International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 3 (2015) pp. 5905-5911
© Research India Publications
http://www.ripublication.com

An Efficient Bundle Range Aggregation Using R-Tree

Ms.S.P.Godlin Jasil[1] and Vennila Mani [2]

*[1] Asst. Professor, Faculty of Computing,
Sathyabama University Chennai, Tamil Nadu, India
Email:godlin_jasil@yahoo.co.in
[2]Student, Department of Computer Science,
Sathyabama University, Chennai, Tamil Nadu, India
Email: vennilacs @ymail.com*

Abstract

In Bundled range aggregation, a series of aggregated queries consecutively on multiple datasets and returning the query result on every outcome. An arbitrarily chosen dataset is enquired and it will track on numerous dataset and yields the result. The main tricky stated in bundle range aggregation is the query cost function. Previously several techniques are conversed for dipping the query cost but those methods didn't proficiently diminish the query cost. An innovative method should be proposed to shrink the query cost function. A fresh procedure using aggregated bundled R –Tree is anticipated and this system will efficiently process I/O demands and reduces the query cost.

Key Terms— bR-tree, Bundled range aggregation, minimum bounding rectangle, Indexing.

I INTRODUCTION

R-trees are tree data structures recycled for accessing spatial methods, i.e., for the tenacity of indexing multi-dimensional material such as geographical coordinates, boxes or polygons. The key hint of the data structure is to bunch nearby objects and symbolize them with the minimum bounding rectangle in the following advanced level of the tree [4]; the "R" in R-tree represents the rectangle. Meanwhile the bounding rectangle encompasses all the, the delimited object cannot be interconnected if the specified request does not overlap the bounding rectangle. Every rectangle is styled as a single object at the leaf level and at the higher levels it is designated as the aggregation of a snowballing quantity of objects. It is nonentity but the increasing granular estimation of the data set.The aggregation calculates an aggregate outcome of the data setsfilling the range predicate; each item may have a value and the total weight of the items in the interval returns range sum query. In the same way, bundled range aggregation can also be performed using other aggregate functions [8]. This can be regarded as the simultaneous execution of a range aggregate query on multiple datasets, returning a result for each dataset [1]. Likewise the set of data items has a key and a weight, each data item also carries a colour [1].

II RELATED WORKS

Spatial Indexing

The data are overloaded into the spatial table either one by bulk loading or transactional loading. On one occasion the data are loaded, the spatial index must be created for accessing the data efficiently. Thus the spatial index will be an R-tree index or a quadtree index. If you create a spatial index without specifying any quadtree-specific parameters, an R-tree index is created. R-tree indexes are built as two, three, or four dimensions of data. By default the number of dimensions for an R-tree index will be two, if it is more than two dimensions, you can use the (sdo_indx_dims) parameter keyword to specify the number of dimensions on which to build the index [7].

Indexing uncertain categorical data

While indexing, the main problem faced is uncertainty in categorical data. This uncertainty of categorical data occurs in many of the applications that include data cleaning, integration of database, and biological annotation. In those domains, the exact value of an attribute is mostly unknown, but may be selected from a reasonable number of alternatives. This type of uncertainty is not provided conveniently by current database management system that is meant for representing or manipulating.Thus two index structures are proposed for efficiently searching uncertain categorical data, one based on the PDR-tree and another based on an inverted index structure. Using these structures, detailed descriptions of the probabilistic equality queries they support are provided [2] [6].

Temporal Aggregates

Some problems arise while maintaining the materialized temporal aggregate in the temporal incremental computing aggregation queries in temporal records [3]. The complications are resolved using a new indexing structure called the SB-tree. The SB-tree incorporates features from both segment-trees and B-trees. Fast lookup of aggregate results based on time is supported by SB-tree, and when the data changes it can be maintained efficiently. This effort extend the basic SB-tree index to handle cumulative (also called moving-window) aggregates, considering cases separately when the size of the window is or is not fixed in advance . Views in a temporal database or warehouse, building and maintaining SB-tree indices for materialized aggregate are proposed instead of the views themselves. By incorporating features from segment-trees, SB-trees are more efficient to maintain than materialized temporal aggregates, particularly in the presence of base tuples with long valid intervals. Furthermore, SB-trees contain enough information to construct the contents of the temporal aggregates that they index. These features make SB-trees a particularly effective structure for supporting temporal aggregates in data warehouses [5].

III PROPOSED SYSTEM

The proposed system uses a bundled R-Tree, this method will perform more fast and efficient then bundled aggregate B-Tree. This method is also dynamic and easily answers all queries. The bR-tree stores the aggregate value of each sub tree in the index record pointing to this sub tree. Computing an aggregate is now faster since the aggregation information is used to eliminate various search paths. Let SD be a spatial database (Crime database with city, crime number, data and month) and C a spatial relation that stores the positions of real-time datasets. C is indexed by a bR-Tree RC. Let R be a spatial relation that stores all the objects that belong to the spatial dimension (i.e. crime reports), at the finest granularity. R is also indexed by a bR-Tree RR. Let AG(·) be the aggregation function. Without loss of generality, we will assume that AG (·) is COUNT, even though any non-holistic function can be used. The bundled R-Tree(bR-tree) is an R-Tree which stores for each minimum bounding rectangle (MBR), the value of the aggregation function that are enclosed by the MBR for all the objects. The bR-tree is built on the finest granularity objects of the spatial dimension; therefore its structure is similar to that of RR (the trees can be different due to the smaller fan-out of the bR-tree). A bR-tree which indexes a set of five crime reports, r1 … r5, who's MBRs are a1 … a5 respectively.The general concept can be applied to different types of queries; for instance, instead of keeping aggregated results of joins the bR-tree could store such results for window queries. Furthermore we could employ the same idea to other data partitioning or space partitioning data structures (e.g., Quadtrees).This method also implements new updating algorithms which includes insertion and deletion process.

The overall architecture is stated above (Fig 1). The work flow of the overall system architecture states that, from the real-time datasets the bR-tree is constructed and it is stored into the server. Meanwhile the user gives a query and thus the query result is obtained from the server using updating process where insertion and deletion process are carried out. That is termed to be as patching. Finally after these processes the requested

Query is sent backfrom the serverto the user. Thus, the result consumes less processing time and is of reduced query cost by the using bR-tree. The phases are explained below in order.

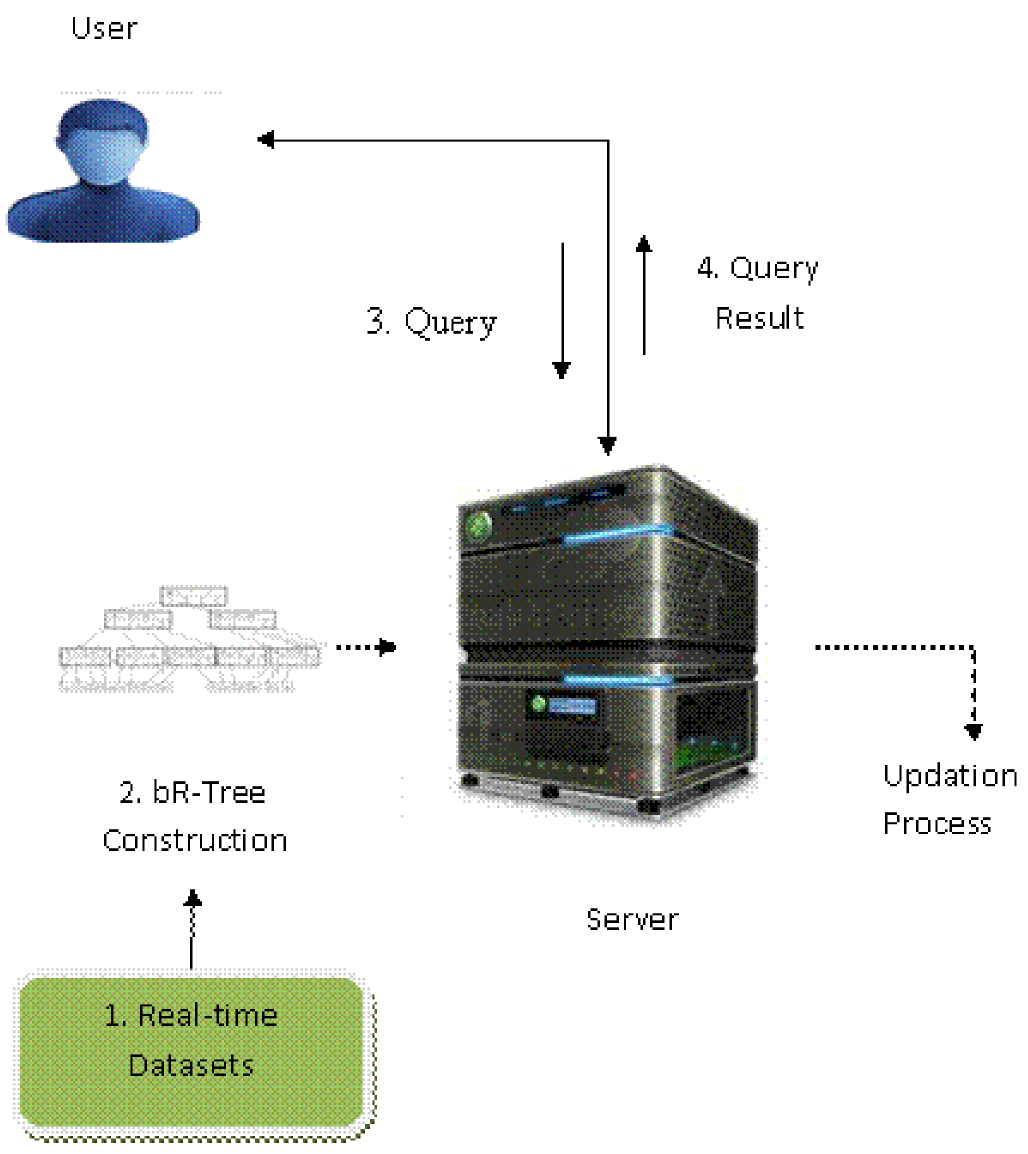

Fig 1: Functional architecture

Construction of bR-tree

The bundled R-Tree (bR-tree) stores the value of the bundled function for all the objects that are enclosed by the Minimum Bound Rectangle. The bR-tree is built on the finest granularity objects of the spatial dimension; therefore its structure is similar to that of RR (the trees can be different due to the smaller fan-out of the bR-tree). The real-time data is considered as one dimensional data for multidimensional data a data cube lattice framework is proposed for multi-dimensional data. Data cube lattice will cover all the data.

Indexing spatial data

In this module, the datasets collected from real-time applications are indexed and stored in the bR-tree. In bR-tree indexing is the important process for quick search, and efficient retrieval of datasets. In this the spatial data is indexed with spatial indexing process and stored in the tree using insertion algorithm. The indexed datasets are stored in their corresponding location.

Query Processing

In this module, a bundle range query is given by user to the server. Then the given query will travel through the tree and run on every bundle range dataset. The aggregation function processes the query and produces the datasets to the requested user to split up and process the query. The Tree traversal is done by using (r.MBR.pointer.Node id)

Update algorithm

In this module, update algorithms are implemented for changes of data. If the data changes in the dataset then updating process takes place. It includes reinsertion and deletion of data from the database. Patching includes insertion and deletion of data in the tree and counter overhaul avoids the overload of patching process.

IV RESULTS AND DISSCUSSION

In this section, a thorough discussion about the result and performance measures of the proposed system is discussed. Consider a scenario where bR-tree and aBB-tree is compared for their retrieval efficiency as well as with their processing time. First consider the comparison of the retrieval efficiency between the bR-tree and aBB-tree where it is depicted in the graph below (Fig 2), in the graph the pink line indicates the retrieval efficiency of the bR-tree and the blue line indicates the retrieval efficiency of the aBB-tree. It is clear that the retrieval is efficient while using bR-tree as the pink line is higher than the blue line.

Next consider the comparison of the processing time between the bR-tree and aBB-tree where it is depicted in the graph below (Fig 3), in the graph the pink line indicates the processing time of the bR-tree and the blue line indicates the processing time of the aBB-tree. It is clear that the processing time of bR-tree is lesser than the aBB-tree, which is the pink line is below the blue line.

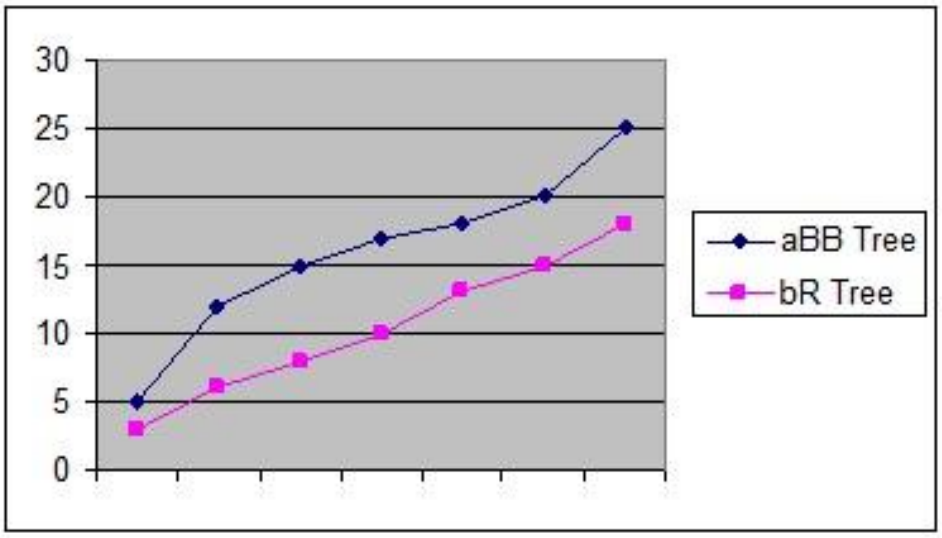

Fig 2: Line Chart depicting retrieval Efficiency.

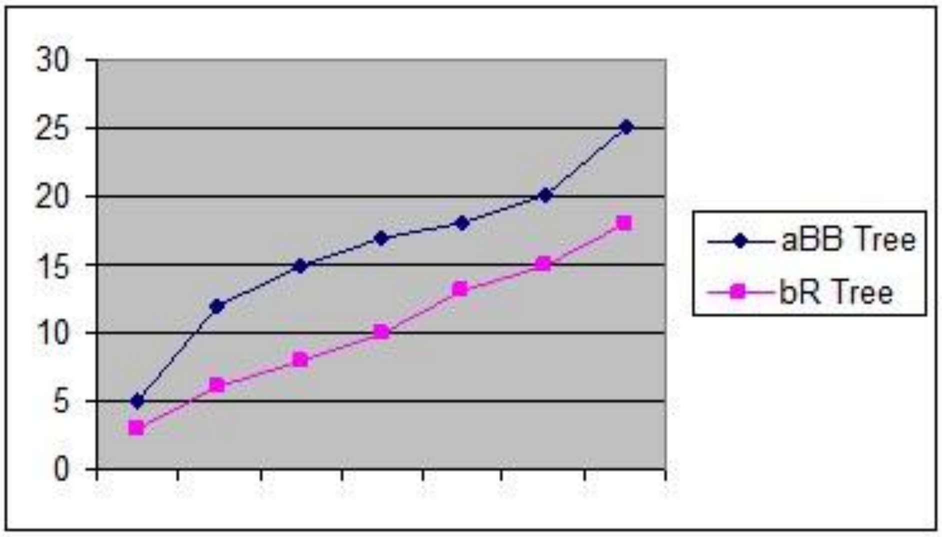

Fig 3: Line Chart depicting query processing time.

V CONCLUSIONS AND FUTURE WORK

A bundle range aggregation is proposed using R-tree. Thusbundled R-tree provides an efficient bundle range aggregation and answers to any bundle I/O queries from the user and run on multiple datasets and produces the result. This method is proposed to minimize the query cost and dynamic in nature. This method reduces query cost and efficiently answers user queries compared to other bundle range aggregation methods.Further, it can be enhanced for implementing bundled function and to support all type of data's in thebRTree.

REFERENCES

[1] Yufei Tao and Cheng Sheng. "I/O-Efficient Bundled Range Aggregation."IEEE Transactions on Knowledge and Data Engineering (2013)

[2] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. E. Hambrusch. Indexing uncertain categorical data. In Proc. of International Conference on Data Engineering (ICDE), pages 616–625, 2007.

[3] J. Yang and J. Widom. Incremental computation and maintenance of temporal aggregates. The VLDB Journal, 12(3):262–283, 2003.

[4] L. Arge. The buffer tree: A technique for designing batched external data structures. Algorithmica, 37(1):1–24, 2003.

[5] D. Zhang, V. J. Tsotras, and D. Gunopulos. Efficient aggregation over objects with extent. In Proc. of ACM Symposium on Principles of Database Systems (PODS), pages 121–132, 2002.

[6] S. Govindarajan, P. K. Agarwal, and L. Arge. CRB-tree: An efficient indexing scheme for range-aggregate queries.InProc.of International Conference on Database Theory (ICDT), pages 143–157, 2003.

[7] D. Papadias, Y. Tao, P. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In ICDE, pages 166–175, 2002.

[8] D. Zhang, A. Markowetz, V. J. Tsotras, D. Gunopulos, and B. Seeger. On computing temporal aggregates with range predicates. ACM Transactions on Database Systems (TODS), 33(2), 2008.