Paper Title (use style: *paper title*) SR-Tree: An Index Structure of Sensor Management System for Spatial Approximate Query

Xing Tong, Yang Liu, Zhao Shi, Peng Zeng, Haibin Yu
Department of Industrial Control Network and System
Shenyang Institute of Automation Chinese Academy of Sciences
Shenyang, Liaoning 110016, China
E-mail: tongxing@sia.cn

***Abstract*—Sensor management plays an important role in the field of Internet of Things. Therefore, the requests of spatial approximate query increase dramatically. Indexing is no doubt a feasible way for efficient spatial approximate search. However, there is a lack of an effective index structure for spatial approximate query. In this paper, we propose a new type of index structure called SR-tree for providing more intelligent retrieval, which is based on R-tree and inverted table. Our index can support for spatial approximate search and work freely either in memory or external memory. The experimental results show that the structure proposed can provide high scalability and fast response time.**

Keywords-sensor management system; index structure; Internet of Things.

I. INTRODUCTION

The system of management and processing sensor data is called sensor management system[1]. In IT systems, the users generally first target some interested sensors through sensor management system, then get the sensor data to analysis and practice. Thus, the sensor retrieval has an important function in sensor management system.

One hand, with the increase of the number and variety of sensors, it is more difficult to locate a sensor in the system. On the other hand, the query criteria can be varied for meeting users while retrieving. In other words, the system returns to the users the available sensor normally. But if it could not completely match the users' demand, similar sensors set is returned to users. Because of these two reasons, it is urgent to build an index structure for sensor management system both supporting the large amount of data and approximate queries.

In the retrieval process, we identify sensors through their characters. First, since the position information is an important factor to recognize sensors, we may make a geographical division for sensors using R-tree. Second, we use the inverted index table structure on other properties of sensors, such as a wireless communication protocol, energy consumption, perceived function, data upload frequency, etc. We call this kind of queries spatial approximate queries related to sensor location and other properties. Ultimately, we build our index structure SR-tree, which combines R-tree and inverted list and supports the realization of spatial approximate query.

The contributions of this paper include:

1 We give a definition of spatial approximate query and present a sensor management system indexing structure that supports spatial approximate query.

2 An efficient search algorithm based on SR-tree is proposed.

3 Extensive experimental on simulated datasets results show that our indexing scheme achieves favorable performance.

The rest of the paper is organized as follows. Section II discusses the necessary background and gives a formal problem definition. Section III presents the basic principles of the SR-tree. Section IV presents details of

search algorithm. Section V presents a comprehensive evaluation of the proposed techniques and Section VI concludes the paper.

II. PRELIMINARIES

In this section, first we discusses the necessary background of R-tree and inverted table and then we give the formal definition of SAQ.

A. R-tree

R-tree structure is commonly used in the geographic information system. The R-tree [2] and its variants (R□-tree in particular [3]) share a similar principle. They first group points that are in spatial proximity with each other into a minimum bounding rectangle (MBR); these points will be stored in a leaf node. The process is repeated until all points in P are assigned into MBRs and the leaf level of the tree is completed. The resulting leaf node MBRs are then further grouped together recursively till there is only one MBR left. Each node in the R-tree is associated with the MBR enclosing all the points stored in its sub-tree. Each internal node also stores the MBRs of all its children. An example of an R-tree is illustrated in Figure 1 where pi denotes one point and Nj represents the area.

B. Inverted Table

In computer science, an inverted index (also referred to as postings file or inverted file) is an index data structure storing a mapping from keywords, such as words or numbers, to its locations in a database file, or in a document or a set of documents. It is the most popular data structure used in document retrieval systems, [4] used on a large scale for example in search engines.

In sensor management system, we use the properties of the sensor as a keyword, use the ID number as the "location" to build the inverted table. We show a simple example in Table 1. In the original table, the first column implies the sensor ID and the second column contains some attributes of each sensor, here represented by a-g. We use the contents of the table to construct a simple inverted table as follows.

TABLE I. AN EXAMPLE OF INVERTED TABLE

Sensorid	Sensor property
1	a,b,c
2	a,b,d,e
3	b,c,f,g
4	b
5	a,b,c,f,g

Keywords	a	b	c	d	e	f	g
Locations	1 2	1 2 3 4 5	1 3 5	2	2	3 5	3 5

C. Problem Formulation

Generally, users give the query conditions including the location of the sensor and attribute information and sensor management system traverses SR-tree to get a result set meeting the query conditions.

We call this kind of query spatial approximate query (SAQ) which consists of three parts: the spatial query Q_r, the sensor property query Q_p and a query threshold T. A range query Q_r is simply defined by a query rectangle r; The sensor property query is defined by one or more query sensor properties p; And our system predefined a threshold which implies the least number of public properties between result set and Q_p. i.e., SAQ = ($Q_r=r$; $Q_p=(p_1,p_2,p_3\ldots)$; T).

Now we define the result set. First of all, we define a sensor $S(L, P)$ where L denotes the location of the sensor and P denotes the collection of properties about sensors. Formally, the result set for SAQ is $\{S(L,P)|L \in Q_r \wedge P \cap Q_p \geq T\}$. In other words, sensors in result set must be in a certain area and contain T properties intersecting with Q_p at least. SAQ is different from queries respect to spatial databases [5,6,7], because what we search for is the property of sensors rather than a string.

III. Proposed Index Structure

In this section, we give the details of SR-tree structure, which is constituted by r-tree and the inverted table. The formal definition of SR-tree is as below:

DEFINITION (SR-tree)

A SR-tree is a tree structure which contains leaf nodes and intermediate nodes. Each leaf node is represented as a double tuple(R, IT) where R is a spatial scope including space of lower nodes the same as r-tree and IT is an inverted table constructed by the attributes of all the sensors in the geographic range R. Each intermediate node is a double node(R, V), where V is a collection of sensor properties locating in R(We use the vector to store V in our system).

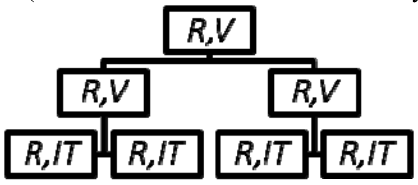


Figure 2 The SR-tree structure.

The Figure 2 above shows a simple SR-tree structure for discussing and SR-tree in applications can be multi-level, multi-branched and unbalanced. In this kind of index structure, it is convenient to do SAQ. We can filter the location and attribute information in internal nodes at the same time. Afterwards, through the inverted list we specific sensors for applications in leaf nodes. In next section, we will give the details of search algorithm.

IV. Search Algorithm Based On SR-Tree

This section presents a search algorithm supported by SR-tree where the input is the query terms SAQ and the root node N of the index structure, the output is a result set RS satisfying SAQ. Search algorithm recursively traverses the entire index structure that can determine whether the sub-tree contains the query results on internal node, and continue to traverse his child nodes if possible. The pseudo-code of search algorithm is as follows:

```
Search_BasedSR-tree(Query SAQ, Tree node N, Result set RS)
    Input: SAQ = ( $Q_r$; $Q_p$; $T$) and root node N

    Output: Result set RS

1: if N is the leaf node then
2:   Candidate set CS=N.IT.Find($Q_p$, $T$)
3:   for each $S \in CS$ do
4:     if $S.L \in Q_r$ then
5:       RS.add(S)
6: else
7:   if $N.R \cap Q_r \neq \emptyset$ && $N.V \cap Q_p \geq T$ then
8:     for each child $C \in N$ do
9:       Search_BasedSR-tree(SAQ,C,RS)
```

During traversing internal nodes, if the intersection between geographical scope of nodes and Q_r is not empty, as well as the intersection of properties contained in the node and Q_p meeting the threshold(Line 7), then the sub-tree rooted at this node may contain query results, thus recursively visiting its child nodes (Line8-9). When processing leaf nodes, the program calls the inverted table query algorithm Find(Q_p, T) (This algorithm has been studied extensively in information retrieval field beyond the scope of article discussion and we use the classic algorithm DivideSkip[8] in our system.) which first finds a candidate set that matches Q_p (Line 2), and then detects Q_r, ultimately adds sensors meeting conditions to the result set (Line 3-5).

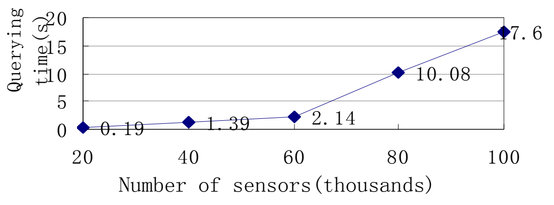
V. Experimental Evaluation

In this section we evaluate the performance of the query respect to SAQ on a simulated dataset.

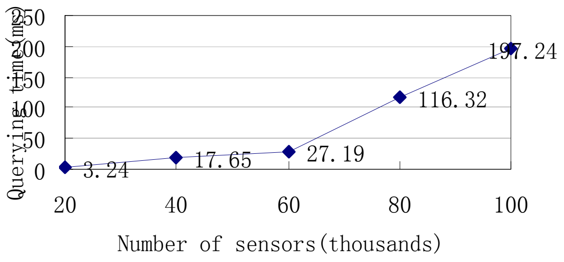
We simulated a sensor data collection which contains 100 thousands sensors. Each sensor is constituted by randomly location information represented by the two-dimensional coordinates in the entire area [(0, 0), (100,100)] and a collection of properties choosing 10-20 items from a total of 100 properties from all sensors. All of data was stored in the file system and we took 100 attribute as keywords in inversed table, the offset of each sensor data in the file system as the id number of sensors.

We compile all the programs in Windows7 using jdk 6.0 with java. The experiments are run on a Intel(R)Core(TM) i5-2400 CPU @3.10GHz with 4 GB main memory.

As far as we know, we are the first to study the index structure of sensor management system for SAQ queries. We give the query time with the increase of the amount of data, as shown in Figure 3. Since the large amount of data is limited by memory capacity, the index structure can be built in memory or on disk. Thus, we had test queries in two setting, one is the index is located in disk while the other is the index is in main memory. In the first type of experiments, we measured the performance of queries when all data required for answering a query needed to be retrieved from disk. In the second setting, the experiment represents the other extreme in which all data required to answer a query is already in memory.



(a)Raw disk



(b) Fully memory

Figure 3 The scalability of SAQ.

The results show SR-tree offers favorable scalability either on raw disk or in fully memory. Because we used filter conditions Q_r and Q_p simultaneously in high level of the tree, we could filter more leaf nodes unsatisfied with the query even though large amount of data.

VI. Conclusion

In this paper we propose a general tree-based index structure to support spatial approximate queries with respect to a threshold. We merge R-tree and inverted list to build our index structure SR-tree, which supports the realization of an effective search algorithm. The experimental results on real datasets show our indexing scheme achieves comparable performance. Furthermore, our index scheme can be easily implemented in existing commercial sensor management systems with existing tree-structured index and small available memory.

Acknowledgement

The authors acknowledge the financial support of the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No.XDA06020302, the Important National Science and Technology Specific Project under Contact No.2010ZX03006-005-01, the National High Technology Research and Development Program of China under 863 Program No.2011AA040103.

References

[1] Li JianZhong, Li JinBao, and Shi ShengFei, "Concepts, Issues and Advance of Sensor Networks and Data Management of Sensor Networks," Journal of Software, vol. 14, 2003, pp. 1717-1727.

[2] A. Guttman, "R-trees: a dynamic index structure for spatial searching," Proc. Special Interest Group On Management Of Data, 1984.

[3] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," Proc. Special Interest Group On Management Of Data, 1990.

[4] Zobel, Justin, Moffat, etc, "Inverted files versus signature files for text indexing," ACM Transactions on Database Systems (TODS), vol. 23, 1998, pp. 453-490.

[5] Sattam Alsubaiee, Alexander Behm, and Chen Li, "Supporting Location-Based Approximate-Keyword Queries," Proc. The 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM Press, 2010, pp. 61-70.

[6] Bin Yao, Feifei Li, Marios Hadjieleftheriou, and Kun Hou, "Approximate string search in spatial databases," Proc. The 26th International Conference on Data Engineering, IEEE Press, 2010, pp. 545-556.

[7] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe, "Keyword Search on Spatial Databases," Proc. The 24th International Conference on Data Engineering, IEEE Press, 2008, pp. 656-665.

[8] Chen Li, Jiaheng Lu, and Yiming Lu, "Efficient merging and filtering algorithms for approximate string searches," Proc. The 24th International Conference on Data Engineering, IEEE Press, 2008, pp. 257-266.

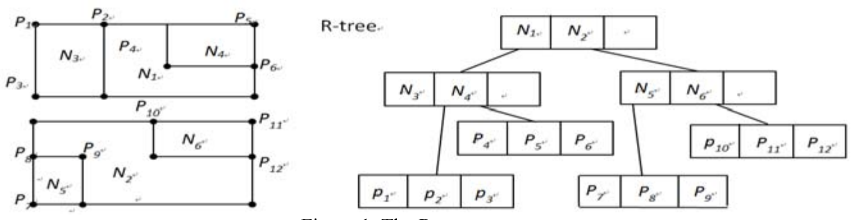


Figure 1 The R-tree structure.