SPATIAL INDEX BASED ON MULTIPLE INTERVAL SEGMENTS

Xiao Weiqi Feng Yucai
Computer Sci. and Eng.,
Huazhong University of Sci. and Tech.
Wuhan 430074, P.R.China Email: dbinst@hust.edu.cn

KEY WORDS

Spatial Index, Spatial Database, Interval segments.

ABSTRACT

This paper presents a spatial index mechanism that aims to improve the performance of spatial information retrieval. This kind of spatial index can be applied to retrieving objects in an arbitrary rectangular region rapidly. We describe this spatial index mechanism in detail, and provide fundamental retrieval and insertion algorithms. Finally, performance analyses are also presented.

1 INTRODUCTION

Modern database management systems have been widely used in many new applications, including: Geographic Information System (GIS)[Mich94], Cartography, Computer Aided Design/Manufacture [Ibra92], City Planning, Real-estate Managing, etc. Generally, a large number of spatial queries would occur in the above application frequently. So, how to access objects that satisfy a specific condition from a large scale spatial object database has become an important research direction[Xiao94].

The best way improving spatial search speed is to construct an efficient index mechanism suitable for spatial operations. In the past decade, several spatial access approaches have been presented. A survey can be found in [Lu93]. Quad-tree, R^+ tree[Seli87] and Ladex[Xiao96] are typical good spatial index structures. However, every one of them has its weakness in different aspects. In this paper, we present a spatial index structure based on multiple interval segments(SIMIS for short).

"Permission to make digital or hard copies of part or all 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. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee."
© 1997 ACM 0-89791-850-9 97 0002 3.50

2 SIMIS SPATIAL INDEX

All objects in an application can be contained in a rectangular region $[0, 0, X_{max}, Y_{max}]$, called *spatial domain*, where X_{max} is the maximal valid value in X-axis, and Y_{max} is the maximal legal value in Y-axis. Any independent spatial object can be bounded by a minimal bounding rectangle $Rect[x_1, y_1, x_2, y_2]$ that is called *effective region* of object e, denoted by $Rect(e)$.

An interval on the X-axis(Y-axis) is called *X-interval(Y-interval)*[Elma90]. Suppose an object's *effective region* is $Rect[x_1, y_1, x_2, y_2]$. Its interval on X-axis $[x_1, x_2]$ is called *effective X-interval*, represented by X(e). Similarly, its interval on Y-axis $[y_1, y_2]$ is called *effective Y-interval*, denoted by Y(e).

The spatial index mechanism we discuss here is based on object record storage system ODB. $ODB = \{e_1, e_2, \ldots, e_n\}$, where e_i is a spatial object record. The search operation is defined formally as follows.

Given a search region $I_s = [x_a, y_a, x_b, y_b]$, find the set of objects that are contained in or intersect with the region:

$$S(I_s) = \{e_i \in ODB \mid (X(e_i) \cap I_s) \neq \varnothing\}$$

Given a search X-interval $I_{sx} = [x_a, x_b]$, find the set of objects whose effective X-interval are contained in or meet with the search interval:

$$S_x(I_{sx}) = \{e_i \in ODB \mid (X(e_i) \cap I_{sx}) \neq \varnothing\}$$

Similarly, we have:

$$S_y(I_{sx}) = \{e_j \in ODB \quad |(X(e_j) \cap I_{sy}) \neq \varnothing\}$$

The major strategy of SIMIS is to transform 2-dimensional spatial data into 1-dimensional linear data, and then use *B⁺tree* techniques to build the index structure. The approach is to decompose the effective regions of all objects in the spatial domain into effective X-intervals and effective Y-intervals, and then draw out the starting point and the next point to the ending point of every interval as *index points*. So, we can use B^+ trees to index on these points on X-axis and Y-axis separately. X-interval and Y-interval index points can be extracted as follows separately:

$$BP_x = \{x_i | e_j \in ODB((x_i = X(e_j).x_1) \vee (x_i = X(e_j).x_2 + 1))\} \cup \{X_{max} + 1\}$$

$$BP_y = \{y_i | e_j \in ODB((y_i = Y(e_j).y_1) \vee (y_i = Y(e_j).y_2 + 1))\} \cup \{Y_{max} + 1\}$$

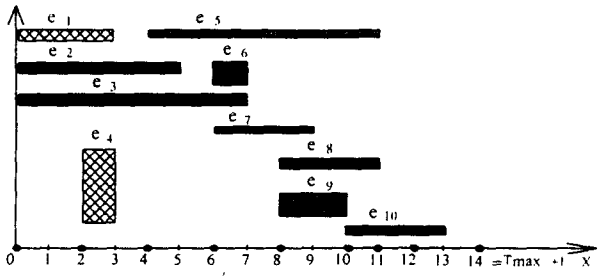

Figure 1 X-interval index points

Figure 1 is an example to show X-interval index points that are marked by big solid dot on X-axis.

Assume x_j is any value on X-axis. It may be in or not in BP_x. $x_j^-(x_j^+)$ is previous (successive) point of x_j in BP_x. $x_j^{-=}$ is the largest in BP_x among those are not larger than x_j.

In the extended B^+ tree, the leaf node has the following format $[x_i, buck]$, where *buck* is a pointer that links to a bucket of object identities(OIDs). Each bucket $B_x(x_i)$ includes all spatial objects intersect with X-interval $[x_i, x_i^+ - 1]$.

In some applications, the same object may be stored in several consecutive buckets in B^+ tree leaves. This kind of data redundancy can be partly solved by referring to incremental storage model[Elma90]. In each leaf node, the leading bucket holds all of the objects that belongs to it, and in the successive buckets, we only store objects that enter or leave away, relative to previous bucket.

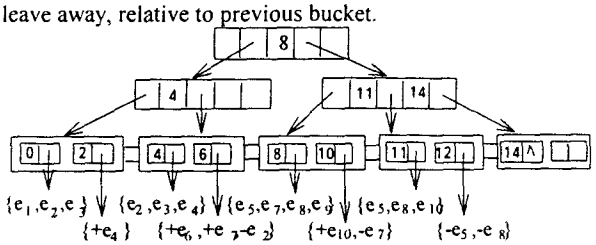

Figure 2: The Incremental Storage Model

Figure 2 shows the storage model by using the example in Figure 1. The non-leading bucket $Bx(x_i)$ at point x_j in leaf nodes can be computed by:

$$B_x(x_i) = B_x(x_h) \cup (\cup_{x_j \in BPx,\ x_h < x_j < x_i} SA_x(x_j)) - (\cup_{x_j \in BPx,\ x_h < x_j < x_i} SE_x(x_j))$$

where $B_x(x_h)$ is the leading bucket that x_i belongs to, $SA_x(x_j)$ is a set of objects whose starting point of effective X-interval is x_j. $SE_x(x_j)$ is a set of objects whose ending point is $x_j - 1$. As to Y-interval, similarly, we can get:

$$B_y(y_i) = B_y(y_h) \cup (\cup_{y_j \in BPy,\ y_h < y_j < y_i} SA_y(y_j)) - (\cup_{y_j \in BPy,\ y_h < y_j < y_i} SE_y(y_j))$$

3 SEARCH AND MAINTENANCE

X-interval Search Algorithm

(1) Assume the X-interval to be $I_{sx} = [x_a, x_b]$, find all of the index points related to I_{sx} on the B^+ tree.

$$PI_x(I_{sx}) = \{x_i \in BP_x | x_a \leq x_i \leq x_b\} \cup \{x_a^{-=}\}$$

(2) Compute the following set as the result of this algorithm.

$$T_x(I_{sx}) = \cup_{x_i \in PI_x} B_x(x_i)$$

Y-interval Search Algorithm

Similarly to the above X-interval search algorithm, given a Y-interval $I_{sy} = [y_a, y_b]$, the result can be obtained through the following steps:

(1) $PI_y(I_{sy}) = \{y_i \in BP_y | y_a \leq y_i \leq y_b\} \cup \{y_a^{-=}\}$

(2) $T_y(I_{sy}) = \cup_{y_i \in PI_y} B_y(y_i)$

Region Search Algorithm

In respect to regional search, suppose the query region is $I_s = [x_a, y_a, x_b, y_b]$. The search result can be computed by: $T(I_s) = T_x(I_{sx}) \cap T_y(I_{sy})$.

The correctness of search algorithms can be guaranteed by Theorem 1, Theorem 2 and Theorem 3 separately.

Theorem 1: $S_x(I_{sx}) \equiv T_x(I_{sx})$

Theorem 2: $S_y(I_{sy}) \equiv T_y(I_{sy})$

Theorem 3: $S(I_x) \equiv T(I_x)$

4 FURTHER EXTENSION

Actually, we can create Y-interval indexes(X-interval indexes) again based on previously built X-interval index(Y-interval index). Thus the *interval index hierarchy* is constructed. Suppose $f_{I_{sy}}(B_x)$ is a function to get all objects that satisfy the condition I_{sy} through the second level Y-interval index in the first level X-interval bucket B_x. We can also assume $f_{I_{sx}}(B_y)$ is a function collect all objects that satisfy the condition I_{sx} through the second level X-interval index in the first level Y-interval bucket B_y. $SA_x(x_j)(I_{sy})$ is a function to get the objects that satisfy the condition I_{sy} through the second level Y-interval index in a first level bucket that x_j belongs to in the set SA_x.

The definitions of $SE_x(x_j)(I_{sy})$, $SA_y(y_j)(I_{sx})$ and $SE_y(y_j)(I_{sx})$ are similar to that of $SA_x(x_j)(I_{sy})$.

The search results can be computed as follow:

$$T(I_x) = \bigcup_{x_i \in PI_x} f_{I_{sy}}(B_x(x_i)) = \bigcup_{y_i \in PI_y} f_{I_{sx}}(B_y(y_i))$$

where functions $f_{I_{sy}}(B_x(x_i))$ and $f_{I_{sx}}(B_y(y_i))$ can be expressed in detail like this:

$$f_{I_{sy}}(B_x(x_i)) = B_x(x_i)(I_{sy}) = B_x(x_h)(I_{sy}) + (\bigcup_{x_j \in BP,\ x_h \le x_j \le x_i} SA_x(x_j)(I_{sy})) - (\bigcup_{x_j \in BP,\ x_h \le x_j \le x_i} SE_x(x_j)(I_{sy}))$$

$$f_{I_{sx}}(B_y(y_i)) = B_y(y_i)(I_{sx}) = B_y(y_h)(I_{sx}) + (\bigcup_{y_j \in BP,\ y_h \le y_j \le y_i} SA_y(y_j)(I_{sx})) - (\bigcup_{y_j \in BP,\ y_h \le y_j \le y_i} SE_y(y_j)(I_{sx}))$$

5 PERFORMANCE ANALYSES

SIMIS mechanism transfers indexing on 2-dimensional data into indexing on 1-dimensional data, and uses the extended B^+ trees as its basic components. Therefore, it almost has the same high performance as that of the conventional B^+ tree. We have worked out a wide variety of performance evaluation strategies to compare SIMIS with other typical spatial indexes, such as Quad-tree and R^+ tree. The results show that, if small region or interval objects dominate in the spatial domain and the search gap(region/X-interval/Y-interval) is relatively small, SIMIS outperforms the other two methods; otherwise SIMIS's performance degenerates as the other two approaches do.

6 CONCLUSIONS

In this paper, we described a spatial indexing mechanism, the SIMIS, for spatial data. Currently, SIMIS index mechanism has been built in a distributed multimedia geographic database management system DM2, which was developed by our research group during 1992 to 1995. Now, a lot of spatial data related applications have been developed on DM2 successfully.

REFERENCES

[Elma90] R.Elmasri, G.T.J. Wuu,and Y.J. Kim., The Time Index: An Access Structure for temporal data. In 16th VLDB, pages 1-12, Aug. 1990.

[Ibra92] Ibrahim Kamel and Christos Faloutsos, Parallel R-trees, ACM SIGMOD 6/92/CA, USA. pp. 195-204.

[Lu93] Lu, H., and Ooi,B.C.,Spatial Index: Past and Future. IEEE Data Engineering, 16(3), pp. 16-21, 1993.

[Mich94] Michael F. Worboys, A unified model for spatial and temporal information, The Computer Journal, Vol.37. No.1, 26-34, 1994.

[Seli87] Sellis T. et al. The R^+ tree: A Dynamic Index for Multi-Dimensional Objects. In Proc. of the 13rd VLDB Conference, pp. 507-513, 1987.

[Xiao94] Xiao Weiqi, FengYucai, Yang Xiaoan, The Design and Implementation of Language MQL for the Map Database, Journal of Huazhong University of Science and Technology,Vol. 22, No.6, 51-55,1994.

[Xiao96] Xiao Weiqi, Feng Yucai, Ladex: A New Index Mechanism for Spatial Database System, International Archives of Photogrammetry and Remote Sensing. Vol. XXXI, Part B3, 930-935, Vienna, 1996.