DOI: https://doi.org/10.53555/nncse.v2i5.446 **Publication URL: http://nnpub.org/index.php/CSE/article/view/446**

DESIGN AN EFFICIENT TOP-K QUERY RETRIEVAL TECHNIQUE FOR GIQ PROBLEMS

V.Kumaresan[1]* S. Vairachilai[2]

*[1]*PG Student, Department of CSE & N.P.R College of Engg and Tech, Dindigul, Tamil Nadu, India*

[2]*Assistant professor, Department of CSE & N.P.R College of Engg and Tech, Dindigul, Tamil Nadu, India*

*[1] kumaresanv2412@gmail.com; [2]Vairachilai2676@gmail.com

Corresponding Author: -

Email ID - kumaresanv2412@gmail.com

Abstract: -

In a given geometric objects, set of objects are retrieved as results for the given query. The problem is defined as geometric query intersection problem. The problem defines that, for the given user query, all the relevant results which are intersected by the query are showed as output. Because of the large number results, it occupies the huge memory space. For that here we propose a solution that we show the top-k results as the output where k is the integer.

This paper gives a general technique to solve any top-k GIQ problem efficiently. The technique relies only on the availability of an efficient solution for the underlying (non-top-k) GIQ problem, which is often the case. Using this, asymptotically efficient solutions are derived for several top-k GIQ problems, including top-k orthogonal and circular range search, point enclosure search, half space range search, etc. Implementations of some of these solutions, using practical data structures, show that they are quite efficient in practice. This paper also does a formal investigation of the hardness of the top-k GIQ problem, which reveals interesting connections between the top-k GIQ problem and the underlying (non-top-k) GIQ problem. In our proposed system we are going to perform multiple query processing. Because in the existing approach it can able to process only one query at the time. But in our proposed work we process multiple queries at the time.

Index Terms—

INTRODUCTION

In a Geometric Intersection Query (GIQ) problem, we are given a set, A, of n geometric objects (e.g., points, lines, hyper-rectangles line segments, balls, etc.) in some ambient space Rd. The goal is to organize A into a space-efficient data structure so that the following query can be answered efficiently: Given a query geometric object, q (e.g., a rectangle or ball), report or count the objects of A that are intersected by q. (In the reporting version, the output is a list of all objects of A that are intersected by q; in the counting version, the output is simply the number of objects of A intersected by q.) Typically, the data structure needs to support a very large number of queries (in the hundreds of thousands), so it is worthwhile to pre-process A beforehand to speed up the queries. The performance measures of primary interest are the space occupied by the data structure and the query time. It is also desirable that the data structure support updates (insertion/deletion) of objects in A. GIQ problems model many real-world query-retrieval problems arising in diverse domains, including GIS, spatial databases, VLSI design, robotics, CAD/CAM, and computer graphics. GIQ problems have been investigated extensively in the computational geometry and database literature and efficient solutions have been designed for many instances of these problems. In recent years, there has been an explosion in the volume of digital data that is being generated and stored, and this growth promises to continue unabated as computing and storage costs drop. Major sources of voluminous digital data include social networks (e.g., Facebook, Twitter), the world-wide web, mobile devices (e.g., smart phones), GIS and spatial applications, business analytics, remote sensing, video surveillance, genomics, high-energy physics, etc. This proliferation of digital data has made it imperative that new ways be developed to query large datasets and make sense of the results. Traditional query methods that simply report all the data items satisfying a query are no longer sufficient as they place an undue burden on the user to sift through a potentially huge answer space for relevant information. Instead, what is really needed is an appropriate summary of the query results that can provide the user with useful information. Creating such a summary involves applying a suitable aggregation function to the query results. A simple, yet effective, aggregation function is the so-called top-k function which returns a set of k objects that best satisfy a query, where the notion of "best" is based on a real-valued weight assigned by the application of interest to each data object. The top k problem has been well studied in various domains, including web search, information retrieval, data mining, business analytics, recommender systems, etc... Ilyas provide a comprehensive survey of the top-k problem.

In this paper, the system investigates the top-k problem in a geometric setting. Informally, we are given a set, A, of n geometric objects in Rd, where each object has an associated real-valued weight. Our goal is to pre-process A into a space-efficient data structure so that for any query geometric object, q, and positive integer, k, the k largestweight objects of A intersected by q can be reported efficiently. The system calls this problem a top-k Geometric Intersection Query (top-k GIQ) problem. It is a useful and non-trivial generalization of the traditional GIQ problem described above and has applications in GIS, spatial databases, VLSI design, scheduling, etc.

The most commonly accepted definition of "data mining" is the discovery of "models" for data. A "model," however, can be one of several things. We mention below the most important directions in modeling. Statisticians were the first to use the term "data mining." Originally, "data mining" or "data dredging" was a derogatory term referring to attempts to extract information that was not supported by the data. This paper illustrates the sort of errors one can make by trying to extract what really y isn't in the data. Today, "data mining" has taken on a positive meaning. Now, statisticians view data mining as the construction of a statistical model, that is, an underlying distribution from which the visible data is drawn. The system considers data in the form of a stream. The difference between a stream and a database is that the data in a stream is lost if you do not do something about it immediately. Important examples of streams are the streams of search queries at a search engine or clicks at a popular Web site. In this chapter, we see several of the surprising applications of hashing that make management of stream data feasible.

Data Mining: This term refers to the process of extracting useful models of data. Sometimes, a model can be a summary of the data, or it can be the set of most extreme features of the data. The system Introduces the market-basket model of data, and its canonical problems of association rules and finding frequent item sets. In the market basket model, data consists of a large collection of baskets, each of which contains a small set of items. We give a sequence of algorithms capable of finding all frequent pairs of items that is pairs of items that appear together in many baskets. Another sequence of algorithms is useful for finding most of the frequent item sets larger than pairs, with high efficiency.

II. RELATED WORK

Several traditional works carried out related to solve the k GIQ problem. They were discussed in this section detail.*G. Navarro and L. Russo* [1] studied various problems on two-dimensional grids that are relevant for data analysis, focusing on achieving good time performance (usually poly logarithmic) within the least possible space (even succinct for some problems). Other applications for such queries were frequently found in Geographic Information Systems (GIS), where the points have a geometric interpretation and the values city sizes, industrial production, topographic heights, and soon. On this set of queries the contribution is to achieve good time complexities within linear and even succinct space. This is relevant to handle large datasets in main memory. While the times we achieve are not competitive when using O(n) integers or more, the system manage to achieve poly logarithmic times within just n log n+o(n log n) bits on top of the bare coordinates and values, which we show is close to the information-theoretic minimum space necessary to represent n points. The system has demonstrated how wavelet trees can be used for solving a wide range of two-dimensional queries that are useful for various data analysis activities. More typical counting and reporting queries, but not for these less understood ones.Another interesting open problem was how to support dynamism while retaining time complexities

logarithmic in the number of points and not in the grid size.For the queries that follow we do not need the exact w (p) values, but just their relative order.

A. Yu, P. Agarwal [2] presented a geometric framework for the problem that allows us to describe the set of affected queries succinctly with messages that efficiently disseminated using content-driven networks. They gave fast algorithms to reformulate each update into a set of messages whose number is provably optimal, with or without knowing all user interests. They developed a geometric framework to support range top-k subscriptions. The geometric framework enables us to view the problem of generating notification messages intuitively as one of tiling a potentially complex region of affected subscriptions (in an appropriately defined subscription space) using simple geometric shapes. The set of tiles form a compact description of the region. Distributed database of objects across multiple nodes are not efficient. Also, the ring geometry does not necessarily needs such rigidity in neighbor selection and supporting more complex queries cannot be done easily in P2P systems, particularly in DHTs.

C. Sheng and Y. Tao [3] presented a fully dynamic structure that retains the same space and query bounds, and can be updated in amortized time per insertion and deletion. The top-K range reporting problem also studied in a variety of areas, e.g., information retrieval, data streams. The use of applications where people would like to identify the best few objects, among only a subset of the dataset satisfying a range predicate. Top-K range reporting without the distinct-score condition Points has distinct scores in the standard top-K range reporting problem. It is obvious that a single insertion in a point set can invalidate its entire logarithmic sketch, thus making the sketch expensive to maintain. The base tree is a Btree on the points of P, where each leaf node contains at least points, and each internal node has at least f/2 children. In this process the retaining the same space and query bound dynamic are not efficient in dynamic structure. As a direct corollary, our structure can be built from scratch in O ((N/B) log M/B (N/B)) time. We thus have completed the proof of Lemma. S. Rahul et al [4] presented an efficient geometric algorithms and data structures for two versions of the top-k problem. In the first version, the top k points can be reported in any order and the underlying set S can be updated through insertion and deletion of points. The top-k problem is a non-trivial generalization of the well-studied range search problem in computational geometry, where the goal is to preprocess S so that the points that lie in q can be reported or counted efficiently. Modern GIS systems allow users to query the underlying spatial dataset for useful information. In the preprocessing phase, all the points of S are sorted based on their weights in non-increasing order. In the query, we are given an orthogonal d-dimensional box and an integer k. The d-dimensional range reporting/counting structures can handle insertion and deletion of points of S.

T. Gagie et al [5] built and store the wavelet tree T for s and, at each internal node v, they store a small data structure that lets us perform time rank queries on v's binary string. The document listing problem is a variation on the classical pattern matching problem. Instead of returning all the positions at which a pattern P occured in the text T , they considered T as a collection of k documents concatenated and our task is to return the set of documents in which P occurs. A range quantile query takes a rank and the endpoints of a sublist and returns the number with that rank in that sub list.*Cheng Sheng et al*[6] Rectangle-intersection-max has also been studied in bi temporal databases and meteorology systems, whereas stabbing-max has been identified as a core operation in packet classification. Our complexity analysis is under the external memory (EM) model which has been successful in capturing the I/O characteristics of database algorithms. Various heuristic access methods are available for orthogonal range aggregation, and have been empirically shown to work well for selected data and query distributions. Our structure that is the BCB-tree only needs to consume space sub-linear in K, while guaranteeing logarithmic query cost. Because of the BCB tree construction it occupies huge amount of memory size. The same process does not apply to F=sum, for which our methods work for integer weights only. Hence anefficient Top-k Query Retrieval technique proposed in this paper to overcome the traditional problems.

III. PROPOSED METHOD

A general technique to solve any top-k GIQ problem efficiently is implemented using the block diagram as shown in fig. 1

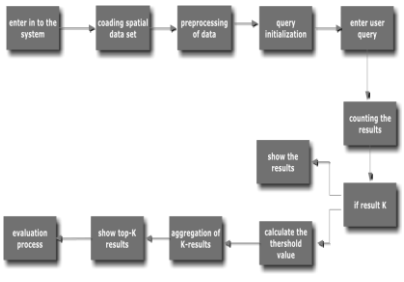

Fig. 1 Block Diagram of Proposed method. The process to eliminate the top-K GID problem implemented by using several modules.

A. Loading and preprocessing of data

In this module we are going to select the input spatial dataset. The input spatial dataset contains X-axis, Y-axis, fine fuel moisture code, duff moisture code, drought code, initial spread index etc. After that load, the spatial dataset which contains geometric relevant information. After loading, view the spatial data. After that we enter into the preprocessing step. In this process we remove the unwanted values like null, missing tuples etc. There are a number of data preprocessing techniques. Data cleaning can be applied to remove noise and correct inconsistencies in the data. Data integration merges

data from multiple sources into a coherent data store, such as a data warehouse. Data transformations, such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.

Data reduction can reduce the data size by aggregating, eliminating redundant features, or clustering, for instance .These techniques are not mutually exclusive; they may work together. For example, data cleaning can involve transformations to correct wrong data, such as by transforming all entries for a Date field to a common format. Data processing techniques, when applied before mining, can substantially improve the overall quality of the patterns mined and/or the time required for the actual mining as shown in fig. 2

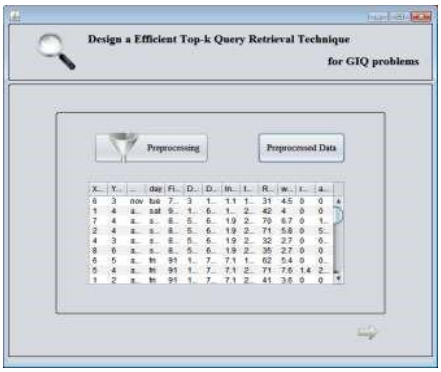

Fig. 2 Loading and preprocessing

B. Querying process

After preprocessing step, we get the preprocessed data. We display the cleaned data. After that we design a query processor model. In that query model, we get the user query. The query is based on the spatial results. The queryprocessing-based approach can also dramatically improve the energy efficiency—the typical measure of performance in sensor networks—of data-collection applications.

After a query has been disseminated, each node begins processing it. Processing is a simple loop: once per epoch, a special acquisition operator at each node acquires readings, or samples, from sensors corresponding to the fields or attributes referenced in the query.

The query processor routes this set of readings, or tuple, through the query plan built in the optimization phase. The plan consists of a number of operators applied in a fixed order. Each operator can pass the tuple to the next operator, reject it, or combine it with one or more other tuples as shown in fig. 3.

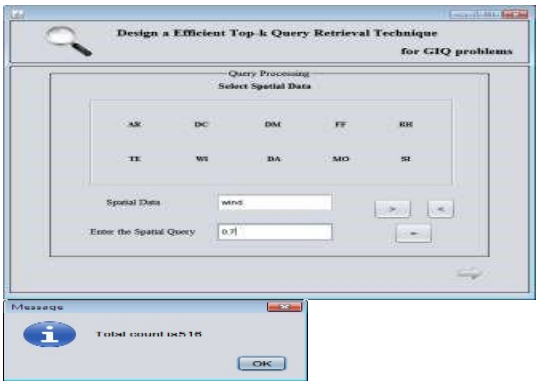

Fig. 3 Querying Process

A node transmits tuples that successfully pass the plan up the routing tree to the node's parent, which can, in turn, forward the result or combine it with its own data or data collected from other children.

C. Calculation of relevant results

During query processing, it first checks for the matching of results. After receiving all the matched results, we separately stored the collected results. In this module, to gather the top-k results we are going to calculate the count of the relevant results. If the relevant results are less than the K value, then we show all the relevant results. If the k values are not specified means we can show all the relevant results. We assign the real weights for all the relevant results as shown in fig. 4.

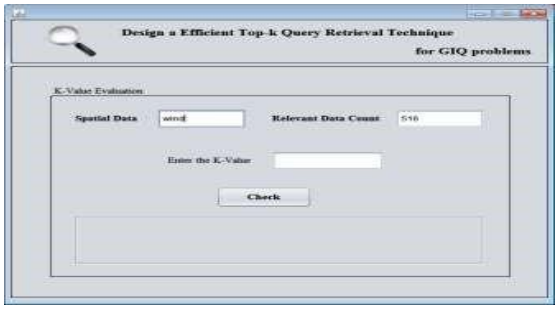

Fig. 4 Relevant Results

D. Implementation of query algorithm

In this module we are going to implement the query algorithm. The input is the group of relevant results for the given query. After that we perform checking process on K value. During the checking, if relevant results are less than the K value then we show all the results. If relevant results are greater than the k value, then we evaluate the threshold value based on the assign real weights to the data. Based on the calculated threshold value, we finally display the top-k results. A top-k query is an important type of query that allows for supporting IR applications on top of database systems. Query optimization is necessary for high level relational queries to provide an opportunity for the database system to systematically evaluate alternative query execution strategies and to choose an optimal strategy.

The function of query optimization is to select an efficient execution strategy. The query optimization process generates the best query execution plan. Query optimization is essential for large information retrieval system. In this paper we describe the top-k queries and general query optimization method for to provide an opportunity for the database system to systematically evaluate alternative query execution strategies and to choose an optimal strategy.

E. Evaluation process

In this module, we are going to evaluate the process. In the evaluation process, we show the accuracy of the result produced for that particular query. Another evaluation is based on the execution time Evaluation process is the systematic collection and analysis of information to make judgments about the effectiveness, efficiency, and appropriateness of a program or initiative.

Evaluation is a dynamic process that assists with the ongoing development and adaptation of programs to better suit the context in which they operate. Therefore, it is of benefit to policy advisors and program coordinators to incorporate an evaluation strategy in the early stages of program planning. A process evaluation begins by identifying the way in which the program was intended to be implemented and to operate. With respect to the way the program was intended to be implemented. The methods used to collect data about the way that a program was intended to operate are similar to those used to collect data about the way that a program actually operates.

An evaluation process shown in fig. 5 is a written document that describes how you will monitor and evaluate your program, as well as how you intend to use evaluation results for program improvement and decision making. Evaluation encourages us to examine the operations of a program, including which activities take place, who conducts the activities, and who is reached as a result. In addition, evaluation will show how faithfully the program adheres to implementation protocols.

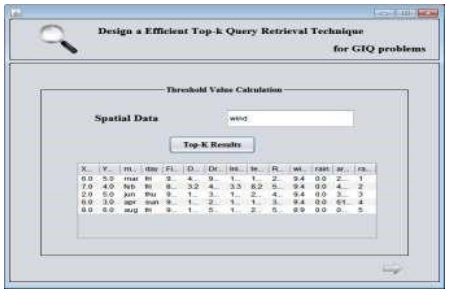

Fig. 5 Evaluation

IV. CONCLUSION

The system have given a general technique that solves any top-k GIQ problem efficiently, provided efficient solutions are available for the counting and reporting versions of the underlying GIQ problem. The system has shown how to use this to obtain asymptotically efficient solutions for several specific instances of top k GIQ problems. The system has also investigated the computational hardness of the top-k GIQ problem. The system have implemented our solution for some of the top-k GIQ problems discussed; using practical data structures (R-trees), and have found them to be quite efficient. The query time of Top-k ORS and Naive Scan increased with dataset cardinality or dimension, but at a rather slow rate. By comparison, the query time of Naive ORS grew at a much faster rate.

We also observe that the query time of Naive ORS degraded more rapidly with increasing dataset cardinality than with increasing dimension. One possible reason is that the performance of Naïve ORS is directly proportional to the number of points inside the query box, which increases with increase in cardinality; on the other hand since an R-tree is being used to retrieve all the points inside the query box efficiently, dataset dimension has less of an impact on query time.

The query time of the three solutions as a function of k on the forest fire and four square datasets. All the algorithms used the same set of 100 query boxes to query the dataset. We note that the query times for Naive ORS and Top-k ORS

do not vary much with the value of k. In the former case this might be attributable to the fact that the number of points in the query range dominates the value of k, while in the latter case the time to taken to identify the threshold point and the canonical nodes in T _ and to query the GIQ data structure at the canonical node.

REFERENCES

[1].G. Navarro and L. M. Russo, "Space-efficient data-analysis queries on grids," in Algorithms and Computation, ed: Springer, 2011, pp. 323-332.

[2].A. Yu, P. K. Agarwal, and J. Yang, “Processing and notifying range top-k subscriptions,” in Data Engineering (ICDE), 2012 IEEE 28th International Conference on, 2012, pp. 810-821.

[3].C. Sheng and Y. Tao, “Dynamic top-k range reporting in external memory,” in Proceedings of the 31st symposium on Principles of Database Systems, 2012, pp. 121-130.

[4].S. Rahul, P. Gupta, R. Janardan, and K. Rajan, “Efficient top-k queries for orthogonal ranges,"in WALCOM: Algorithms and Computation, ed: Springer, 2011, pp. 110-121.

[5].T. Gaige, S. J. Puglisi, and A. Turpin, “Range quantile queries: Another virtue of wavelet trees," in String Processing and Information Retrieval, 2009, pp. 1-6.

[6].C. Sheng and Y. Tao, "New results on two-dimensional orthogonal range aggregation in external memory,” in Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2011, pp. 129-139.