Spatial Hierarchy and OLAP-Favored Search in Spatial Data Warehouse

Fangyan Rao, Long Zhang, Xiu Lan Yu, Ying Li, Ying Chen
IBM China Research Laboratory, Beijing, 10085, China
{raofy,longzh,yuxl,lying,yingch}@cn.ibm.com

ABSTRACT

Data warehouse and Online Analytical Processing(OLAP) play a key role in business intelligent systems. With the increasing amount of spatial data stored in business database, how to utilize these spatial information to get insight into business data from the geo-spatial point of view is becoming an important issue of data warehouse and OLAP. However, traditional data warehouse and OLAP tools can not fully exploit spatial data in coordinates because multi-dimensional spatial data does not have implicit or explicit concept hierarchy to compute pre-aggregation and materialization in data warehouse. In this paper we extend the traditional set-grouping hierarchy into multi-dimensional data space and propose to use spatial index tree as the hierarchy on spatial dimension. With spatial hierarchy, spatial data warehouse can be built accordingly. Our approach preserve the star schema in data warehouse while building the hierarchy on spatial dimension, and can be easily integrated into existing data warehouse and OLAP systems. To process spatial OLAP query in spatial data warehouse, we propose an OLAP-favored search method which can utilize the pre-aggregation result in spatial data warehouse to improve the performance of spatial OLAP queries. For generality, the algorithm is developed based on Generalized Index Searching Tree(GiST). To improve the performance of OLAP-favored search, we further introduce a heuristic search method which can provide an approximate answer to spatial OLAP query. Experiment result shows the efficiency of our method.

Categories and Subject Descriptors

E.m [**Miscellaneous**]:

General Terms

Algorithms, Performance

Keywords

Spatial Hierarchy, Spatial OLAP, Spatial Data Warehouse

DOLAP'03, November 7, 2003, New Orleans, Louisiana, USA.

1. INTRODUCTION

Spatial data exists pervasively in business information systems. It is claimed that 80% of the overall information stored in computers is geo-spatial related, either explicitly or implicitly [6]. With the advent of mobile computing and location-based services, even more amount of spatial data is collected and stored in database. While storing spatial data is the first step, it is more important to analyze spatial data in business intelligent systems to provide insight into business from the geo-spatial point of view. For example, when choosing site when opening a store, spatial related information like distance to the nearest highway, income of neighborhood residents, and/or daily traffic volume on the nearby roads should be taken into consideration.

Data warehouse is "a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process." [15]. The purpose of data warehouse is to support OLAP, which is especially designed for knowledge workers (executives, managers, analysts) to systematically organize, understand, and use their data to make strategic decisions. In data warehouse and OLAP, business data is viewed from multiple dimensions and modeled as multidimensional data cubes defined by dimensions and facts [10]. Each dimension represents some business perspective, like customers, products and time, and facts are numerical measures to be analyzed, such as sales in dollars. Cell in data cube is referred as cubiod. To provide user with flexible views of data from different perspectives on multiple summarization levels, concept hierarchy is defined on dimension. OLAP materializes these views to improve the performance of OLAP query processing. Materializing, or pre-aggregation, computes aggregate measure in cubiods.

While taking spatial data into consideration, data warehouse and OLAP can provide insight into data from the geo-spatial point of view. Location data in databases is often described in text strings, like street address, and has implicit hierarchy like (street $<$ city $<$ state or province $<$ nation). This location string is not what we called multidimensional spatial data, and has limited capacity in spatial analysis. For example, query like "what's the total sales of gas stations less than 10 km from the mall" can not be processed easily when the mall's location is represented in street address. Geocoding technique translates street address into two-dimensional point data. Some commercial database vendors, such as IBM and Oracle, provide spatial extensions in their database systems [19][1] to store spatial data and has certain level of spatial analysis capabilities.

We refer spatial data as multi-dimensional spatial objects described in coordinates. Traditional data warehouse and OLAP tools can not fully exploit spatial data because spatial data does not have implicit or explicit concept hierarchy. While trying to address the

problem under the framework and data model of traditional data warehouse and OLAP, we adopt Han's definition of spatial-to-spatial dimension as spatial dimension, whose primitive level and all of its high level generalized are spatial [13]. The spatial dimension differentiates from the non-spatial dimension in that there is little a priori knowledge about the concept hierarchy on the dimension[17][18]. Without concept hierarchy, data warehouse can not pre-aggregate on spatial dimension.

In this paper we extend the traditional set-grouping hierarchy into multi-dimensional data space and propose to use spatial index tree as the spatial hierarchy. With this approach, concept hierarchy on spatial dimension can be automatically generated. Based on spatial hierarchy, data cube and pre-aggregation can be computed in spatial data warehouse while preserving the star schema in data warehouse. To improve the efficiency of spatial OLAP query processing, we propose an OLAP-favored search method which utilizes the pre-aggregation result when processing spatial OLAP queries in spatial data warehouse. For generality, the algorithm is developed based on Generalized Index Searching Tree(GiST) proposed in [14]. To further improve the query performance, we propose a heuristic OLAP-favored tree searching method which can generate an approximate answer with controlled error. Experiment conducted on three different data sets shows the efficiency of our method.

The rest of the paper is organized as follows. Motivating example is introduced in Section 2. Section 3 details the method of generating the hierarchy on spatial dimension. Section 4 outlines the OLAP favored query processing algorithm based on GiST. The related work is covered in section 5. Section 6 concludes our work and points out future works.

2. MOTIVATING EXAMPLE

Star schema is the most common modeling paradigm in data warehouse [12]. Fig. 1 illustrates the star schema of an example data warehouse with sample data. It's a gas sales data warehouse to analyze the gas sales concerning different gas stations, gas types, customer's vehicles and time.

Station

Station_ID	Station_Name	Address	Location
77	Beiti Gas Station	2th Street, Shangdi, Beijing	(116.37,39.51)

Gas

Gas_ID	Unit_Price
90	3.25
93	3.60

Sales

Trans_No.	Station_ID	Gas_ID	Vehicle_Group_ID	Time_ID	Quantity	Sales
5478912	77	90	2	3455	40	130.00

Vehicle

Vehicle_Group_ID	Type
1	Private car
2	Taxi
3	Bus

Time

Time_ID	Time	Day	Month	Year
3455	7:20	12	Feb	2002
3456	7:25	13	Feb	2002

Figure 1: Star Schema in Data Warehouse

In the data warehouse there are four dimension tables: *Station, Gas, Vehicle and Time*. The Station table is the spatial dimension in the data warehouse. Its attribute *location* gives the spatial location of a gas station. The fact table is *Sales* which stores the sales data of each transaction.

Users can analyze the data from multiple perspectives including the spatial perspective. A spatial-related OLAP query may be "what's the total sales of all gas stations within the given rectangular region in January, 2003?" We refer to the query as Q1 hereafter in this paper. The gas sales data warehouse and the dynamic spatial-related OLAP query in particular are employed as our motivating example.

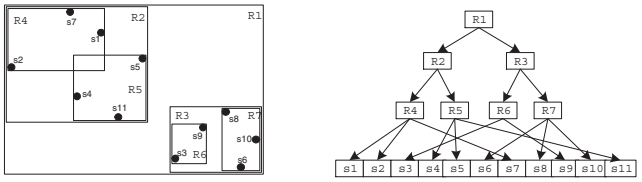


Figure 2: Spatial Data and Rtree

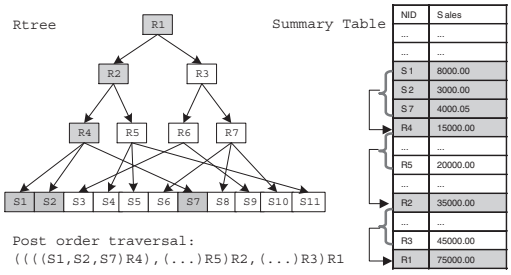


Figure 3: R-tree, Post order traversal and Aggregation path

3. HIERARCHY OF SPATIAL DIMENSION

We introduce the term *spatial index tree* to denote the spatial indexing method that organize spatial data in hierarchies. Various spatial index trees have been surveyed [8], including Quadtree [7], K-D-B-tree [20], R-tree [11], its variation R*-tree [3] and R+-tree [21], etc. One characteristic of the spatial index tree is that a tree node spatially encloses its children nodes. Taking R-tree as an example, the spatial data set (points representing gas stations) and the associated Rtree is shown in Fig. 2. Based on this characteristic, we propose using spatial index tree as the multidimensional set-grouping hierarchy on spatial dimension because it provides a grouping schema in multi-dimensional space.

Unlike the traditional hierarchies, the spatial hierarchy can be very complex. Post order traversal on the index tree generates all the aggregation paths from the bottom to the top of the tree. The relationship between R-tree, its post order traversal and aggregation path is depicted in Fig. 3.

Taken spatial index tree as the set-grouping hierarchy, data warehouse can build data cubes following aggregation pathes derived from the index tree. A sample data cube based on spatial hierarchy in Fig. 3 is shown in Table 1. The *Group By Cube*, an OLAP extensions to SQL provided by DB2, is employed here to aggregate all non-spatial dimensions[5].

4. OLAP-FAVORED QUERY PROCESSING

4.1 OLAP-favored Query Processing Overview

Due to the complexity of spatial hierarchy, the query processing in spatial OLAP depends on spatial index to find out what cubiod in the data warehouse need to be exploited. For example, to answer query Q1 where the region is defined ad-hoc, OLAP query processing engine needs to first check the spatial index on the location dimension to get all the intermediate aggregated nodes ID. With those IDs, associated cubiods in the summary table can be located, and conditions on other dimensions will be further checked.

Table 1: Sample content of OLAP data cube

NID	Vehicle	Gas	Year	Month	Sales
...	...	...	...	...	...
R4	1	90	2002	Jan.	27865.00
R4	1	90	2002	Feb.	24500.00
R4	1	90	2002	...	...
R4	1	90	2002	-	190234.00
R4	1	90	-	-	348976.00
R4	1	93	2002	Jan.	3456.00
R4	1	93	...	...	...
R4	1	93	-	-	1098768.00
R4	1	...	...	...	...
R4	1	-	-	-	2310087.00
R4	2	...	...	...	...
R4	-	-	-	-	45619876.00
...	...	...	...	...	...
R2	1	90	2002	Nov.	3017.00
R2	1	90	2002	-	455231.00
R2	1	90	null	-	1765234.00
R2	1	-	-	-	28754395.00
R2	-	-	-	-	63549857.00
...	...	...	...	...	...

Thus spatial index tree is not only needed to build up hierarchy in data warehouse, but also plays a key role in spatial-related OLAP query processing. Traditional tree searching method returns all the tuples satisfying the query. However, it can not exploit the advantage of pre-aggregation in spatial data warehouse. In order to better utilize pre-aggregtion, we modify the search algorithm to return the intermediate tree node if all the descendant tuples indexed by the node satisfies the query. The intermediate tree node refers to a cubiod in the data cube. By using pre-aggregate in cubiod instead of raw data in fact table, the performance of OLAP query processing can be greatly improved. We call the modified search method the OLAP-favored search.

In the case of the query Q1, we need not to retrieve every single gas station within the query region. In R-tree, each node has a minimum bounding rectangle(MBR) that encompasses all the children nodes' MBR. In the case of spatial predicate "within", if a tree node's MBR are within a certain query region, all the descendant spatial objects rooted from the node are within the query region. Thus, we can exploit pre-aggregated measures stored in data cube by searching the R-tree in a top-down way from the root and finding all the tree nodes whose MBR are within the query region, and the gas stations satisfying the query predicate, but whose parent node is not fully within the query region. Each intermediate tree node's aggregation has been computed and the aggregation of measure is stored in the data cube.

Compared with traditional search method, OLAP-favored search method returns less number of tuples and visits less number of "edges" in the tree. Fig. 4 illustrates the advantages of OLAP-favored search against traditional search.

4.2 OLAP-favored Search On GiST

Generalized Index Search Tree(GiST) [14] provides a flexible index mechanism that both data and access method can be extended. Both one-dimensional index like B-tree and multi-dimensional index like R-tree can be implemented under the framework of GiST. For generality, we develop the OLAP-favored search algorithm based on GiST's index framework. GiST provides the capability to extend by introducing two interfaces that need to be implemented by

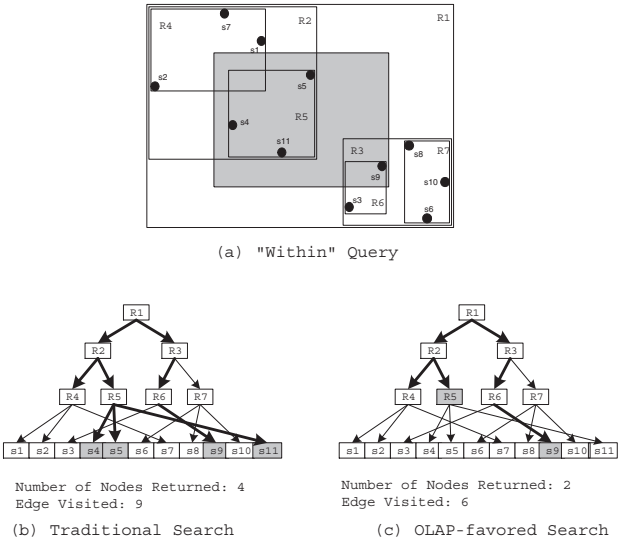


Figure 4: Traditional search and OLAP-favored search

Table 2: Consistent(E,q)

Consistent(E,q)	
Input:	Index entry E = (p; ptr), query predicate q where p is the predicate of the index entry, ptr is the pointer to the child node in the index entry.
Output:	returns false if for each descendant entry(DE) of E, $DE.p \wedge q$ can be guaranteed unsatisfiable, true if for each descendant entry(DE) of E, $DE.p \wedge q$ can be guaranteed satisfiable, partial true otherwise.

index implementors: *Predicate* interface and *Gist* interface. Predicate interface defines an important method *consistent* that evaluate the consistency between query predicate and tree node predicate. The search algorithm in GiST invoke this method to get all the leaf nodes that are consistent with the query predicate.

We enhanced GiST's *predicate* interface by introducing a new predicate consistent state *Partial True*. Table 2 gives the detail definition of the new interface.

For example, in Fig. 4, tree node R3 is not fully within the query region, but we can not guarantee that all its descendant leaf nodes are outside the given region. Actually s9 is within the query region. So for the within predicate on R3, Consistent should return partial true.

For R-tree's implementation of Consistent, Table 3 depicts the spatial predicate of *within*.

With the third consistent state, we improve GiST's query algorithm to favor OLAP aggregation queries. Table 4 outlines the improved search algorithm.

In general, while an intermediate tree node satisfies the query predicate in OLAP-favored search, searching stops traversing the tree and returns the intermediate tree node instead of all its descendant leaf nodes.

4.3 Heuristic OLAP-favored Search

If the partial true consistent state can be quantified and the searching algorithm can make some heuristic estimation on the final result of Q1, the performance of OLAP-favored search can be further

Table 4: OLAP-favored Search Algorithm

Search(R, q)	
Input:	GiST rooted at node R, search predicate q
Output:	all nodes/tuples that satisfies q
Sketch:	Recursively descend all paths in tree whose keys are partially consistent with q
Procedure:	List ret = null
	//1. [Search subtrees]
	IF R is not a leaf THEN
	FOR Each entry E on R
	BEGIN
	IF Consistent(E,q) is true THEN
	add E to ret
	ELSE
	IF Consistent(E,q) is partial true THEN
	// invoke Search on the subtree whose root node is referenced by E.ptr
	List subret = Search(E.ptr, q)
	add subret to ret
	END
	//2. [Search leaf node]
	ELSE // R is a leaf
	FOR Each entry E on R
	BEGIN
	IF Consistent(E,q) is true THEN
	add E to ret
	END
	RETURN ret

Table 3: R-tree's Implementation of Consistent

Consistent(E,q)	
Input:	Index entry E=(p; ptr), where p has the MBR of the node, ptr is the pointer to node of the query entry and the query predicate q has query region's MBR.
Output:	true, false, or partial true

Procedure:	IF q.Predicate == Within THEN
	IF q.MBR is contained by p.MBR THEN
	RETURN true
	ELSE
	IF q.MBR is intersected with p.MBR THEN
	RETURN partial true
	ELSE
	RETURN false

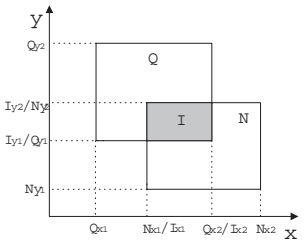


Figure 5: MBR, Query Window and Intersected Region

improved. We introduce an approximate factor(AF) to each tree node. AF is a value between 0 and 1 indicating the ratio of the approximate query result to the actual aggregation value of the node's subtree when the node's predicate partially satisfies the query predicate. To control the error generated by approximate search, we introduce Minimal AF threshold(MinAF). The searching method will traversal the tree top-down until AF is larger than MinAF. The AF value can be applied on the aggregation of the subtree rooted from the node and generate an approximate aggregation.

AF can be computed based on the node's MBR and the distribution of underlying spatial data. Here we take uniform distribution as the example to generate an AF evaluation method. Other sophisticated distribution may generate complex AF computation method, which in turn produces more exact AF values if it can better model the underlying data distribution.

Node's MBR, N, can be described by a quartuple: (N_{x1},N_{x2}, N_{y1},N_{y2}), where $N_{x1} < N_{x2}, N_{y1} < N_{y2}$. (N_{x1}, N_{x2}) and (N_{y1}, N_{y2}) give the range that the MBR extends in x and y dimension respectively. Similarly, query window, Q, can be described by another quartuple: $(Q_{x1}, Q_{x2}, Q_{y1}, Q_{y2})$. The intersected region, I, can be described by: $(I_{x1}, I_{x2}, I_{y1}, I_{y2})$, where $I_{x1} = min(N_{x1}, Q_{x1})$, $I_{x2} = max(N_{x2}, Q_{x2})$, $I_{y1} = min(N_{y1}, Q_{y1})$, $I_{y2} = max(N_{y2}, Q_{y2})$. The relationship between N, Q and I is depicted in Fig. 5.

Assume $f(x, y)$ is the distribution of spatial objects over two dimensional space. Generally AF can be computed by the following equation:

$$AF = \frac{\int_{I_{y1}}^{I_{y2}} \int_{I_{x1}}^{I_{x2}} f(x,y) d_x d_y}{\int_{N_{y1}}^{N_{y2}} \int_{N_{x1}}^{N_{x2}} f(x,y) d_x d_y} \quad (1)$$

If the spatial data follows uniform distribution, AF is ratio of the area of I to the area of N.

$$AF = \frac{(I_{x2} - I_{x1})(I_{y2} - I_{y1})}{(N_{x2} - N_{x1})(N_{y2} - N_{y1})} \quad (2)$$

With AF, interface Consistent can be defined as Table 5.

Consequently, search can give approximate result with AF. While

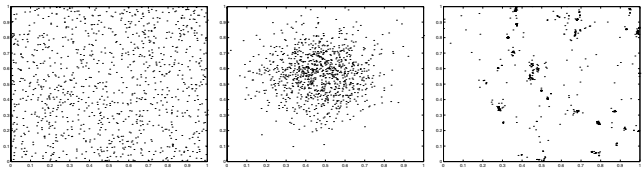
Table 5: Consistent(E,q)

Consistent(E,q)	
Input:	Index entry E = (p; ptr), query predicate q where p is the predicate of the index entry, ptr is the pointer to the child node in the index entry.
Output:	return 0 if for each descendant entry(DE) of E, $DE.p \wedge q$ can be guaranteed unsatisfiable, return 1 if for each descendant entry(DE) of E, $DE.p \wedge q$ can be guaranteed satisfiable, return AF($0 < AF < 1$) otherwise.

searching against the Rtree, heuristic search calculates AF when certain tree node partially satisfies the search predicate. Heuristic search will stop traversing the tree till AF at on node is above a predefined minimum threshold(MinAF). The approximate search result can be calculated by first multiplying the summerized number by AF of each node respectively, and then summarizing the total multiplied results. Because of limited space, we do not detail the heuristic search algorithm in this paper.

5. EXPERIMENTS

To evaluate the efficiency of proposed method, we conduct experiments on two simulated point data sets and one real point data set. The two simulated data sets are generated following uniform and normal distribution respectively. We simulate 1000 points spreading over the space (0, 0) and (1, 1). The simulated point's coordinates are generated following uniform or normal distribution independently. Real data set is retrieved from 2002 TIGER/Line file published by U.S. Census Bureau[23]. We use the point landmark feature in a county in California as our real data set. The number of points in real data set is 504. The data sets are depicted in Fig. 6. Sales data is simulated following normal distribution.



(a) Uniformly Dist. Data (b) Normally Dist. Data (c) Real Data

Figure 6: Testing Data

9 groups of queries are generated with 100 query regions in each group. Query regions in each group are same in size while the group region size grows from 10% to 90% of the whole data area. Experiment metrics of each group is given as the average metrics of all queries in the group. Query region's central location is generated following the normal distribution.

In Fig. 7, we compare OLAP-favored search algorithm with traditional search algorithm by counting the total number of nodes and tuples returned by the OLAP-favored search algorithm and the total number of tuples returned by the traditional search method. It shows that OLAP-favored search algorithm can reduce result set size to as much as 1/10 of the traditional query result. It also shows that fanout impacts the algorithm's efficiency. The advantage of OLAP-favored search is bigger advantage when the fanout is smaller.

We measure the cost advantage of our method by defining the aggregation ratio as the number of nodes and tuples returned from OLAP-favored search to the number of tuples returned by the traditional method in the same query. Smaller aggregation ratio indicates better cost improvement. In the experiments, we found that aggregation ratio has a close relationship with the fanout of the tree. In Fig. 8, the aggregation ratio decreases with fanout. It is because that smaller fanout makes the tree "taller", i.e., having more levels, generating more tree nodes with different size of bounding area. Thus more tree node can fully satisfy the "within" query predicate and stop the tree traversal on certain level. While bigger fanout generates a flatter tree, and decreases tree node's possibility of fully satisfying the query predicate. Therefore more leaf nodes, or tuples are returned instead of tree nodes.

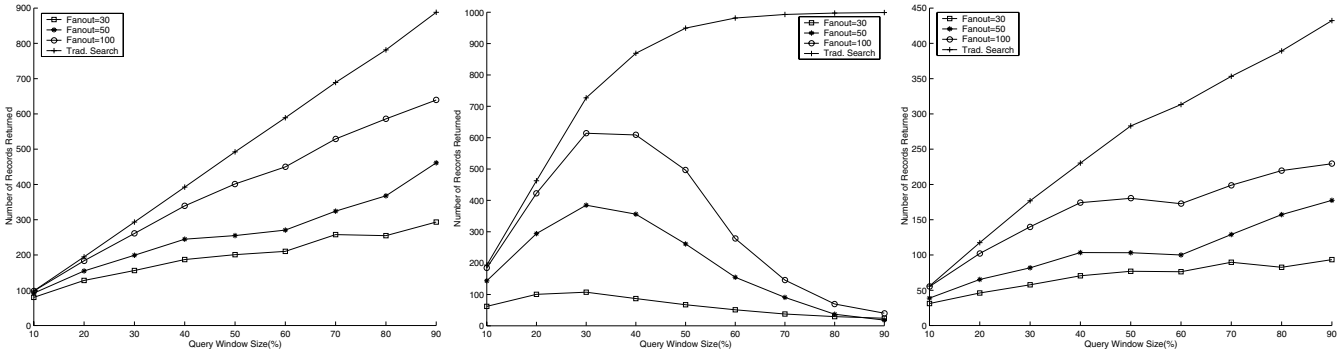
The number of nodes and tuples returned by the search algorithm impacts the cost of retrieving data from the summary table. To further quantify the internal cost of tree traversal and its relationship with the fanout parameter, we compare the number of edge visited in the OLAP-favored search to the traditional search algorithm. We also compute the ratio of edge visited in OLAP-favored search to that in traditional search algorithm. The relationship of the percentage and the fanout is depicted in Fig. 9. Smaller fanout has smaller ratio which indicates bigger saving in cost.

Experiments are also conducted to evaluate heuristic OLAP-favored search method. We evaluate AF using the method of Equation 2 developed based on uniform distribution. To investigate the error brought in the heuristic OLAP-favored search, we conduct the test on uniformly distributed , normally distributed and real data set. The number of records returned compared to OLAP-favored search and traditional search method is depicted in Fig. 10. The error measured in percentage in heuristic OLAP-favored search is depicted in Fig. 11. MinAF in heuristic OLAP-favored search algorithm can also be regarded as an indicator of expected error rate. When the sales data follows uniform distribution and spatial data follows the the same distribution model as used in evaluation of AF, we have $MaxErrorRate = 1 - MinAF$. For example, when MinAF is 0.9, the expected error rate should be less than 10%. Experiments on these data sets shows that actual error rate is far below the expected error rate, even though the distribution model and AF evaluation method may mismatch.

6. RELATED WORKS

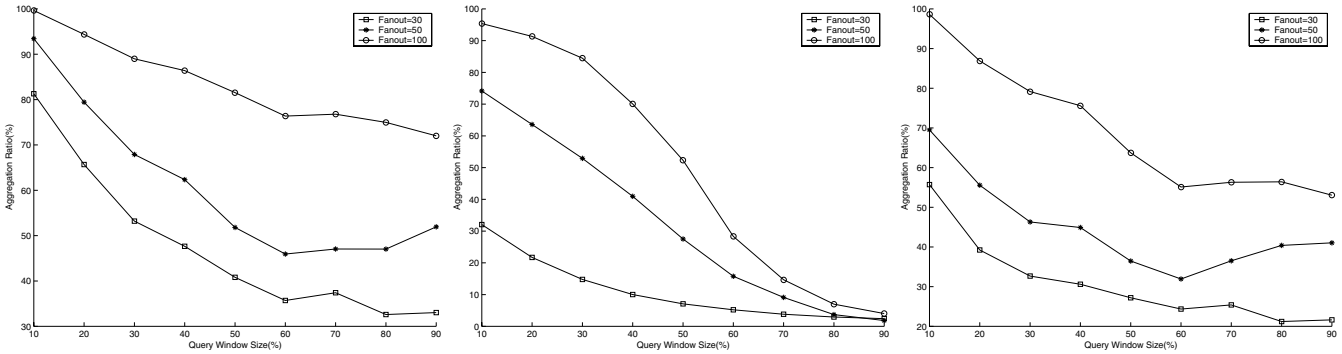
Although data warehouse and OLAP technology arose intensive interest both in academy and industry, spatial data warehouse has just become an active research topic recently. Han et al [13] did some pioneer work in this area and proposed a spatial data warehouse framework. They focused on the spatial data as measure and proposed a method to select spatial objects for materialization. Our method differs from Han's work in that we address the problem of aggregation path on spatial dimension, while he tried to manage the spatial measure and materialization. Papadias et al [17] [18] proposed an approach to store aggregation results in the index tree. Papadias's approach modifies the classical star schema in the existing data warehouse, while our method preserves the star schema in data warehouse.

Li et al[16] propose a method to enable OLAP to process predefined spatial related OLAP queries based on spatial database. The method addresses the problem of static spatial related OLAP query whose query operator and parameters don't change. In this paper, we focus on more dynamic spatial-related OLAP queries like query Q1 in which the region is defined ad-hoc by user at run time.



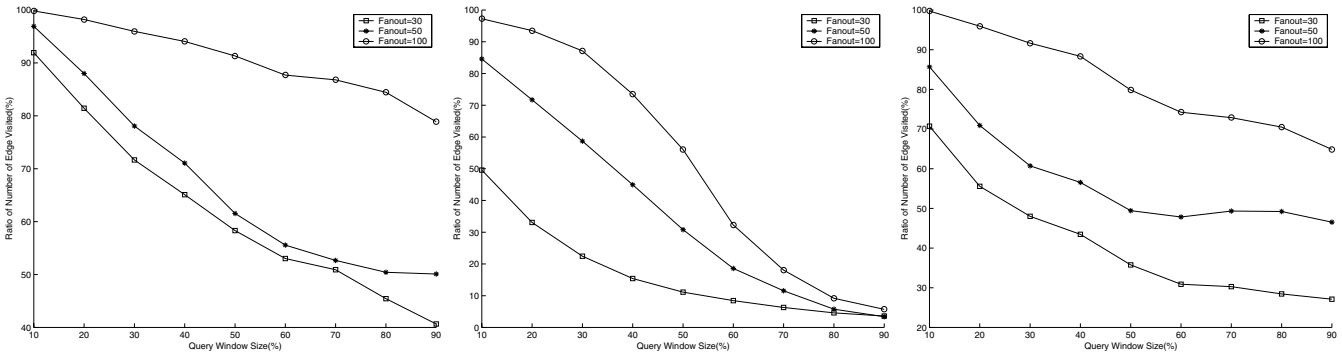
(a) Uniformly Dist. Data (b) Normally Dist. Data (c) Real Data

Figure 7: OLAP-favored Search



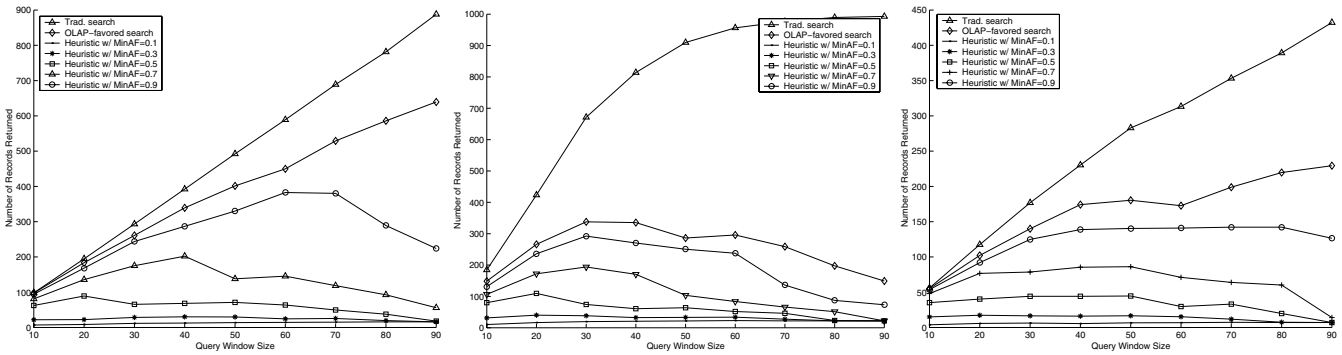
(a) Uniformly Dist. Data (b) Normally Dist. Data (c) Real Data

Figure 8: Aggregation Ratio In OLAP-favored Search



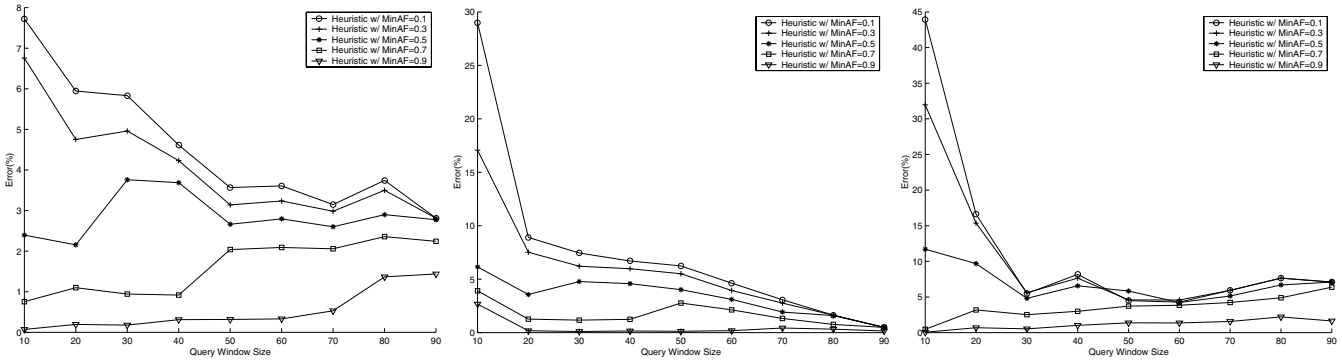
(a) Uniformly Dist. Data (b) Normally Dist. Data (c) Real Data

Figure 9: Number of Edge Visited in OLAP-favored Search



(a) Uniformly Dist. Data (b) Normally Dist. Data (c) Real Data

Figure 10: Heuristic OLAP-favored Search With Fanout = 100



(a) Uniformly Dist. Data (b) Normally Dist. Data (c) Real Data

Figure 11: Error in Heuristic OLAP-favored Search With Fanout = 100

From the access methods's point of view, statistics access methods such as SIAM [9], TBSAM [22] and LST [4] compute the scalar aggregate function over one-dimensional domain by utilizing the pre-computed aggregate result stored in the statistics tree. Our method extends the statistical access methods to multi-dimensional data and stores the pre-aggregation result in the summary table. Traversal algorithm in statistical access method has at most two edge descents, while in multi-dimensional data space there is more descents because of the bounding predicate overlap. From our experiments, fanout has a great impact on the number of "edge descents" in query processing. Smaller fanout generates less "edge descents". Aoki et al[2] mentioned a generalized statistical access method based on GiST. In this paper we outline the OLAP-favored search method based on GiST. Furthermore, we provide the heuristic search method by introducing AF as well as its evaluation method.

7. CONCLUSIONS AND FUTURE WORKS

With the increasing amount of spatial data stored in business information systems, how to utilize these spatial information when analyzing business data is becoming an important issue of data warehouse and OLAP. Traditional data warehouse and OLAP system could not exploit spatial data because spatial data is multi-dimensional and does not have a natural hierarchy to compute pre-aggregation and materialization in data warehouse and OLAP.

The contribution of this paper is three folds. Firstly, we extend the grouping-set hierarchy into multidimensional data space and use spatial index tree as the hierarchy in spatial data warehouse. With hierarchy on spatial dimension, data cube and pre-aggregation can be computed in spatial data warehouse. This approach preserve the data warehouse's star schema while providing the spatial processing capabilities in data warehouse. Secondly, we develop an OLAP-favored search algorithm that can utilize the pre-aggregation on spatial dimension, and thus can enable OLAP search engine to handle spatial-related OLAP queries. Thirdly, we propose a heuristic query method which can further improve the OLAP query performance.

Experiment results show the efficiency of our method. We found that the tree's fanout and query cost is closely related. Smaller fanout reduce the edge visited during query processing, but result in bigger tree and summery table in the meanwhile. The cost trade-offs need to be further investigated. We introduce the factor of AF and propose a heuristic search algorithm. General evaluation method and a particular method based on uniform distribution of underlying spatial data are also given. Further study can explore the detailed AF computation method on different data distribution

in space, and model the error brought by the heuristic search.

We implement a prototype system based on the method [24]. The prototype shows the feasibility of seamlessly integrating spatial data into traditional OLAP systems. Experiment on the prototype also shows the efficiency of the method.

8. ACKNOWLEDGEMENT

We would like to thank Dr. Shi Xia Liu for her careful reading and constructive criticism.

9. REFERENCES

[1] D. W. Adler. DB2 spatial extender - spatial data within the RDBMS. In *Proc. of VLDB*, pages 687–690, 2001.

[2] P. M. Aoki. Generalizing "search" in generalized search trees. In *Proc. 14th ICDE*, pages 380–389, 1998.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proc. of SIGMOD*, pages 322–331, 1990.

[4] M. Chen and L. McNamee. On the data model and access method of summary data management. *IEEE Transactions on Knowledge and Data Engineering*, 1(4):519–529, 1989.

[5] N. Colossi, W. Malloy, and B. Reinwald. Relational extensions for OLAP. *IBM SYSTEMS JOURNAL*, 41(4), 2002.

[6] I. Daratech. Daratech: Geographic information systems markets and opportunities. 2000.

[7] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[8] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[9] S. Ghosh. Siam: Statistics information access method. In *Proc. of the 3rd International Workshop on Statistical and Scientific Database Management*, pages 286–293, 1986.

[10] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *J. Data Mining and Knowledge Discovery*, pages 29–53, 1997.

[11] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. of SIGMOD*, pages 47–57, 1984.

[12] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publisher, Inc, 2001.

[13] J. Han, N. Stefanovic, and K. Koperski. Selective materialization: An efficient method for spatial data cube construction. In *Research and Development in Knowledge Discovery and Data Mining, Second Pacific-Asia Conference, PAKDD'98*, pages 144–158, 1998.

[14] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. Generalized search trees for database systems. In *Proc. 21st VLDB*, pages 562–573, 1995.

[15] W. H. Inmon. *Building the Data Warehouse*. John Wiley Sons Publishing Company, 1992.

[16] Y. Li, Y. Chen, and F. Y. Rao. The approach for data warehouse to answering spatial OLAP queries. In *the Fourth International Intelligent Data Engineering and Automated Learning*, 2003.

[17] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP operations in spatial data warehouses. *Lecture Notes in Computer Science*, 2001.

[18] D. Papadias, Y. Tao, P. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In *ICDE*, 2002.

[19] S. R. Ravi Kanth V Kothuri and D. Abugov. Quadtree and R-tree indexes in Oracle spatial: A comparison using GIS data. In *Proc. of SIGMOD*, June 2002.

[20] J. T. Robinson. The k-d-b-tree: a search structure for large multidimensional dynamic indexes. In *Proc. of 1981 SIGMOD*, pages 10–18. ACM Press, 1981.

[21] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A dynamic index for multi-dimensional objects. In *The VLDB Journal*, pages 507–518, 1987.

[22] J. Srivastava and V. Y. Lum. A tree based access method(TBSAM) for fast processing of aggregate queries. In *ICDE*, pages 504–510, 1988.

[23] TIGER/Line files technical documentation, US Census Bureau, 2002.

[24] L. Zhang, Y. Li, F. Rao, X. Yu, and Y. Chen. An approach to enabling spatial OLAP by aggregating on spatial hierarchy. In *Proc. Data Warehousing and Knowledge Discovery (DaWaK)*, 2003.