Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 4 (2012) 766 – 771

Procedia Technology

C3IT-2012

New Database Architecture for Smart Query Handler of Spatial Database

Parthasarathi Boyal[a], Rituparna Chaki [b]

[a]*West Bengal University of Technology, BF42 SaltLake City, Kolkata-700064, India*
[b]*West Bengal University of Technology, BF42 SaltLake City, Kolkata-700064, India*

Abstract

A spatial database system is a database system with additional capabilities for handling spatial data. It also supports spatial data types in its implementation, providing spatial indexing and efficient algorithms for spatial join. The retrieval of data values from a spatial database involves searching through the huge repository of data, involving huge cost. Thus query optimization on spatial database takes more time as compared to RDBMS. The current state of art shows that during the execution of a query in a spatial database management system (SDBMS), the query optimizer creates all possible query evaluation plans. All plans are equivalent in term of their final output but vary in their execution cost, the amount of time to run. Once the data is retrieved the query and its plans are deleted from memory to free the space for future usage. This is repeated for the next query even if the query is already executed. This leads to increased storage overhead and execution time. In this paper, a new database architecture is proposed, which uses a buffer based query optimization technique for faster data retrieval.

Keyword: Query Optimization, Spatial database, Spatial indexing;

1. Introduction

A spatial database system is specialized for handling spatial data. It is mainly used in Geographical information system (GIS), meant for capturing, storing, analyzing, and managing data which are spatially referenced to earth. The data retrieval cost increases in a spatial database due to the requirement of huge space and size. The query optimization on spatial data also takes more time as compared to RDBMS**.** Spatial query is the most frequent operation in distributed spatial databases system. One of the most important goals of spatial database is to efficiently process the queries related to the stored data. Many query algorithms work on a single database for both spatial as well as attribute data. These include point location queries, range queries, nearest neighbor queries and reverse nearest neighbor queries. The nearest neighbor query algorithm [8] is used to find k nearest neighbor to a given point in the space. In this algorithm R-tree traversal algorithm is used to find a nearest point to a particular given point. The R-tree

doi:10.1016/j.protcy.2012.05.125

was used here to settle the index problem. The main strategy of R-tree is to recursively cluster the multidimensional spatial nodes using minimum bounding rectangles (MBR) which is the smallest rectangle enclosing spatial nodes. The main problem of the R-tree lies in overlapping those MBRs' leading to multiple search paths. To solve these problems Range nearest-Neighbor Query [8] was proposed. But performance of this algorithm degrades as k-nodes increases. In our approach we solve this problem using GPS enabled node. However, another type of queries involve relating data from multiple databases for both spatial as well as attribute data. Here several databases for spatial data and attributes joined into one system are called hybrid system [10]. In this paper we have done optimization on this hybrid system. A Hybrid system stores geometrical data and attribute data in two separate databases. In hybrid database system [10] all these approaches use optimization technique the handle multiple table. In the hybrid system Common challenges are: Optimization is generally done on spatial data and not on attribute data; Attribute data constitutes a major portion of a spatial query. Therefore, optimization on spatial data takes more time as compared to RDBMS; huge databases require proper optimization for efficient search.To overcome the above challenges of the standard database architecture, new database architecture was proposed by introducing an extra schema "Intelligent Schema" [10] in between the External schema and the Conceptual Schema for ubiqutious system. To optimize the spatial DBMS query optimization [10] is used here.

2. Algorithm Description

It is observed from the study of the current state of art that during the execution of a query in a spatial database management system (SDBMS), the query optimizer creates all possible query evaluation plans and chooses the best possible plan. Once the data is retrieved the query and its plans are deleted from memory to free the space for future usage. For the next query same technique is repeated even if the query is already executed. This leads to increase storage overhead and execution time. In the following section we discuss indexing to achieve the parallel processing concept for distributed spatial database system and system architecture of the proposed Smart Query Handler. A buffer based technique is introduced that can store previously executed plan. This helps in faster processing. We use spatial indexing that makes the retrieval of data faster. Here every lead node has nth level of information of its MBR. So we can search a node in multiple paths also. This problem can be solved by using intelligent query optimizer and all lead node stores the least cost path.

2.1 Spatial data partitioning

In the data partitioning phase, we assume there are m sites and n spatial nodes. Let S = $\{s_1, s_2, s_m\}$ be a set of sites. Let N= $\{N_1, N_2, .N_n\}$ be a set of spatial nodes. We adopt the same strategy as the packing R-tree method. The only difference is we store spatial nodes in different site according to special rule. It can be specified as follow:

1. Sort N by x-coordinate ascending, we get a new set N= $\{N_1, N_2 \ldots N_n\}$;
2. Sort N' by y-coordinate, we get a new set N"= $\{N_1, N_2 \ldots .Nn\}$;

Here all nodes are GPS enabled, so distances can be found easily using their coordinates. Let coordinates of N1 node (x_1, y_1) and N2(x_2,y_2) so their distances will be $\sqrt{\{(y_2 - y_1)^2 + (x_2 - x_1)^2\}}$.

2.2 Spatial index building

The Basic thought of spatial index, which is also the fundamental thought of spatial query, is to carry out the thought of approximation. By this means, it is possible for index structure to manage data according to one or several conditions. The most popular index structure is R-tree. In this Section, the problem of

constructing index building of 2-d nodes is being considered to eliminate multiple paths using GPS enabled node .This algorithm is based on packing R-Tree indexing using GPS enabled nodes.

Algorithm for index building

Step 1: Store all nodes in an array DB[i] and sort the array in descending order.
Step2: Take the 1st node and calculate distances with other nodes.
Step3:

- Select the nearest node of the 1st node. /*using RNN algorithm for finding nearest node.*/
- Build the cluster using nearest node of the 1st node.
- Store 1st node and nearest node to cluster[ ]
- After building the cluster delete the nodes from the array DB[i]

Step5: Do step 1 to step 3 starting with the new database
Step6: Repeat step 3, step4 and step5 until DB[i] is empty.
Step 7: Build the index using these cluster.

- Construct the root node: Assign the cluster list as a root.
- Assign the internal node by using the member of individual cluster.
- Assigning leaf node: Spilt the individual cluster put the member of the cluster in different site.

3. Smart Query Handler

After indexing, optimization of the data retrieval is important for spatial database. The Smart Query Handler (SQH) consist of new query handler and optimizer(NQHO), Spatial query matcher (QM), buffer, query bank (QB) and Spatial query optimizer and execution plan loader (SQOEPL).The function of each parts of intelligent query optimizer is described in the following section

- *New query handler and optimizer*

Every query is passed to the NQHO and NQHO passes that query, as it is to the QM. QM compares the query with those in buffer.

- *Query matcher*

This component of intelligent QM searches the input query from buffer. When a query comes for processing new query handler passes the query to QM for checking availability of the query in buffer. Here three cases may arise, query is available in buffer; query is not available in buffer but is in Query Bank; query is not available in Query bank. If query matches with those in the buffer, query plan is accessed and executed directly. After execution WM increase the query weight by one. If the query does not match with those in buffer and QB then NQHO passes the query to SQOEPL. The SQOEPL swaps next queries of higher weight from QB into buffer according to buffer size and takes the already existing queries into the QB. This process is repeated until the QM finds the required query. When query is found its execution plan is accessed and processed. In the third case match is not available in the Query Bank. The Spatial Query optimizer and Execution Plan Loader creates all possible execution plans.

- *Buffer*

Buffer temporarily holds the processed queries and their best plans of the most frequent query. The size of buffer is limited so limited number of queries is loaded in the buffer. So it reduces searching cost if the best plans locally available. When database application is closed all queries and their plans are moved to Query Bank.

- *Weight Manager*

The WM is responsible for assigns weight to processed queries. It increases the weight of query by one after processing the query and assigns weight one to query, which is executed first time

- *Spatial Query Optimizer And Execution Plan Loader*

If QM searches the query plan in buffer and QB and fails to find then it passes the new query to SQOEPL, then SQOEPL parsed the query in different ways then presented to a query processor that select the best plan and load it in to the QB. Then the best plan is passed to the NQHO. After the execution of best plan the WM assigns weight one to the query and query with its executed plan is saved into QB and SQOEPL once again refreshes the buffer, it takes the buffers queries into QB and loads the queries of highest weight from QB into buffer again.

- *Query Bank*

All executed queries and their best plans are stored in Query Bank in descending order according to weight assigned by Weight Manager. When database application is closed all queries and their plans are moved to Query Bank.

- *Query Coordinator*

For the creation of new execution plans query coordinator provides database statistics from data dictionary and for the processing of query it provides data from database also. If the data is not available in local database then Query Coordinator searches the index in different site simultaneously retrieve and by finding out spatial nodes whose MBR enclose the appoint point.

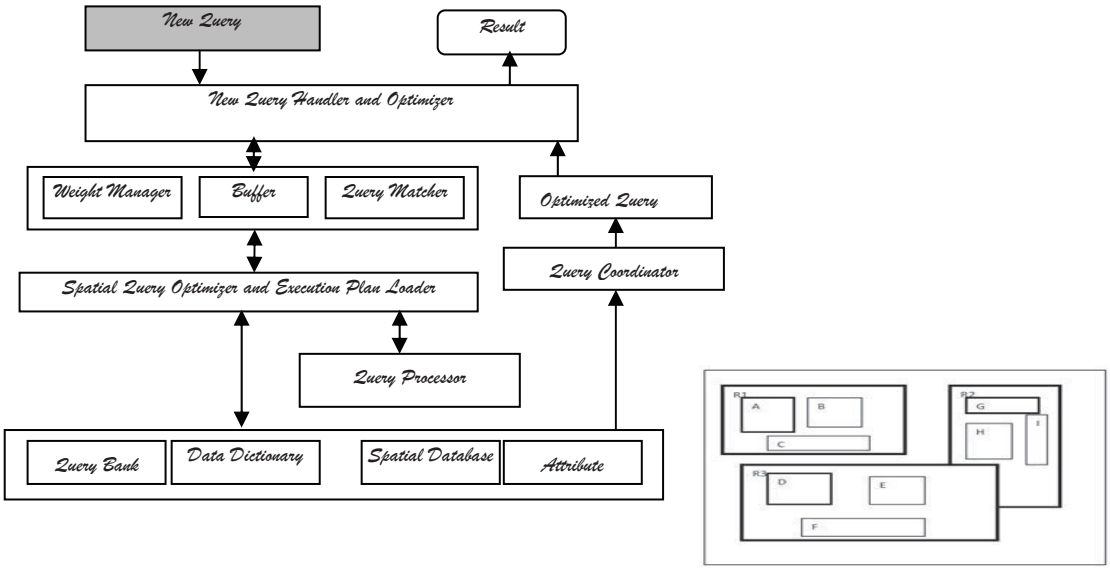


Fig: Architecture of Smart Query Handler and clustering of node

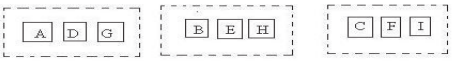
4. Case Study

For a disaster management system we need some optimization for retrieval data at least cost and time. For this city must be divided into different region that region must be represented by GPS enable lead node that can have routing information for outer region and inner region. Here node A need information about exit route from a disaster affected area then it's required some optimize Query plan .Here the sorted initial set is:A,B,C,D,E,F,G,H,I

Step1: Store the initial set in a array DB[i]
Step2: Take the first node A and calculate distances with other node.
Step3: If distance (A, B)> distance (A, F), then A, and B are stored in list R_1.
Step5: Form the cluster list by repeating step 3 for entire list
Step6: Assign the entire list as a root
Step 7: Assign the internal node by using the member of individual cluster

Step8: Assigning leaf node: Spilt (data partitioning) the individual cluster put the member of the cluster in different site.

By using data partitioning technique on this initial data set we get this following data set



Let's take the following queries:

select all from roads where in_window (road_coords, w) or road_name ="route1" (weight 3)

select all from roads where in_window (road_coords, w) and road_name! ="route1" (weight 2).

select s.sname from reserves r, sailors swhere r.sid=s.sid and r.bid=100 and s.rating >5 (weight1).

The SQOEPL search the query in the QB using keywords and operator. SQOEPL can't find any match with previously executed query. The SQOEPL passed the query to query processor .Queries are parsed [8] by Query Processor (QP). QP generates alternative plan and present all alternative plan to the SQOEPL. The SQOEPL choose the plan with the least cost. WM assign weight to processed queries depending upon the degree of execution. Weight of the query is increased by one after execution of the query for the first time. If Node B generates same query then weight of this query will be increased by one. If this query achieves higher weight among the query in the buffer then SQOEPL swap this query with another query in the buffer. Intelligent query optimizer returns the optimized query plan that can retrieve the data at least cost and time. If the data is not available in local database then QC searches through index in different site simultaneously. Here in this index structure Node A, D and F in site1and all node is directly connected to other site by internal nodes of the index. So Node A is directly connected to Site3 by this following path (site1) A→D→F (site3)

5. Future Work

This paper presents an efficient algorithm which helps the organization of spatial data and use some optimized technique to retrieve data from intelligent system. Further, based on this buffering technique the retrieval of data in distributed spatial database will be more efficient. In this technique once optimal execution plan is selected for a query then there is no need to create any execution plan for similar query in future. So when this proposed model will be applied in the distributed spatial database, then the people can access the spatial data all over the network at an affordable cost. The proposed algorithm has some limitations in replication of index will waste plenty of storage space when the spatial dataset is huge. In addition, it would be worse than the serial algorithm when spatial query failed to find any spatial object which satisfied the query requirements. In future our focus will be the implementation of the system along with load balancing. These problems mention above need us to research deeply in future. In the conclusion a clam can be made the proposed algorithm is efficient enough in comparison to other existing intelligent query optimizer for distributed spatial database. This will have tremendous impact on automatic database tuning and other query optimization processes. The more experiments in future may substantiate this clam.

Reference

1. Kamel I, and Faloutsos C., "On packing R-trees", Proceeding of the 2nd conference on information and knowledge management (CIKM), Washington DC. Pp.490-499, November1993.
2. Abel D., "Spatial Join Strategies in Distributed Spatial DBMS", Proceedings of fourth international symposium on Large Spatial Databases, Maine, pp.348-367, August 1995.
3. Feng Lu, and Chenghu Zhou., "A GIS spatial indexing approach based on Hilbert ordering code", cad$cg, Vol 13, No. 5, pp. 424-429, May. 2001.
4. J. Ronald Eastman, Michele Fulk, and JamesToledano: The GIS Handbook. Clark University, 2003
5. Spatial Databases: Authors: Shashi Shekhar and Sanjay Chawla. Publisher: Prentice Hall, 2003

6. Nearest-Neighbor Query by NickRoussopoulos Stephen Kelley Frederic Vincent ,Deptment of Computer science , University of Maryland IEEE Transactions on knowledge and data engineering, vol. 18, no. 1, january 2006.
7. Beomseok Nam, and Alan Sussman., "Spatial indexing of distributed multidimensional datasets", Proceeding of International Symposium on Cluster Computing and the Grid, Cardiff. pp. 743-750, May 2005.
8. Range Nearest-Neighbor Query Haibo Hu and Dik Lun Lee IEEE Transactions on knowledge and data engineering, vol. 18, no. 1, january 2006.
9. Providing Diversity in K-Nearest Neighbor Query Results Anoop Jain, Parag Sarda, and Jayant R. Haritsa Database Systems Lab, SERC/CSA Indian Institute of Science, Bangalore 560012, india November2006.
10. An Intelligent Framework for Distributed Query Optimization of Spatial Data In Geographic Information Systems Prashanta Kumar Patra Chittaranjan Pradhan Animesh Tripathy IJCSNS International Journal of Computer Science and Network Security, vol.8 no.5, may 2008.