RESEARCH PAPER | Management | Volume : 2 | Issue : 3 | December 2012 | ISSN - 2249-555X

Data Mining for Moving Object Data

KEYWORDS: Data mining, moving object data

Mr. Kalpesh R. Rakholiya
Ass. Pro. (HOD) Shri patel kelavani mandal college of technology junagadh

Dr. Dhaval Kathiriya
IT Director,University Bhavan,
Anand Agriculture University, Anand

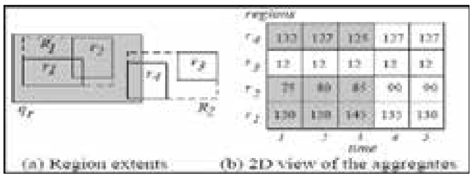
ABSTRACT *It is easy to observe that the number of moving objects in moving objects databases like those used in transportation systems, or air traffic control centers may be very large. To achieve an acceptable level of performance with such large volumes of continuously changing data, in answering moving object queries, it is not desirable to examine the location of each moving object in the database. Indexing the location attribute is hence necessary. The widely used mechanisms for indexing spatial data, like R Trees, MVB Trees, and Quad Trees etc would not the serve the purpose well since the data in spatio-temporal applications have to be continuously updated. Movement of a point object represents the trajectory of the moving point object. Data is typically treated as a set of line segments that collectively describe the trajectory of a moving object in the database. One simplifying approach suggested in [1] is to consider indexing structures to be appendonly with respect to time.This means,data grows mainly in the temporaldimension.*

THE FM ALGORITHM AND COUNTING SKETCH

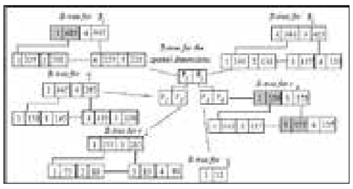
Estimating the number of distinct objects in a dataset has received considerable attention. Many methods in the literature are based on the FM algorithm developed by Flajolet and Martin [FM85]. FM requires a hash function h which takes as input an object id o, and outputs a pseudorandom integer h(o) with a geometric distribution, that is, Prob[h(o)=v] = 2-v for v>=1. A sketch consists of r bits, whose initial values are set to 0. For every object o in the dataset, FM sets the h(o)-th bit (of the sketch) to 1. After processing all objects, FM finds the first bit of the sketch that is still 0.

THE aRB TREE

The aRB-tree facilitates aggregate processing by eliminating the need to descend nodes that are totally enclosed by the query. As an example, consider the query in Figure (with interval qt=[1,3]). Search starts from the root of the R-tree. Entry R1 is totally contained inside the query window and the corresponding Btree is retrieved. Since the entries of the root node in this B-tree contain the aggregate data of interval [1,3] (and [4,5]), the next level of the B-tree does not need to be accessed and the contribution of R1 (i.e., the contribution of r1, r2) to the query result is 685. The second root entry R2 of the R-tree partially overlaps the query rectangle qr; hence, the algorithm visits its child node, where only entry r4 intersects qs, and thus its B-tree is retrieved. The first root entry suggests that the contribution of r4 for interval [1,2] is 259. In order to complete the result, we have to descend the second entry and retrieve the aggregate value of r4 for timestamp 3 (i.e., 125). The total number of objects in these regions during the interval [1,3] is the sum 685+259+125. Nevertheless, the aRB-tree does not take into account multiple object occurrences. Therefore, aRB-trees are not directly applicable for applications that require distinct counting.



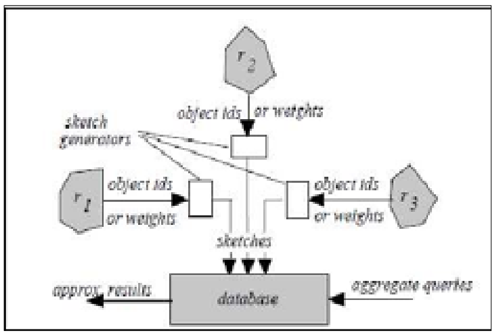
REGIONS AND THEIR AGGREGATE DATA



THE ARB TREE

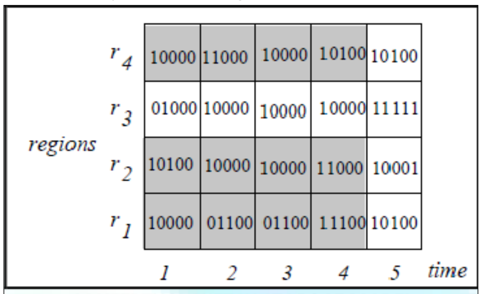
DISTINCT SPATIO – TEMPORAL AGGREGATION IMPLEMENTATION (SKETCH INDEXING STRUCTURE)

Using the FM algorithm discussed in Section 2, for each region ri (1=i=m) and timestamp t we maintain a sketch si(t) that captures the (ids of) objects in ri at t. Figure 3.1 presents the system for distinct aggregation. At each timestamp, every object reports its id (or measure, for DS queries) to the region that covers its location. The region has a sketch generator that creates the corresponding sketches based on the object information, and transmits them to the database.



SYSTEM ARCHITECTURE

The sketches received by the database can be stored in a two dimensional array shown in Figure.



regions					
r_4	10000	11000	10000	10100	10100
r_3	01000	10000	10000	10000	11111
r_2	10100	10000	10000	11000	10001
r_1	10000	01100	01100	11100	10100
	1	2	3	4	5 time

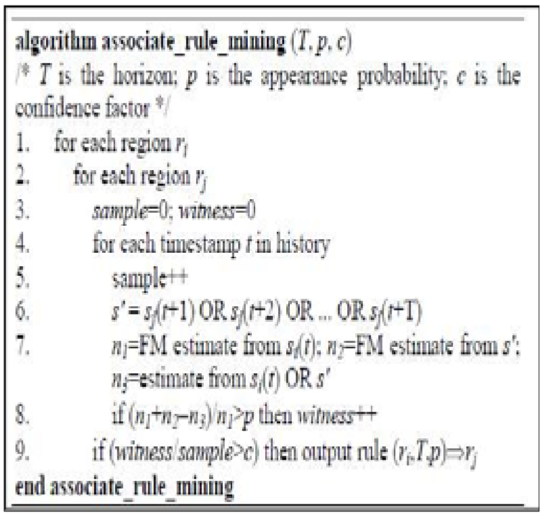
CONCEPTUAL SKETCH STORAGE MODEL

MINING SPATIO – TEMPORAL ASSOCIATION RULES

Consider a user in region ri at time t. What is the probability p that this user will appear in region rj by time t+T? We denote such a spatio-temporal association rule with the syntax (ri,T,p)⇒rj. Inferring such rules is important in practice. For example, in mobile computing, they can identify trends

Volume : 2 | Issue : 3 | December 2012 | ISSN - 2249-555X

in user movements and lead to better allocation of antenna bandwidth to cater for potential network congestions in the near future. Additional constraints, such that ri and rj must be within certain distance, may also be specified. Then, the number of objects that appear in ri at time t and then appear in rj during [t+1, t+T] equals n1+n2-n3. This idea naturally leads to a simple brute-force algorithm for discovering the association rules, checks all possible instances of (ri, rj, t).



```
algorithm associate_rule_mining ($T, p, c$)
/* $T$ is the horizon; $p$ is the appearance probability; $c$ is the
confidence factor */
1.  for each region $r_i$
2.      for each region $r_j$
3.          sample=0; witness=0
4.          for each timestamp $t$ in history
5.              sample++
6.              $s' = s_j(t+1)$ OR $s_j(t+2)$ OR ... OR $s_j(t+T)$
7.              $n_1$=FM estimate from $s_i(t)$; $n_2$=FM estimate from $s'$;
                $n_3$=estimate from $s_i(t)$ OR $s'$
8.              if $(n_1+n_2-n_3)/n_1>p$ then witness++
9.          if (witness/sample>$c$) then output rule $(r_i,T,p)\Rightarrow r_j$
end associate_rule_mining
```

OBJECTIVES AND CONTRIBUTIONS

As described in the previous section, spatial indexing structures do not serve the purpose well enough when they have to deal with spatio-temporal data. Spatio-temporal indexing structures described in the previous section are mechanisms that are designed to deal with continuously changing data points. Another class of spatio temporal indexing structures are designed to deal with moving object data that changes only discretely. Such data can be indexed efficiently by simply coalescing indexing structures like R Trees, Quad Trees etc with those useable for versioning data, like B Trees and B+ Trees. Examples of such indexing structures include MVB Tree. These indexing structures are easy to implement, manage and have their own class of applications. One such application under study as part of this project work is aggregation querying on moving objects data.

AGGREGATION QUERIES

Given a region (a bounding box), a spatial aggregation query is expected to retrieve aggregated data about all moving objects in the specified region. For some applications of moving object databases, this class of queries is very important. Traffic data analysis is one important example. As a motivating example, consider the query "find the road segments with the heaviest traffic near the centre" or, given a medical emergency, "which is the hospital that can be reached fastest, given the current traffic situation" [aR Tree ref]. In both cases, we are interested only in the number of cars and not the specific details. Now, if the positions of the cars and line segments representing roads are indexed in two different R Trees, the join could be time-consuming. Answering a query such as "give me the traffic for every road segment in an area of 1km radius around each hospital" would require a spatial join between the indexing structures. Such a join is inherently costly. Also, analytical/aggregation queries in the spatio-temporal context are different from those involving non-spatial attributes in the sense that there is very little apriori

knowledge about the grouping hierarchy. In addition to some predefined regions, a user may request groupings from an arbitrary grid in a selected window.

INDEXING FOR AGGREGATION QUERIES

Aggregation R Tree[ref] is an example of an indexing structure that is designed to index spatial aggregate data. This structure however does not deal too well with temporal attributes of data points. In fact, it uses mechanisms like individual incremental updates (IIU) and batch incremental updates (BIU) to make moving object updates more efficient. A 'temporal lifting' of such a structure is to add versioning support using B trees that represent versions. A similar yet more advanced structure for spatio-temporal aggregation queries has been proposed, called the aRB Tree[ref]. The aRB or the aggregate R- B- Tree uses 'sketches' to represent moving object data in an R Tree. The details of the aRB Tree have already been dealt with in the previous section.

IMPLEMENTING THE aRB TREE

As described in the previous sections, the aRB Tree uses the R- and B- Trees to maintain spatial and temporal aggregation data respectively. We attempt to implement this structure and design algorithms for answering a decent range of aggregation queries. It is shown in the following sections that, user queries that ask for aggregations from arbitrary groupings can efficiently answered using the indexing structure. The implementation also aims at eliminating certain inherent drawbacks of using R Trees in this indexing structure. Also, for range queries involving time, we suggest the use of B+ Trees instead of B Trees. Also, the indexing structure introduces the concept of sketches for representing moving objects data. This idea is actually very interesting and useful in applications that look for frequent moving patterns. For example, a query that asks for the region at time 't ' with the maximum population density can be easily answered using algorithms on the aRB Tree that deal just with the sketch bit strings. The same is the case with a query that asks for say, 'What is the percentage population in-flow for Electronics City during the time interval (t1, t2)'? Algorithms to a host of similar queries will be described and demonstrated in the sections to follow.

IMPLEMENTATION OF QUERYING FEATURES USING aRB TREE

This section introduces various querying features implemented by the team as part of this project. The detailed algorithms will follow in the next chapter of the report. The simplest aggregation query in this data structure would be to ask for the total number of objects in a given bounding box/area during a given time interval (t1, t2). This would simply involve operations like 'OR'ing of and counting the number of 1's in a given bit string, similar to the algorithm explained in the previous section. The more complex queries could involve asking for the time of day when a given bounding box observed its maximum population density. This query is equivalent to asking a moving object database, "When during the day, did Electronics City's population peak?"

The advantage of using aRB trees with sketches lies in the fact that it houses enough granularity to answer queries like "When did the bus with Route Number 111 enter R T Nagar?", despite the fact that aRB Tree is a structure designed to support aggregation queries. Another interesting query could ask for "List all fighter planes that passed through area - Id BG235".

CONCLUSION

The proposed aRB Tree is implemented using the C language and various querying features are demonstrated as discussed in the previous sections. The structure despite being designed keeping aggregations in mind, serves really well for queries requiring higher granularities. We discover a few drawbacks of the structure proposed in [ref] and suggest implementation mechanisms to overcome them. Also, as mentioned the section 2, structures like R*- trees and B+ Trees would be better options for implementing the structure. Spatio-temporal pattern mining is also a much more simplified task using sketches in aRB Trees.

Volume : 2 | Issue : 3 | December 2012 | ISSN - 2249-555X

REFERENCE [1] Markus Schneider Ralf HartmutGting. Moving objects databases. Morgan Kaufmann Publishers, 2005 | [2] Dieter Pfoser, Christian S. Jensen, and YannisTheodoridis. Novel approaches in query processing for moving object trajectories. In VLDB '00: Proceedings of | the 26th International Conference on Very Large Data Bases, pages 395{406, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. | [3] SimonasSaltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez.Indexing the positions of continuously moving objects. In SIGMOD '00: | Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pages 331{342, New York, NY, USA, 2000.ACM. | [4] http://www.postgresql.org/docs/8.4/static/gist.html | [5] Spatio-Temporal Aggregation Using Sketches Yufei Tao, George Kollios, Jeffrey Considine, Feifei Li, Dimitris Papadias | [6] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP Operations in Spatial Data warehouses.