Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 2, Issue 2, 2016
ISSN: 2455 - 7528

Efficient processing of top-k spatial Boolean queries for distributed system

*Akash Porwal Dept. of Information Technology,*Suresh Gyan Vihar University, jaipurRaj. India akash.compound@gmail.com
Savita Shiwani HOD, Dept. of Information Technology Suresh Gyan Viahr University, jaipur Raj, India

Abstract— The widened use of spatial data imposes the heavy load on databases and data sizes. Now a day's spatial data is a part of smart computing because location is mostly responsible for user activities. In this paper we focus on reviewing a more efficient system to process spatial data in distributed system by implementing a combined approach for geospatial indexing and query formatting. Previous research suggests a positive improvement in efficiency of system through a new approach. A review of techniques for indexing and query processing is presented in this paper.

Keywords—Spatial data; GeoHash; GeoSpatial indexing; Boolean Query;

I. INTRODUCTION

Today, in the age of technology where the information is not a limited or costly resource, where you can find what you demand on just some clicks. The evolution of technology comes with both faces of coins, positive side is, our workforce and production efficiency increases and we became much smarter. The other face is, we depend on technology for assisted decisions but also helps us to take smarter decisions with the help of information. The prevailing trend of mobile computing gives an opportunity for area/location based services. The survey conducted in recent months shows that more and more users are using their mobile phones for almost all tasks then personnel computers. Increased access of internet through mobile also improves the reach of computing power in remote or rural areas of India. From the current scenario's one of the tech giant announced that the "company would now develop services "mobile first", meaning that the services are developed first for mobile devices and only then adopted to desktop devices and users" [1]. Location data on a mobile device provides various services as-

Finding routes- the services works between two geographical points on earth's surface and finding the minimum distance between points. The basic idea of finding minimum distance works for distance via air. But for navigation purposes a database containing all routes and public roads is used to execute query where distance between points at every turn is computed and stored in database. During a query execution the sum of the distance is shown to user.

Nearest Neighbor- Nearest Neighbor works around a user's location computed through different technologies. Nearest Neighbor queries takes input of user's location and text input (keywords) user willing to search. The accuracy and preciseness of results is based on two factors- (i) Accuracy of user's location- if the location of user is pinpointed then the system can find more nearest results in a circular area with least distance. (ii) The keywords also play a vital role in computing results.

e.g. – As if user search for the stores near his location who sells "shoes" and "jeans". Then, his search would look like "shoes and jeans" here we can see that the search processor takes "and" as keyword. But it needed to be taken as Boolean symbol.

Most importantly all these services depend on a set of data standardization for earth surface known as Euclidean space where every point on earth surface is provided a unique numerical value for the earth surface. A sphere, the most perfect spatial shape according to Pythagoreans, also an important concept in modern understanding of Euclidean spaces represented in fig. 1

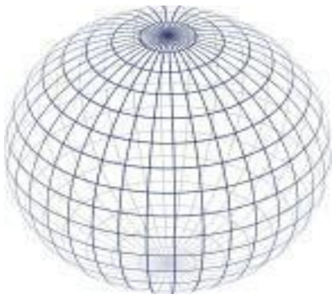
Fig.1 Sphere Wireframe

A. *LOCATION DATA*

Location data is the address of a point on earth surface in terms of coordinates. Coordinates of earth surface in terms of degree (angle) where of center of earth is considered as centre of plane. Most common choice of coordinate is latitude, longitude; sometimes elevation data can also be required.

- *Cartesian Coordinates*
  Cartesian coordinates can be explained as mapping of earth surface in the form of plane coordinate system(x y z) which helps to simplify mathematical calculations related to point mapping. The origin is center of mass earth.
- *Shape of earth*
  The earth is not a perfect sphere, but an irregular shape approximating a ellipsoid. The ellipsoidal effect is due to visible bulge at equatorial hence difference of radius is seen between the poles and equators. The shorter axis at poles approximately coincides with axis of rotation. Various systems have been developed and implemented at different technologies. The system used by GPS, WGS84, differs at Greenwich from the one used on published maps OSGB36 by approximately 112m. the military system ED50, used by NATO, differs by about 120m to 180m [3].

B. *SPATIAL DATABASES*

Spatial data is a set of coordinates which can be represented in different formats- where representation of direction is optional. Number of digits after the point represents the preciseness of location or pinpointed location. As spatial database and their objects are always related to each other and the nearest objects are more related than the farthest objects.

Representing the objects-

- *Point:[x:real, y: real]*
- *Node[point, <arc>]*
- *Arc: [node-start, node-end, <point>]*
- *Polygon: <point>*
- *Region: {polygon}*

Table 1 shows an example of record type instances. Two chains, respectively identified as TLID 8086 and 8087, feature the start (FR) and end (TO) nodes, in longitude (LONG) and latitude (LAT). For example, FRLAT stands for the latitude of the start node [2].

As soon as the chain is not a line segment, we need additional information on intermediate (shape) points. **Table 1** Example of record type (chains).

TLID	FRLONG	FRLAT	TOLONG	TOLAT
…	...	…	…	…
8086	-7215654	+41957498	-72161936	+41958117
8087	-7219712	+41957206	-72197197	+41957669
…	…	…	…	…

C. *SPATIAL DATASTRUCTURES*

We classify the geo-textual indices in to three different categories: the spatial indexing scheme used, the text index employed, and hybrid manner of the spatial index and the text index [4].

1) *Spatial Indexing scheme*

Considering the spatial indexing scheme used, we classify the indices into three categories, namely R-tree based indices, grid based indices, and space filling curved based indices.

- *R-tree based*- This category of indices use the R-tree [5] or a variation (e.g. the R*-tree [6]). Most geo-textual indexing is done using the same format and inverted file is used for text indexing.
- *Grid based*- Grid based indexing is a combination of grid indexing with a text index (e.g. the inverted file). These grid

indices divide space into a predefined number of equal-sized squares of rectangular cells. The grid index and the text index can be organized either separately or combined tightly.

- *Space filling curve based-* These indices combines files with a space filling curve, a Hilbert curve based and Z-curve based index is included. The indices are based on the property of distance proportionality where the distance between points in the native space is proportional to distance between points on the space filling curve.

2) ***Text indexing scheme***

- *Inverted file-* An inverted list is a list of vocabulary where every keyword is stored with an indexing of location of same keyword used in different locations.
- Bitmap- Some R-tree based indices use bitmaps to index the text information in subtrees. The presence or absence of a term is represented by bit in a bitmap. 0 represents the absence of terms where 1 represents the presence.

3) ***Combination Scheme***

During implementation geo-textual indices combine spatial and text indexing. The categorization of indices can be done through how they are combined namely text-first loose combination, spatial-first loose combination, and tight combination.

A text-first loose combination index implements the text indexing scheme (e.g. inverted list) as the top-level index and then the postings are arranged in inverted list in a spatial indexing scheme(e.g. R-tree, a grid or a spatial filling curve), vice versa in spatial-first loose combination. Where, as the tight combination index prunes the search space simultaneously during query processing.

D. HIERARICHIAL SPATIAL DATA STRUCTURE

GeoHash is a geocoding system for latitude and longitude. A geohash code, represented as a string, basically denotes a rectangle (enclosed area) on the earth. Spatial hierarchy (the precision of location based on the enclosed area) is defined on the basis of string length. The bigger length of string denotes less area covered. And hence can also be used to save space in database based on the precision.

E. SPATIAL KEYWORD QUERYING

Spatial keywords queries are now become the part of our daily life. As for Smartphone users spatial queries can be used with different applications. Many Google services are provided on the location basis like- weather forecast, nearest POI (point of interest- parking location, shopping centers, and restaurants). Three types of spatial keyword queries are frequently used- Boolean kNN query, the top-k kNN query, and the Boolean range query [4].

- **Boolean kNN Query:** “Retrieve the k objects nearest to the user’s location” such that object’s text description contains the keyword searched.
- **Top-k kNN Query:** Retrieve the k objects with highest ranking scores, measure as a combination of their distance to the query location and the relevance of their text description to the query keywords.
- **Boolean Range Query:** Retrieve all objects whose text description contains the keywords “searched” and whose location is within 10km of the query location.

II. LITERATURE REVIEW

In this section we will discuss the previous research done for the processing of Spatial Queries. Where, we study the different Geo-Textual indexing, different types of spatial queries and their Boolean processing in distributed storage system.

A. Geo-Textual Indexing

1) R-Tree Based Indices

- *IF-R* and R*-IF*

  The combination of R-tree and inverted file is one of the best combination to process spatial queries where IF-R* (inverted file- R- Tree) and R*-IF(R-tree –inverted file) is the loosely coupled geo-textual indexing [8]. Both of them were designed to generate results for spatial query in a pre-specified region i.e. BRQ (Boolean range query).

However the results show that it cannot be effectively used for T*k*Q (Top-*k* query [4].

- *KR*-Tree*
  KR*-Tree (keyword R*-Tree) is a structure where nodes of the tree are augmented in the inverted list as objects which helps to prune the tree nodes which do not contain the query keywords [9].
  The approach is proposed for BRQ and cannot effectively work for T*k*Q [4].
- IR^2*-Tree*
  IR^2 -Tree contains a signature file attached to every node of tree in the form of bitmap which stores the fan-out of the tree [10].
  It is efficient to process BRQ but cannot process T*k*Q because of unavailability of frequency information
- *Hybrid Spatial-Keyword Indexing(SKI)*
  SKI works on the combination of R-tree and bitmaps where every super node of R-tree stores the bitmap in the form of inverted file. SKI is similar to R*-IF however, the SKI uses the bitmap version of inverted file where as R*-IF uses the original inverted file [7].
- *IR-tree index and its variants*
  Combination of inverted file and R-Tree where every node of tree holds a pointer to the list of vocabulary which is a inverted list. IR-Tree can process all three types of queries, namely, BRQ, B*k*Q (Boolean *k*- result query*)*, and *Tk*Q.
- *WIR-Tree*
  WIR-Tree is also a variant of IR-Tree. It works on finding the list of words which helps to prune lesser number of nodes. It requires an approach to divide the list of words in two parts where list one contain most frequent where the second list contains the less frequent words. Then again the list is divided in two parts based on their frequency and this process is iterative until the list contains only a certain number of words. The final list is applied on the nodes of tree where an improved version WIBR tree. Here, the bitmaps used to count the frequency of word in near future [11].
- *Spatial Inverted Index(S2I)*
  S2I is based on the R-Tree and inverted file which implements different techniques to partition the list of frequent and infrequent terms [12].
  It is originally designed for T*k*Q but can also be used for B*k*Q and BRQ [4].

2) ***Grid Based Spatial-Textual Indices***
   - *ST and TS*
     ST(Spatial-first text) and TS(Text-first Spatial) are the simplest Grid based indices based on the loosely coupled combination scheme [13]. TS is more efficient than ST and hence only TS is used. TS is primarily designed for BRQ and cannot be used for B*k*Q and T*k*Q.
   - *Spatial-Keyword Inverted File(SKIF)*
     SKIF is quite different approach to implement the query because it uses an inverted list for both spatial and text data. It considers every point as the region. It is basically designed to process a query different from all three types but the query looks similar to BRQ and SKIF can be used on BRQ [14].
3) *SFC-QUAD*
   Several hybrid approaches were proposed combining the space filling curve and inverted file. In which SFC-QUAD is seen to perform best. SFC-QUAD inverted list contains the docIDs and frequencies of objects [15].

B. *Spatial Queries*

There are different types of spatial queries, based on the need like k Nearest Neighbor query, Top-k kNN spatial Boolean query and Boolean range query for different applications. We focus on selected query where we have to process Boolean algebra efficiently for précised results like Top-k kNN spatial Boolean query. Top-k queries are used to find the matching results for user's keyword as in database thousands of objects exist for the same keyword but the Boolean processing between keywords helps to choose objects from different data space which suits more to user requirement.

Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 2, Issue 2, 2016
ISSN: 2455 - 7528

C. *Spatial Query Processing*

In hierarchal spatial data structure (GeoHash) containing location information stored in the form of String. Data is stored in the form of objects where a spatial object includes its geometry and can have any information about the object about its name, address. GeoHash- our spatial index supports generally all geometries including points, lines, rectangles, curves and polygons. To store a spatial object in database and to be indexed we first calculate a set of minimum bounding boxes *geo-hash set*, which fully cover the geometry of spatial object. Generation of too many bounding boxes to define the object and increased overhead can be controlled through graining of rectangular boxes and hence defining the maximum number of bounding boxes for each geometry is important. The number of bounding boxes can be implemented differently for different applications. Different geohash codes length does not accompany a precision in geometry.

Our spatial query processing consists of two steps: filter step and refinement step. In the filter step, we find spatial objects, which satisfy the query condition, by pruning non-qualifying spatial objects. In next step, we examine each candidate spatial objects to determine whether the object is actually satisfying the query condition.

The definition of geohash shows longer geohash codes will generate smaller geometries. If a local area is well developed and have many objects stored in the database with their geometries then a small area can occupy large number of rows in database which affects the performance of our spatial query processing.

To improve performance of processing we consider a large area or fix the length of string generated through geohash where many rows contain the same string / rectangular box helps to save space and improve performance but affects the location precision.

To execute spatial queries in geohash we work on finding only relevant objects related to query by calculating the minimum geohash set and then finding the relative context of object.

Boolean keyword querying

Consider a spatial database *D* = { $o_1, o_2, o_3 \dots, o_n$ } is a set of objects such that every

o ∈ *D* has a pair of attributes < *p* , *T*>, where *p* ∈ *E* is a point in a metric space *E* with distance dist(p_1, p_2), and *T* = {t_1 , t_2 ....} is a document as a set of terms.

A *Top-k* spatial Boolean (*k-SB)* query Q is a triple <*l, k, B*>, where *l* ∈ *E* is the query location (*spatial constraint*), *k* is the *desired output size and B is the conjunctive Boolean predicate* (text constraint). *B* is a set of keywords prefixed with Boolean operators { ∧, ∨, ¬}, conjunctively connected as follows:

$$B = [\wedge(A = \{ a_1, a_2 \dots\} \})\wedge\vee(C = \{c_1, c_2 \dots\} \})\wedge\neg (G = \{g_1, g_2 \dots\} \})] \quad (1)$$

A (*AND*-semantics), *C* (*OR*-semantics), *G* (*NOT*-semantics) are subsets of terms prefixed with ∧, ∨, and¬, respectively. An object *o* ∈ *D* satisfies *B* if:

$$[(\forall a \in A : o.T \cap a \neq \emptyset)\wedge(\exists c \in C : o.T \cap c \neq \emptyset)\wedge(\forall g \in G : o.T \cap g = \emptyset)] \quad (2)$$

The result of the k-SB query Q is the list:

$$L = \{o_i \in D, i = 1 \dots n_L \mid o_i \text{ satisfies } B \wedge n_L < k\} \text{ such that:}$$

$$\forall o \in (D \setminus L): [dist (o.p, l) \geq arg \quad dist (r.p, l) \vee \neg (o \text{ satisfies } B)] \quad (3)$$

Objects in *L* are sorted by distance to *l* in decreasing order. In other words, a *k-SB* query Q returns the *k* nearest neighbor objects to the query location *l* that satisfies the conjunctive Boolean predicate B. In this work, we assume E is the Euclidean space. The problem is how to efficiently compute *L* [7].

D. *Distributed Storage Systems*

A large number of non-relational distributed databases are available in the market which are not based on the SQL and are used in big data applications and analytics because they are designed for large data with data replication enabled with fault tolerance. These databases are also known as NOSQL databases because they are schema-less or key-value store.

III. CONCLUSIONS AND FUTURE WORK

All the above techniques and methods implementation focuses on efficient processing of location based services. The techniques we have studied help us to understand query processing in spatial databases, Geo-Textual indexing, implementation of Boolean algebra and Hierarchical data structure- GeoHash for spatial data.

A lot of research is still in process to improve efficiency. We also suggest an approach for distributed systems where Top-*k* Boolean queries are being processed on GeoHash where GeoHash can help by saving space and query processing time

Suresh Gyan Vihar University, Jaipur
International Journal of Converging Technologies and Management (IJCTM)
Volume 2, Issue 2, 2016
ISSN: 2455 - 7528

by reducing the number of rows. On the other hand, Boolean processing helps by improving the preciseness and context of result set.

References

[1] Cao, X. e. (2012). "Spatial keyword querying" Conceptual Modeling. Springer Berlin Heidelberg, 16-29.
[2] Rigaux, P. M. (2001). Spatial Databases: with application to GIS.
[3] A Guide to coordinate system in Great Britain. (2007).
[4] Chen, L.e.(2013). "Spatial keyword query processing: an experimental evaluation". Proceedings of VLDB Endowment.
[5] A.Guttman. (1984). R-trees: Adynamic index structure for spatial searching. SIGMOD.
[6] N. Beckmann, H.-P. k. (1990). The R*-tree: An efficient and robust access method for points and rectangles. SIGMOD.
[7] Cary, A., Wolfson, O., & Rishe, N. (2010, January). "Efficient and scalable method for processing top-k spatial Boolean queries". In Scientific and Statistical Database Management. (pp. 87-95) Springer Berlin Heidelberg.
[8] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. (2005). Hybrid index structures for location-based web search. In CIKM, pages 155–162.
[9] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. (2007). Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In SSDBM, page 16.
[10] I.D. Felipe, V. Hristides, and N.Rishe. (2008). Keyword search on spatial databases. In ICDE, pages 656-665.
[11] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. (2012). Joint top-k spatial keyword query processing. IEEE TKDE, 24(10):1889–1903.
[12] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørv˚ag. (2011). Efficient processing of top-k spatial keyword queries. In SSTD, pages 205–222.
[13] S. Vaid, C. B. Jones, H. Joho, and M. Sanderson. (2005). Spatio-textual indexing for geographical search on the web. In SSTD, pages 218–235.
[14] A. Khodaei, C. Shahabi, and C. Li. (2010). Hybrid indexing and seamless ranking of spatial and textual features of web documents. In DEXA, pages 450–466.
[15] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel. (2011). Text vs. space: efficient geo-search query processing. In CIKM, pages 423–432.