "*The Design and Analysis of Spatial Data Structures*", Addison-Wesley, Reading MA.
- [16] SAMET H. (1990) "*Application of Spatial Data Structures*", Addison-Wesley, Reading MA.
- [17] SELLIS. T., ROUSSOPOULOS, N. and FALOUTSOS, C. (1987) "*The R+-tree: A Dynamic Index for Multidimensional Objects* ", Proceedings of VLDB, pp. 507-518
- [18] VCKOVSKI A. (1995) "*Representation of Continuous Fields*", Proceedings of 12th Auto Carto 12, Vol.4, pp.127-136.