RESEARCHERID
THOMSON REUTERS

[ICRTCET-2018]

ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

EFFICIENT GEO-TEXTUAL QUERY PROCESSING USING GRID-INVERTED FILE HYBRID INDEX AND CONCEPTUAL PARTITIONING

SSulbha K. Powar[*1] & Dr. Ganesh M. Magar[2]

[*1]Research Scholar, P. G. Department of Computer Science, SNDT Women's University Mumbai, India

[2]Associate Professor, P. G. Department of Computer Science, SNDT Women's University Mumbai, India

ABSTRACT

Growth of geo-spatial data has increased its use in variety of applications. Geographic Information System(GIS) technology has tremendously evolved to handle this data. Handling huge location-based data efficiently in decision support systems is vital. Location based data also has textual and temporal components along with latitude and longitude. Efficient indexing techniques support searching and retrieval of geo-textual data in optimized way. Various hybrid techniques are studied and developed in the past. In this paper efficient hybrid geo-textual indexing technique based on grid index and inverted file is presented. Its use in geo- textual query is also demonstrated. The paper also presents the implementation of conceptual partitioning monitoring (CPM) algorithm. Conceptual Partitioning reduces the query processing time by reducing the number of hits to the cells. Geo- textual hybrid indexing, and conceptual partitioning of the grid file together process kNN query for geo-textual data efficiently

Keywords: *Conceptual Partitioning, Geo-Textual data, Geo-Textual Query, Grid index, Inverted file, nearest neighbour.*

I. INTRODUCTION

With wide availability of various technologies like Global Positioning System (GPS), radio frequency identification (RFID), sensor technologies, satellite, smart phones and other devices, location-based data is made available on huge scale. This data is useful in variety of Decision support systems like weather forecasting, urban planning, transportation planning, agriculture, emergency management and many other.

Geo-textual objects have latitude, longitude and attribute data describing these objects. Many applications need collection and maintenance of geo-textual data. Geographic Information System (GIS) has capabilities of handling these objects. These objects are captured, stored, analyzed and manipulated by GIS. Mainly 3 types of queries are answered over the geo-spatial data viz. spatial range queries, nearest neighbour queries and spatial join queries [1]. Most of the operations on location data need to answer these kinds of queries. Hence efficient techniques are required to handle geo-textual queries involving location as well as attribute data.

Geo-textual query takes location and keywords of the query objects and retrieves objects those are spatially and textually relevant to the query arguments. To answer these queries efficiently, geo-spatial data is indexed. Several well-known techniques available to index data are either based on location or based on text [1]. Spatial indexing techniques like R-tree based indices, grid-based indices and space filling curve -based indices are most popular. Inverted files-based indices and signature file-based indices are commonly used text indexing techniques. To answer geo-textual queries, spatial indexing and text indexing techniques are applied one after the other in any order. This method is inefficient as it does not support filtering out the objects early enough, which do not satisfy either of location or text criteria. If spatial and text indexing techniques are combined tightly, both location and textual attributes can be used to prune the search space simultaneously for query processing.

The geo -textual indices combine spatial and text indexing to create hybrid index [1]. Hybrid indices can be classified as text-first loosely combined, spatial-first loosely combined, text-first tightly combined and spatial-first

tightly combined [2]. This paper presents tightly combined index structure which combines grid structure and inverted files. With tight combination both distance and keywords can be used to prune the search space simultaneously during query processing. R -Tree is more popular spatial data structure. Grid Index is simple data structure as compared to R-Tree.

The main objective of this study is to suggest an efficient geo-textual indexing technique built using grid index and inverted files. It also suggests the efficient algorithm to answer nearest neighbor queries based on this indexing technique and conceptual partitioning.

II. RELATED WORK

In [3] authors have proposed index structure called Spatial -Keyword Inverted File to handle location-based web searches. For implementing this space is partitioned into several grid cells. Grid index is used to organize the geo -textual objects by [4]. The entire space is partitioned according to uniform grid and each object is stored in a grid cell it belongs to. Inverted list is maintained in each cell for the keywords of the objects stored in it. In [5] authors have implemented grid file which decomposes the space into a set of grids and store signatures of spatio-textual objects into grid for faster access and efficient spatial pruning techniques. SEA-CNN algorithms have implemented based on grid structure which outperforms R-Tree based kNN algorithm in In [7] ISGrid is implemented that provides efficient query processing based on Voronoi diagram.

Authors in [8] have proposed Ordered-Cell Group (OCG) based grid index structure to avoid skewedness of grid files. An efficient query processing algorithm is developed that can effectively utilize OCG cells to speed up the processing of spatial queries. [9] proposed two schemes for grid based geo-textual indexing. Spatial Primary Index (ST) and Text Primary Index (TS) are the earliest grid based geo-textual indices. They can be classified as spatial-first and text -first loose combination respectively. In [10] grid index is used for query indexing. In [11] authors have used conceptual partitioning to monitor continuous nearest neighbour query. It helps in processing objects which are nearer to the query object. In this implementation the data objects are indexed using grid file.

The outline of the paper is as follows. Section 3 explains the Grid-inverted file hybrid index. Section 4 discusses the conceptual partitioning technique. Section 5 discusses experimental work and section 6 concludes the paper.

III. GRID-INVERTED FILE HYBRID INDEX

The grid index is created using a 2 -dimensional array. The grid indices divide space into a predefined number of equal-sized square or rectangular cells. Each cell of the array represents a region of a space

Grid – Inverted file index combines a grid index with the inverted file to organize geo-textual objects. The grid index and the text index are combined tightly. Each cell C of the grid is associated with (i) the list of objects within its extent and (ii) the inverted list of the keywords of the objects in its extent. Each inverted list has vocabulary of all distinct words appearing in description of the objects belonging to the cell and a posting list for each word. Figure 1 illustrates hybrid index structure. Tight combination of grid file with Inverted file helps in pruning the search space efficiently. Dataspace is mapped to grid cells by normalizing the latitude and longitude of objects to x and y coordinates of the grid. Each geo-textual object has Latitude and longitude describing the point location and a text description of the object. Latitude and longitude is mapped to the grid cell depending upon the size of the grid structure. The object is added to the object list of the grid cell it is mapped to and its inverted file is updated for this object.

ISSN 2348 – 8034
Impact Factor- 5.070

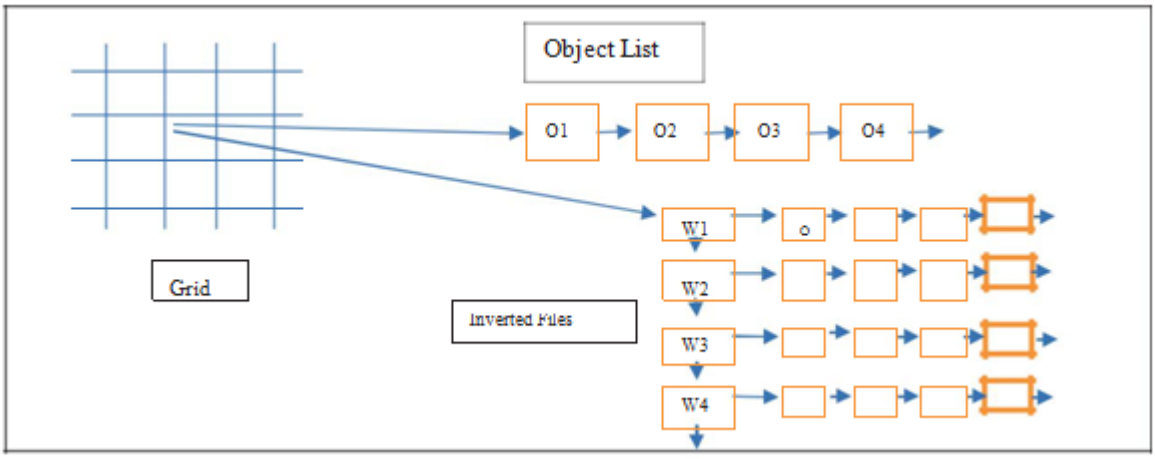

Figure 1:Grid-Inverted File Hybrid Index Structure

IV. CONCEPTUAL PARTITIONING

Conceptual partitioning as described in [11] is used to find nearest neighbour (NN) of the query point efficiently. Query object has latitude and longitude describing the point location and a set of keywords. Latitude and longitude of a query point q is mapped to the gird cell Cq. The mindist(C, q) is the minimum distance between any object p א C and the query point q. The best_kNN denotes the list of k best nearest neighbours of q found so far maintained in the order of mindist. The best_dist is the distance between the kth NN and q. The symbols used in the paper are described in the Table 1.

Table 1: Symbols

Symbol	Description
q	The query point
Cq	The cell containing query point q
dist(p,q)	Euclidean distance between object p and query point q
best_kNN	The best nearest neighbours of q
best_dist	The distance between the kth NN and q
mindist(c,q)	Minimum distance between cell c and point q
PQ	Priority Queue
DIR	Direction of the rectangle - Down, Left, Right, Up

The naive way to process kNN for q has to sort all cells C א G in mindist(C,q) order and visit them in ascending order. It is optimal with respect to the number of processed cells but is expensive because it must find mindist for all cells and sort them. Conceptual partitioning [11] avoids unnecessary computations by partitioning the grid space and doing computations only if required.

ISSN 2348 – 8034
Impact Factor- 5.070

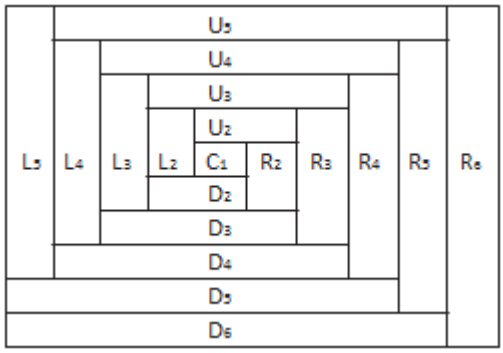

Figure 2: Conceptual Partitioning (adapted from [11])

Every rectangle in the figure 2 is labelled by the direction (Down, Left, Right, Up) and the level number with respect to the query cell with query cell as level 1. kNN algorithm visits cells and rectangles in ascending order of mindist(C, q). This preserves the property of processing the minimal set of cells.

kNN algorithm maintains the priority queue (PQ) with the nodes in the increasing order of mindist as the key. The node either has the boundary coordinates for a cell or for a rectangle. Initially the cell Cq is inserted in the PQ with mindist(Cq, q) = 0. Then all level 1 rectangles are inserted with mindist(DIR1, q). Then it starts popping the elements iteratively from the PQ. If the popped element is a cell and there are objects inside the cell, it checks inverted file for the cell. If inverted file satisfies the query keywords, then for each object in the object list of the cell, dist(p, q) is calculated and best_kNN is updated for this object. If the popped entry is a rectangle, it inserts each cell of the rectangle in the PQ with its mindist and inserts the next level rectangle in same direction with its mindist. Algorithm terminates when the PQ is empty or best_kNN has k objects and the next entry in PQ has key greater than or equal to best_dist. Conceptual partitioning gives correct result because (i) the PQ has the key as mindist which is arranged in the ascending order. (ii) Every rectangle that is inserted in the PQ has the mindist value which is greater than the rectangles inserted till now with no rectangle missed out. (iii) these rectangles in the PQ act as bounding boxes. While computing kNN, if it is found that mindist(C,q) > best_dist, C can be safely pruned because the objects contained in C will be farther than all the current kNN's of q. Figure 4 shows the algorithm to compute kNN.

Figure 3 depicts the example of how the cells are typically visited with conceptual partitioning. Cell numbered 1 is the query cell, which is visited first. Then the surrounding cells are visited in the increasing order of minimum distance of the cell with the query point. With this nearer cell (hence objects) are visited before visiting the farther cells (hence objects). Inverted list in each cell further restricts the search space by only visiting objects which satisfy the query criteria.

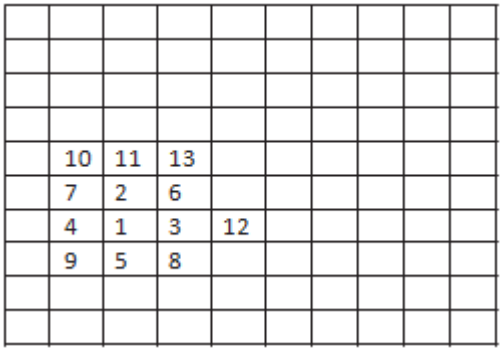

Figure 3: Example of CP cells visits

ISSN 2348 – 8034
Impact Factor- 5.070

knn_search(G, Q, k)

Input: G- Grid index,
q – query point with latitude, longitude, and keywords,

k – number of nearest neighbours

Output: set of k nearest neighbours

Algorithm:
best_dist = ’
best_NN = NULL

PQ = NULL

Insert cell <Cq, 0> into PQ

For each direction insert rectangle <DIR1, mindist(DIR1, q)> into PQ

While(!empty(PQ) and ((k objects not found) or (k objects found and key of PQ < best_dist)))

pop(PQ)

if it is cell entry
check inverted list of the cell for query keywords

if keywords found

update best_NN and best_dist if necessary

else

Insert each cell of rectangle into PQ

Insert next level rectangle in the same direction into PQ

Figure 4 kNN algorithm

V. EXPERIMENTAL WORK AND ANALYSIS

Dataset having 1,208 health center locations (latitude and longitude) from Mumbai and its attributes such as type of hospital and kind of services is used for creating grid -inverted file index structure and evaluating the kNN query. Experiments are carried with 5 different grid structure sizes. The grid data space is divided into5 different grid sizes viz. 5 x 5 grid (25 cells), 10 x 10 grid (100 cells), 15 x 15 grid (225 cells), 20 x 20 grid (400 cells) and 25 x 25 grid (625 cells) with each cell of size of 1 x 1. kNN query is evaluated with set of 10 queries for each grid structure size. Figure 5 shows the comparison of inverted nodes created with inverted nodes visited for each grid size. It is seen that as the grid size is increased the number of cells is increased and hence the number of inverted files created are also increased. But when the number of cells is less, the length of inverted file is more and hence it must visit more number of nodes to check for textual component. This is observed because as the number of cells are increased the objects are distributed among more number of cells. The performance is improved by increasing the number of cells in the grid. This observation is also true with object list.

ISSN 2348 – 8034
Impact Factor- 5.070

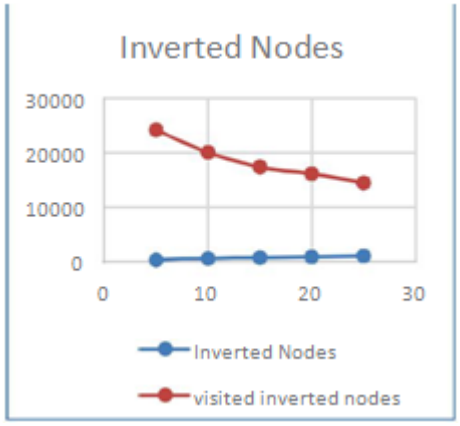

Figure 5: Inverted Nodes versus Grid size

Figure 6 shows the number of cells created in PQ, number of PQ nodes visited, and number of grid cells and rectangles visited while processing the kNN query. It is seen that with increase in number of cells these numbers are also increased. With increase in number of cells the object distribution is scattered and hence the number of hits is increased. Hence finding the appropriate grid size is essential in improving the query performance.

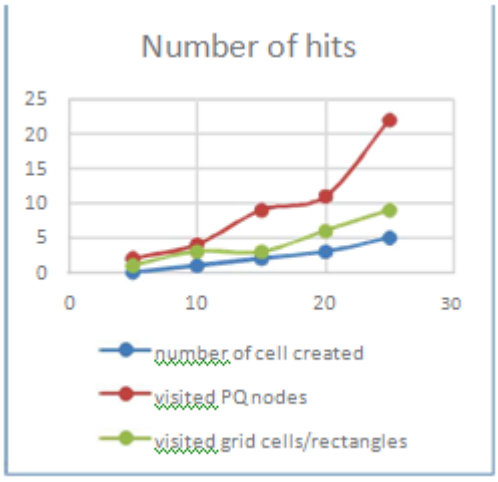

Figure 6: Number of hits for cells, rectangles, PQ nodes

The result of finding nearest private sonography centers is shown in Figure 7, where the gray markers are health center locations, green marker is query point and red markers are resulting nearest locations satisfying both spatial proximity and textual relevancy.

Figure 7: Result of geo-textual k nearest neighbour query

VI. CONCLUSION

Uniform grid makes easy to calculate the cell for latitude and longitude of given object in O(1) time. Accessing the object from database is done in just two hops with grid files. With hybrid indexing having grid files and inverted files tightly combined the search space narrowed down tremendously. This also reduces the number of false positive hits to database records. Conceptual partitioning restricts the space to be search around the query point (objects those lie near the query point) and helps in retrieving objects in efficient manner in kNN processing. Combining hybrid index and conceptual partitioning improves the query performance considerably. Skewedness in the spatial resolution of the objects is the disadvantage of the grid indexing. This can be improved by increasing the size of the grid cells. Techniques like Ordered-Cell Group (OCG) based grid index structure can be used to improve the spatial resolution.

REFERENCES

1. *L. Chen, G. Cong, C. S. Jensen, and D. Wu, "Spatial Keyword Query Processing: An Experimental Evaluation," p. 12.*
2. *S. Powar and D. G. Magar, "A REVIEW ON HYBRID GEO-TEXTUAL INDEXING TECHNIQUES," Int. J. Adv. Res. Comput. Sci., p. 4, 2017.*
3. *A. Khodaei, C. Shahabi, and C. Li, "Hybrid Indexing and Seamless Ranking of Spatial and Textual Features of Web Documents," in Database and Expert Systems Applications, vol. 6261, P. G. Bringas, A. Hameurlain, and G. Quirchmayr, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 450–466.*
4. *X. Cao, G. Cong, C. S. Jensen, and M. L. Yiu, "Retrieving regions of interest for user exploration," Proc. VLDB Endow., vol. 7, no. 9, pp. 733–744, May 2014.*
5. *J. F. G. Li, L. Z. S. Chen, and J. Hu, "SEAL: Spatio-Textual Similarity Search," p. 12.*
6. *Xiaopeng Xiong, M. F. Mokbel, and W. G. Aref, "SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio-temporal Databases," 2005, pp. 643–654.*
7. *Y. Park et al., "A New Spatial Index Structure for Efficient Query Processing in Location Based Services," 2010, pp. 434–441.*
8. *Y. Park, L. Liu, and J. Yoo, "A fast and compact indexing technique for moving objects," 2013, pp. 562–569.*
9. *S. Vaid, C. B. Jones, H. Joho, and M. Sanderson, "Spatio-textual Indexing for Geographical Search on the Web," in Advances in Spatial and Temporal Databases, vol. 3633, C. Bauzer Medeiros, M. J. Egenhofer, and E. Bertino, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 218–235.*
10. *D. V. Kalashnikov, S. Prabhakar, and S. E. Hambrusch, "Main Memory Evaluation of Monitoring Queries Over Moving Objects," Distrib. Parallel Databases, vol. 15, no. 2,pp. 117–135, Mar. 2004.*
11. *K. Mouratidis, D. Papadias, and M. Hadjieleftheriou, "Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring," 2005, p. 634.*