IR-Tree Query Processing in Spatial Database

K. Prabhakar
Assistant professor, Dept. of Information Technology, KITS-Warangal, 506015, Telangana, India

A. Devender
Assistant professor , Dept. of Information Technology, KITS-Warangal,506015,Telangana, India

Abstract— Standard spatial inquiries, such because range search and local neighbour access, involve simply conditions in objects' geometric properties. Today, many modern-day applications necessitate novel varieties of queries that seek to find things satisfying both equally a spatial predicate, as well as a predicate on the associated scrolls. For example, instead connected with considering all of the restaurants, a local neighbour question would instead ask for the restaurant this is the closest those types of whose selections contain "steak, spaghetti, brandy" all at the same time. Currently, the best solution to such queries will depend on the IR2-tree, which, as shown on this paper, incorporates a few deficiencies that severely impact it's efficiency. Determined by this kind of, this paper produce a new accessibility method termed the spatial inverted index in which extends the standard inverted index to cope with multidimensional information, and is sold with algorithms that could answer local neighbour inquiries with keywords in real time.

Keywords— Spatial Indexing, Neighbour Search

I. INTRODUCTION

Some sort of spatial data bank manages multidimensional things (such as points, rectangles, and so on.), and offers fast usage of those objects determined by different collection criteria. The benefit of spatial databases is reflected because of the convenience involving modelling agencies of reality within a geometric manner. For case, locations involving restaurants, lodges, hospitals and the like are frequently represented as points within a map, while more substantial extents for example parks, seas, and landscapes often as a combination of rectangles.

Many functionalities of the spatial database are helpful in other ways in specific contexts. For instance, in any geography info system, range search may be deployed to discover all restaurants within a certain area, while local neighbour access can find the restaurant closest with a given tackle. Today, the widespread by using search engines has caused it to be realistic to post spatial requests in a fresh way. Conventionally, queries concentrate on objects' geometric properties only, for example whether a point is within a rectangle, or the way close a couple points usually are from 1 another. This paper have experienced some modern applications that involve a chance to select objects determined by both of their geometric coordinates along with their related texts. One example is, it could be fairly practical if the search engines can be used to find the nearest restaurant that has "steak, spaghetti, and brandy" all simultaneously. Note that this may not be the "globally" local restaurant, even so the nearest eating place among only those providing all of the demanded foods and beverages. There usually are easy ways to support requests that incorporate spatial along with text attributes. The key drawback of those straightforward approaches is that they'll fail to deliver real period answers about difficult inputs. Spatial requests with keywords haven't been broadly explored. In the past years, town has started enthusiasm in studying search term search in relational databases.

In that paper, this paper design any variant involving inverted index that's optimized intended for multidimensional points, and is thus named the spatial inverted index (SI-index). This gain access to method successfully incorporates position coordinates right into a conventional inverted index having small added space, owing to a sensitive compact storage scheme. At the same time, an SI-index maintains the spatial surrounding area of data points, and possesses an R-tree built on just about every inverted listing at minor space cost to do business. As an end result, it presents two contending ways intended for query digesting. This paper can (sequentially) mix multiple lists like merging classic inverted provides by ids. Otherwise, this paper may also leverage the R-trees to look at points coming from all relevant provides in climbing order of their distances towards the query position. As shown by studies, the SI-index substantially outperforms the IR2-tree in query proficiency, often by the factor involving orders involving magnitude.

II. PREVIOUS WORK

Relational databases are commonly searched using structured query languages. The user needs to know the data schema to be able to ask suitable queries. Search engines on the Web have popularized an alternative unstructured querying and browsing paradigm that is simple and user-friendly. Users type in keywords and follow hyperlinks to navigate from one document to the other. No knowledge of schema is needed.

With the growth of the World Wide Web, there has been a rapid increase in the number of users who need to access online databases without having a detailed knowledge of schema or query languages; even relatively simple query languages designed for non-experts are too complicated for such users. Query languages for semi-structured/XML data are even more complex, increasing the impedance mismatch further.

Unfortunately, keyword search techniques used for locating information from collections of (Web) documents cannot be used on data stored in databases. In relational databases, information needed to answer a keyword query is often split across the tables/tuples, due to normalization. As an example consider a bibliographic database shown in Figure below. This database contains paper titles, their authors and citations extracted from the DBLP repository. The schema is shown in Figure below. Figure below shows a fragment of the DBLP database. It depicts partial information paper title and authors about a particular paper. As we can see, the information is distributed across seven tuples related through foreign key references. A user looking for this paper may use queries like ”sunita temporal” or ”soumen sunita”. In keyword based search, we need to identify tuples containing the keywords and ascertain their proximity through links.

In general, the importance of a link depends upon the type of the link i.e. what relations it connects and on its semantics; for example, in the bibliographic database, the link between the Paper table and the Writes table is seen as a stronger link than the link between the Paper table and the Cites table. The link between Paper and Cites tables would have a higher weight. The weight of a tree is proportional to the total of its edge weights, and the relevance of a tree is inversely related to its weight.

The example illustrates that some links point towards the root of the tree, instead of away from the root as required by our model. For instance, the Writes relation has foreign keys to the Paper and Author relations, whereas we require paths from Paper to Author, traversing a foreign key edge in the opposite direction. However, we cannot simply regard the edges as undirected.

Ignoring directionality would cause problems because of “hubs” which are connected to a large numbers of nodes. For example, in a university database a department with a large number of faculty and students would act as a hub. The backward edges ensure that there is a directed tree rooted at the paper, with a path to each leaf. To illustrate the effect of backward edge weights, let us return to the university department example. A forward edge from a student to her department and a back edge from the department to another student would form a path between each pair of students in the department. If there are more students in a department, the back edges would be assigned a higher weight, resulting in lower proximity (due to the department) for each pair of students, than if there are fewer students registered. In contrast, in the bibliographic database, papers (typically) have smaller numbers of authors, and the backward edge weights from Paper to Writes nodes would be less resulting in higher proximity between co-authors.

We may restrict the information node to be from a selected set of nodes of the graph; for example, we may exclude the nodes corresponding to the tuples from a specified set of relations, such as Writes, which we believe are not meaningful root nodes. In the example from Figure above, let the keyword nodes be SunitaS, SoumenC and ByronD. These nodes, which are author nodes, have a relationship induced due to paper node ChakrabartiSD98. The tree shown in Figure above (with backward edges from the Paper node to the Writes nodes) would be a connection tree for the keyword nodes, with the paper node as the information node.

We incorporate another interesting feature, namely node weights, inspired by prestige rankings such as PageRank in Google. With this feature, nodes that have multiple pointers to them get a higher prestige. In our current implementation we set the node prestige to the indegree of the node. Higher node weight corresponds to higher prestige. E.g., in a bibliography database containing citation information, if the user gives a query Query Optimization our technique would give higher prestige to the papers with more citations. As another example, in a TPCD database storing information about parts, suppliers, customers and orders, the orders information contains references to parts, suppliers and customers. As a result, if a query matches two parts (or suppliers, or customers) the one with more orders would get a higher prestige. [1]

Existing work treats objects as independent when ranking them for a given query. However, spatial web objects are not independent. A relevant object whose nearby objects are also relevant to the query is preferable when compared to a relevant object without relevant nearby objects. One reason is that if the object a user chooses to visit does not work out then there are other nearby relevant objects. Another is that a user may intend to visit several objects (e.g., to compare prices). For example, a user may prefer to visit a location with many restaurants or shops instead of a location with only one restaurant or shop.

A preference for clusters of relevant objects may explain the phenomenon that similar businesses tend to co-locate. For example, car dealerships tend to co-locate. We speculate that they benefit from the spatial proximity: together, they attract more customers to such an extent that this compensates for the increased competition. It is the objective of this paper to support this phenomenon in spatial web search. This is done by developing a

notion of object "prestige" that takes into account the presence of nearby objects that are also relevant to a query. This notion of prestige is then used for the ranking of the query results. [4] With the proliferation of geo-positioning, e.g., by means of GPS or systems that exploit the wireless communication infrastructure, accurate user location is increasingly available. Similarly, increasing numbers of objects are available on the web that have an associated geographical location and textual description. Such spatial web objects include stores, tourist attractions, restaurants, hotels, and businesses.

This development gives prominence to spatial keyword queries. A typical such query takes a location and a set of keywords as arguments and returns the single spatial web object that best matches these arguments. We observe that user needs may exist that are not easily satisfied by a single object, but where groups of objects may combine to meet the user needs. Put differently, the objects in a group collectively meet the user needs. For example, a tourist may have particular shopping, dining, and accommodation needs that may best be met by several spatial web objects. As another example, a user may wish to set up a paper consortium of partners within a certain spatial proximity that combine to offer the capabilities required for the successful execution of the paper.[5]

Historical background Bloom filters yield an extremely compact data structure that supports membership queries to a set. Their space requirements fall significantly below the information theoretic lower bounds for error-free data structures. They achieve their efficiency at the cost of a small false positive rate (items not in the set have a small constant probability of being listed as in the set), but have no false negatives (items in the set are always recognized as being in the set). Bloom filters are widely used in practice when storage is at a premium and an occasional false positive is tolerable. They have many uses in networks: for collaborating in overlay and peer-to-peer networks, resource routing, packet routing, and measurement infrastructures. Bloom filters are used in distributed databases to support iceberg queries, differential files access, and to compute joins and semi joins. Bloom filters are also used for approximating membership checking of password data structures, web caching, and spell checking.

Several variants of Bloom filters have been proposed. Attenuated Bloom filters use arrays of Bloom filters to store shortest path distance information. Spectral Bloom filters extend the data structure to support estimates of frequencies. In Counting Bloom Filters each entry in the filter need not be a single bit but rather a small counter. Insertions and deletions to the filter increment or decrement the counters respectively. When the filter is intended to be passed as a message, compressed Bloom filters may be used instead, where parameters can be adjusted to the desired tradeoff between size and false-positive rate.[6]

III. PROPOSED SYSTEM

A. Construction and Inverted Indexes

In this module construction of the spatial data is performed where let P be a set of multidimensional points. As our goal is to combine keyword search with the existing location-finding services on facilities such as places. We will assume that the points in P have integer coordinates, such that each coordinate ranges in 0 - t, where t is a large integer. Therefore, we could as well convert everything to integers with proper scaling. Inverted indexes have proved to be an effective access method for keyword-based document retrieval. In the spatial context, nothing prevents us from treating the text description Wp of a point p as a document, and then, building an I-index. Each word in the vocabulary has an inverted list, enumerating the ids of the points that have the word in their documents. Note that the list of each word maintains a sorted order of point ids, which provides considerable convenience in query processing by allowing an efficient merge step. In NN processing with IR-tree, a point retrieved from the index must be verified i.e., having its text description loaded and checked. Verification is also necessary with I-index, but for exactly the opposite reason. For IR-tree, verification is because we do not have the detailed texts of a point, while for I-index, it is because we do not have the coordinates. Specifically, given an NN query q with keyword set Wq, the query algorithm of I-index first retrieves by merging the set Pq of all points that have all the keywords of Wq, and then, performs to get the coordinates of each point in Pq in order to evaluate its distance to q.

B. Merging and Distance Browsing

In this module since verification is the performance bottleneck, we try to avoid it. There is a simple way to do so in an I-index: one only needs to store the coordinates of each point together with each of its appearances in the inverted lists. The presence of coordinates in the inverted lists naturally motivates the creation of an R-tree on each list indexing the points. Next, we perform keyword-based nearest neighbor search with such a combined structure. The R-trees allow us to remedy an awkwardness in the way NN queries are processed with an I-index. Recall that, to answer a query, currently we have to first get all the points carrying all the query words in Wq by merging several lists one for each word in Wq. This appears to be unreasonable if the point, say p, of the final

result lies fairly close to the query point q. This would become a reality if we could browse the lists synchronously by distances as opposed to by ids. In particular, as long as we could access the points of all lists in ascending order of their distances to q, such a p would be easily discovered as its copies in all the lists would definitely emerge consecutively in our access order. Distance browsing is easy with R-trees. In fact, the best-first algorithm is exactly designed to output data points in ascending order of their distances to q. However, we must coordinate the execution of best-first on R-trees to obtain a global access order. This can be easily achieved by, for example, at each step taking a peek at the next point to be returned from each tree, and output the one that should come next globally. This algorithm is expected to work well if the query keyword set Wq is small. For sizable Wq, the large number of random accesses it performs may overwhelm all the gains over the sequential algorithm with merging.

C. Inverted List Compression

In this module compression is performed which is widely used to reduce the size of an inverted index in the conventional context where each inverted list contains only ids. In that case, an effective approach is to record the gaps between consecutive ids, as opposed to the precise ids. For example, given a set S of integers, the gap-keeping approach will store integers, where the ith value (i >= 2) is the difference between the ith and (i -1) th values in the original S. As the original S can be precisely reconstructed, no information is lost. The only overhead is that decompression incurs extra computation cost, but such cost is negligible compared to the overhead. Note that gap-keeping will be much less beneficial if the integers of S are not in a sorted order. This is because the space saving comes from the hope that gaps would be much smaller than the original values and hence could be represented with fewer bits. Compressing an SI-index is less straightforward. The difference here is that each element of a list, a point p, is a triplet (idp, xp, yp), including both the id and coordinates of p. As gap-keeping requires a sorted order, it can be applied on only one attribute of the triplet. For example, if we decide to sort the list by ids, gap-keeping on ids may lead to good space saving, but its application on the x- and y-coordinates would not have much effect.

D. Building R-Trees

In this module R-Trees are built, remember that an SI-index is no more than a compressed version of an ordinary inverted index with coordinates embedded, and hence, can be queried in the same way, i.e., by merging several inverted lists. The R-Trees, allow us to process a query by distance browsing, which is efficient when the query keyword set Wq is small. Our goal is to let each block of an inverted list be directly a leaf node in the R-tree. This is in contrast to the alternative approach of building an R-tree that shares nothing with the inverted list, which wastes space by duplicating each point in the inverted list. Furthermore, our goal is to offer two search strategies simultaneously: merging and distance browsing. As before, merging demands that points of all lists should be ordered following the same principle. This is not a problem because our design in the previous section has laid down such a principle: ascending order of Z-values. Moreover, this ordering has a crucial property that conventional id-based ordering lacks: preservation of spatial proximity. The property makes it possible to build good R-trees without destroying the Z-value ordering of any list. Specifically, we can group consecutive points of a list into MBRs, and incorporate all MBRs into an R-tree.

IV. RESULTS

The concept of this paper is implemented and different results are shown below, The proposed paper is implemented in Java technology on a Pentium-IV PC with minimum 20 GB hard-disk and 1GB RAM. The propose paper's concepts shows efficient results and has been efficiently tested on different Datasets.

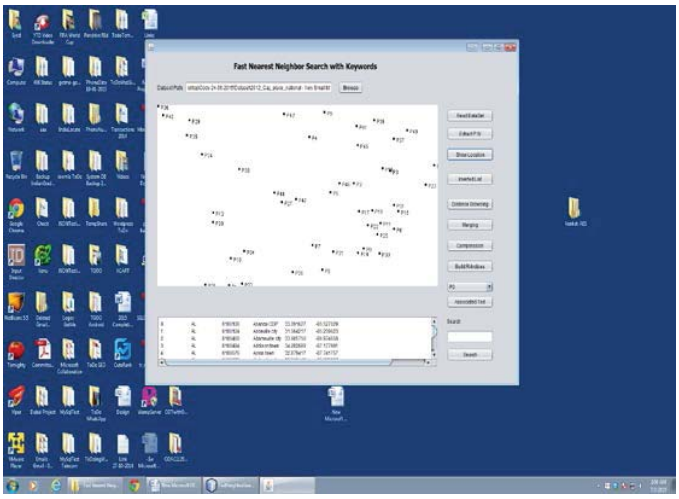

Figure 1: Displaying Locations of Extracted Points

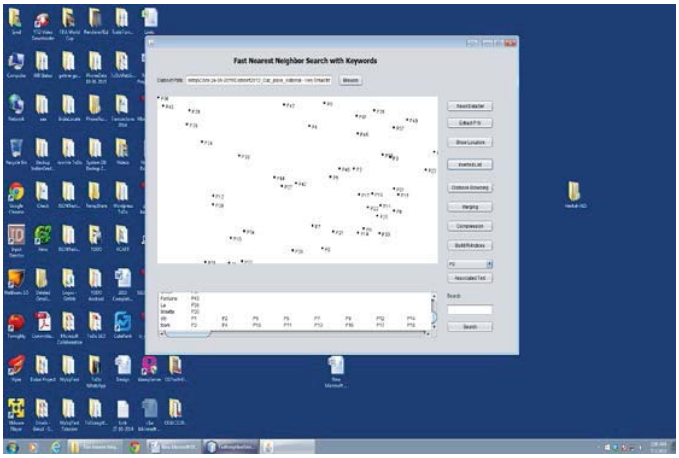

Figure 1: Computing Inverted Indexing

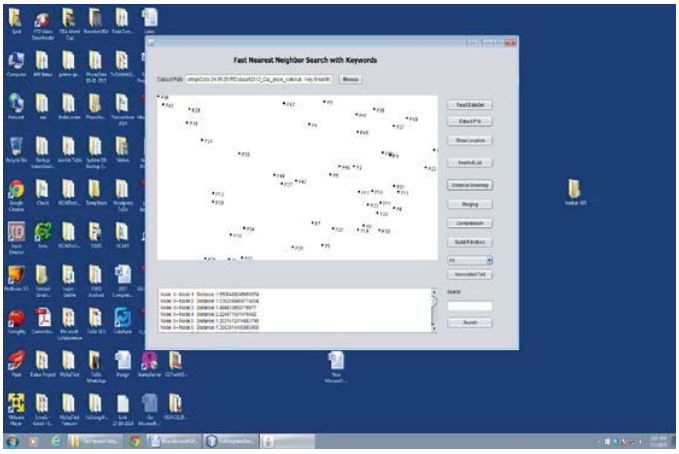

Figure 3: Computing Distance Browsing and Merging

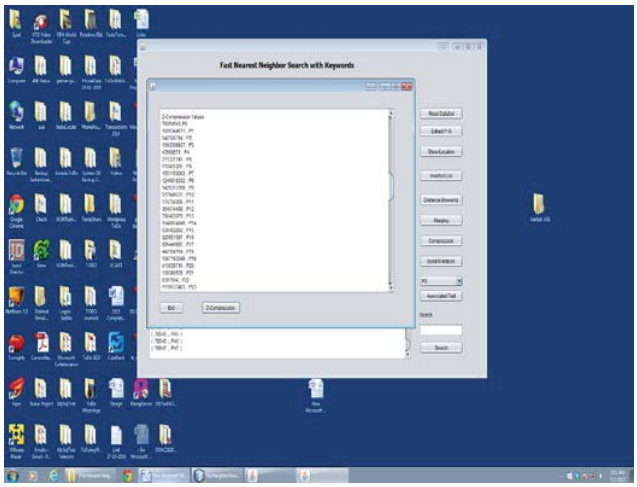

Figure 4: Computing Z-Compression

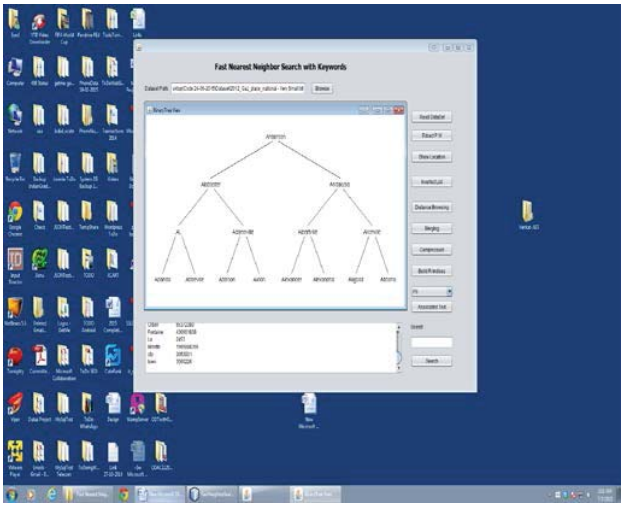

Figure 5: Displaying Sub IR-Tree

V. CONCLUSIONS

We now have seen plenty of applications getting in touch with for a search engine that has the ability to efficiently service novel sorts of spatial queries which have been integrated along with keyword seek. The existing solutions to such queries either incur too high space usage or can't give real-time answers. On this paper, we have now remedied the matter by establishing an gain access to method termed the spatial upside down index (SI-index). Not just that the SI-index is reasonably space affordable, but also it has the ability to perform keyword-augmented nearby neighbour search soon enough that is in the order of a multitude of milli-seconds. Moreover, as the actual SI-index will depend on the standard technology associated with inverted list, it is usually readily incorporable in a commercial search results that is true massive parallelism, implying its immediate manufacturing merits.

REFERENCES

[1] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, “Keyword Searching and Browsing in Databases Using Banks,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 431-440, 2002.

[2] X. Cao, L. Chen, G. Cong, C.S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M.L. Yiu, “Spatial Keyword Querying,” Proc. 31st Int’l Conf. Conceptual Modeling (ER), pp. 16-29, 2012.

[3] S. Agrawal, S. Chaudhuri, and G. Das, “Dbxplorer: A System for Keyword-Based Search over Relational Databases,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 5-16, 2002.

[4] X. Cao, G. Cong, and C.S. Jensen, “Retrieving Top-k Prestige- Based Relevant Spatial Web Objects,” Proc. VLDB Endowment, vol. 3, no. 1, pp. 373-384, 2010.

[5] X. Cao, G. Cong, C.S. Jensen, and B.C. Ooi, "Collective Spatial Keyword Querying," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 373-384, 2011.
[6] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal, "The Bloomier Filter: An Efficient Data Structure for Static Support Lookup Tables," Proc. Ann. ACM-SIAM Symp. Discrete Algorithms (SODA), pp. 30- 39, 2004.
[7] Y.-Y. Chen, T. Suel, and A. Markowetz, "Efficient Query Processing in Geographic Web Search Engines," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 277-288, 2006.
[8] E. Chu, A. Baid, X. Chai, A. Doan, and J. Naughton, "Combining Keyword Search and Forms for Ad Hoc Querying of Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2009.
[9] G. Cong, C.S. Jensen, and D. Wu, "Efficient Retrieval of the Top-k Most Relevant Spatial Web Objects," PVLDB, vol. 2, no. 1, pp. 337- 348, 2009.
[10] I.D. Felipe, V. Hristidis, and N. Rishe, "Keyword Search on Spatial Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 656-665, 2008.
[11] V. Hristidis and Y. Papakonstantinou, "Discover: Keyword Search in Relational Databases," Proc. Very Large Data Bases (VLDB), pp. 670-681, 2002.