STAH-TREE
Hybrid Index for Spatio Temporal Aggregation

Marcin Gorawski and Michał Faruga
Institute of Computer Science, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland

Keywords: Spatial databases, temporal databases, aggregation index, multiversion access methods.

Abstract: This paper presents a new index that stores spatiotemporal data and provides efficient algorithms for processing range and time aggregation queries where results are precise values not an approximation. In addition, this technology allows to reach detailed information when they are required. Spatiotemporal data are defined as static spatial objects with non spatial attributes changing in time. Range aggregation query computes aggregation over set of spatial objects that fall into query window. Its temporal extension allows to define additional time constraints. Index name (i.e. STAH-tree) is English abbreviation and can be extended as Spatio-Temporal Aggregation Hybrid Tree. STAH-tree is based on two well known indexing techniques. R– and aR–tree for storing spatial data and MVB-tree for storing non-spatial attributes values. These techniques were extended with new functionality and adopted to work together. Cost model for node accesses was also developed.

1 INTRODUCTION

Nowadays, standard relational database systems are not able to assure efficient processing for detailed and aggregate queries over multidimensional spatial data that have some non-spatial attributes changing over time ('spatio-temporal data'). There are many reasons for that, naming the most important: (i) multicolumn key for single object, (ii) spatial relations are hard to model, (iii) large volume of spatial data implies enormous number of temporal data. Efficient handling of such data becomes more and more urgent, as importance of telemetric solutions and user requirements (concurrent users, higher sampling rate, higher number of meters and more demanding time constraints) grow rapidly. Literature presents a number of dedicated solutions for spatial data processing (R-trees, quad-trees, …)(Guttman, 1984), (Beckerman, 1990), (Manolopoulos, 2003), (Kamel, 1994) temporal data processing (multiversion and overlapping techniques) (Becker, 1996), (Bercken, 1996), (Tao, 2002) spatial aggregates processing (aR-tree, aP-tree) (Zhang, 2002), (Tao, 2004), moving spatial objects (TPR*-tree) (Tao, 2003) and finally temporal aggregate processing (Zhang, 2001). But no solution addressesing detailed and aggregate queries for spatio-temporal data was found.

This paper presents new hybrid index based on cooperating spatial (aR-tree) and temporal (MVB-tree) indexes. These technologies were bounded together and modified in order to provide requested system properties.

2 MOTIVATING EXAMPLE

Last years were time of rapid growth of database solutions especially based on relational model. Despite many advantages this model is not applicable for some solutions. Telemetric system is one of those areas. Telemetric data are asynchronously gathered from meters (gas, water…) that have spatial location and unique identifier. Measurement is characterized by a timestamp and a meter identifier. Relational model for these data is very easy to create. But continuous updates and large volume of data cause query response time not acceptable. To illustrate this problem lets assume 50000 spatial objects and data acquisition performed every hour. The volume of temporal data is reaching 36 millions after a month (0,5 billion after a year) Efficient searching, aggregating or processing becomes impossible even with usage of large computer systems. Such telemetric system can be used in two modes. Normal user requires access to a detailed (history of changes during last month) and

Gorawski M. and Faruga M. (2007).
STAH-TREE - Hybrid Index for Spatio Temporal Aggregation.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - DISI*, pages 113-118
DOI: 10.5220/0002386601130118
Copyright © SciTePress

aggregate (average energy consumption within a year) data about his energy consumption. Time constraints are not too high. On the other hand, producer of specific medium (gas, water, etc.) requires aggregate information about meters that are located in a specific region in various time scales (year summary, predicting future usage). Answer time should be known in advance and should be short.

3 MODIFIED AR-TREE

AR-tree is one of two applied indexes. R-tree family indexes are widely recognized and described in details in literature (Manolopoulos, 2003), (Beckerman, 1990), (Guttman, 1984), (Kamel, 1993), (Kamel, 1994). This paper put emphasis on modifications that were introduced to base the structure and algorithms.

Described system uses static R-trees (spatial data are known in advance, the algorithm builds tree from bottom to top. Such indexes are characterized by maximum node capacity usage. AR-Hilbert Tree (implementation based on (Kamel, 1994) and KNNR-tree (developed in previous research) is used.

These trees were extended with aggregate info (stored in each node) about number of spatial objects contained in each sub tree (count aggregate). Every node is identified with a global unique identifier.

3.1 Specific Queries

This paragraph presents types of spatial queries that are used in the implementation. In all the cases query area is determined by a rectangle Q *Range Query* (**RQ**). This query is identical as in case of standard R-tree. It retrieves all objects that intersects Q *Aggregate Query* (**AQ**). This query was inspired by the range aggregate query. Query result contains aggregate value and identifiers of objects, leaves and nodes that are contained in Q. In other words, query returns objects that are contained in query range despite of their level. *Update Query* (**UQ**). Two input parameters (identifier and MBR) are used to find specific object. MBR is used to reach leaf level and identifier is used for filtering. List of identifiers located on search path from root to leaf is returned. This list is used to update the temporal index.

4 MODIFIED MVB-TREE

STAH-tree uses MVB-tree that was presented in (Becker, 1996), (Tao, 2002). This paragraph describes modifications required for STAH-tree.

Single temporal data entry is described with unique identifier (ID) and temporal validity range (T_1,T_2). It presents measurement made in T_1 for object identified by ID. Value is valid till T_2.

Base MVB-tree version was designated for timestamp queries for single dimensional data and reaches asymptotically optimal node access cost. STAH-tree introduces modifications to achieve high efficiency for other query types.

4.1 Structural Modifications

This paper presents two structural modifications of the MVB-tree structure. There is no limitation to use them at the same time. Both require additional storage space and both accelerate query processing.

4.1.1 Reverse Pointers

Reverse pointers is a technique for efficient time range query processing. Every entry is connected with previous measurement with a pointer – similar to backward-links presented in (Bercken, 1996).

Pointer is created when new measurement is added into the MVB-tree as an insertion procedure locates previous version by default. With the pointers it is possible to retrieve all measurements for single object. Time range query finds object version (in T_2) and follows reverse pointers till value for T_1 is found. This approach is optimal in terms of node accesses as it visits leaves containing measurements changes and traverse tree height only once.

4.1.2 Temporal Aggregates

There is a number of applications for which access to a detailed data is not important or is forbidden and emphasis is put on fast retrieval of aggregates and summaries. STAH-tree is designed to answer aggregate queries in efficient manner, but it also allows to reach detailed data as a standard R-tree and MVB-tree does. It also answers detailed time range queries when reverse pointers are used.

Time range aggregate query summarizes or aggregates values for object with identifier equal to ID in given time range (T_1,T_2). Each entry in modified MVB-tree contains data that aggregates whole history of a specified object. Thanks to that it is possible to answer aggregate query with two timestamp queries and some additional computation of final results based on the retrieved values. First query is executed for T_1 and the second for T_2. It is worth mentioning that MVB-tree is asymptotically optimal for this type of queries (Becker , 1996).

This technique was inspired by aP-tree (Tao, 2004). It gathers aggregates during measurement update. Beside value update (detailed data) system stores aggregates selected by user (sum, average etc.). These additional data increases space consumption and update time (by constant factor) but decreases answer time as aggregate computation is done once at the load time.

4.2 Group Processing Algorithm

Group processing algorithm uses previously described modified MVB-tree. It is based on an assumption that I/O operations dominate CPU cost. Multiple objects are queried in a single operation (at a single timestamp). It calls only these nodes that would be accessed by any of single queries. Many nodes that would be accessed multiple times are loaded only once. Further enhancement includes key sorting. This solution increases CPU cost and response time for single query. Amortized cost is many times lower but there is upper boundary for number of objects in query CPU cost becomes dominant.

5 STAH-TREE

This paragraph presents basic version of STAH-tree that does not use additional accelerating techniques. It makes the idea easier to understand. These extensions will be provided in the next paragraphs.

5.1 Structure

STAH-tree consists of two indexes. The first is responsible for spatial data (R-tree, without aggregates) It provides identifiers of objects that fulfill specific spatial conditions. The second is a modified version of MVB-tree (described above). Spatial objects generate value updates. These measurements are stored in MVB-tree with aggregates and reverse pointers.

5.2 Queries

There are several types of queries that may be posed to STAH-tree. STAH-tree supports original R- and MVB- tree queries as efficiently as original indexes. It allows also to pose new types of queries on both indexes (detailed time range query). The most important is the ability to pose queries that are handled by both cooperating indexes.

Queries which operate on spatial attributes in order to retrieve list of identifiers (range query, nearest neighbors etc.) reference the R-tree. Any R-tree query may be executed without performance loss.

Queries with time constraints (range query, timestamp query etc.) requires list of identifiers for which they shall be executed. They may be passed directly in the query (user knows his meter identifier) or may be selected by a spatial query.

This approach may look not efficient as trees are processed separately. But these indexes are efficient and MVB-tree queries are optimized as there are many queries for the same timestamp. Thanks to buffering most of the operations are performed on nodes already loaded into memory

This section presents the most important queries along with their processing techniques. **Range query** (**RQ**) operates only on spatial index (R-tree). Returns spatial objects (identifiers) which are contained within query range. **Timestamp query** (**TQ**). It is a standard query concerning value for object with known identifier in a specific timestamp. This query is executed directly on MVB-tree. **Time range query** (**THQ**). This query retrieves values for object in specific time range (T_1,T_2). It is executed directly on MVB-tree that applies reverse pointers technique. The algorithm was described in a section prsenting modified MVB-tree. **Timestamp query for a set of objects.** It is a query concerning values in a specific timestamp (T) for set of objects that fulfill spatial query (Q) that is (**RQ+TQ**). Such a query is processed in two steps. First, a set of spatial objects (identifiers) is taken from spatial index. Next, for each identifier a timestamp query is performed on MVB-tree (TQ). **Time range query for a set of objects.** This query retrieves values in a specific time range (T_1, T_2) for a set of objects that fulfill spatial condition (Q). That is (RQ+THQ). This query is processed in the same way, but the query is does not concern a single value of T but retrieves time range values for all objects. **Time aggregate query for set of objects.** This query computes aggregate over a time range (T1, T2) for objects that fulfills spatial condition (Q). It is performed in two steps. First a set of spatial identifiers is retrieved from the spatial index. Next, range queries for all identifiers are performed over MVB-tree and overall result is calculated.

6 STAH-TREE EXTENSIONS

Idea that was presented in previous point can be extended to achieve better performance. This section presents implemented extensions. All of them have strong (positive) impact on performance for all types of compound queries. Impact on standard query types in not negative.

6.1 Using aR-tree

Substituting R-tree with aR-tree (aggregate index) requires large structure modifications. Changes in loading and querying are also required. AR- tree that is used in STAH was already described in paragraph 3. This tree is 'equipped' with 'count' aggregate, all nodes have identifiers.

New value insertion is is performed in two steps. In first step, update query (UQ) is performed on aR-tree for object that generates update. This query returns list of identifiers, object itself and identifiers of all nodes on the path to root. Next, update value is inserted for all identifiers from the list. Thanks to that, the values for whole nodes can be taken from spatial index if the whole node fulfils spatial condition. In other words, aR-tree properties for aggregating on higher levels were moved directly to MVB-tree. MVB-tree is not 'aware' that it contains various types of data – so no additional changes are required within the index. The only disadvantage of this solution is a larger number of temporal data (higher space consumption), but this has no impact on query efficiency. This technique will be referenced as **AGG.**

6.2 Mapping of Spatial Identifiers

Mapping of spatial identifiers was implemented based on experimental results of batch query processing. The number of buffer accesses was higher than expected because object identifiers were not following their spatial location. And identifiers in a query group were dissolved on large key range what leads to large number of node accesses. The best approach is to perform query for list of consecutive identifiers. Mapping is a way to achieve this effect. Identifiers in R-tree are mapped to elements in the same node having consecutive identifiers. Objects are stored with these modified identifiers. While answering aR-query elements form near nodes will create a list of consecutive (more or less) identifiers. That list is sorted and used for querying MVB-tree.

This solution needs to keep additional mapping (old-new key) if the detailed queries on original keys are supported. Aggregate query does not require that. If it is possible to change identifiers permanently, mapping in not required. This technique will be referenced as **MAP**.

6.3 Batch Processing

The number of queries that needs to be performed on MVB-tree may be large (especially for large queries) as their number is proportional to the number of spatial identifiers returned by an R-tree query. This situation is ideal to introduce batch processing technique with no modification in trees structures. Only the new algorithm, able to process multiple spatial objects at the same time is required. These queries reference the same timestamp so they require the same nodes. Most of disk operations will be buffered. This will reduce I/O cost but will increase CPU cost as query processing algorithm is more demanding. This technique will be referenced as **BATCH**.

7 COST MODEL

Cost model predicts query performance as a number of node accesses. It may be used to optimize query processing plan in query optimization module. This section shortly presents the model for STAH-tree with the following assumptions: (1) aR-tree is used (AGG technique); (2) spatial data are uniform; (3) buffering is not taken into consideration – so predicted number of node accesses will be higher than in reality; (4) BATCH and MAP techniques are not used. This model require two equations: (1) selectivity of query on aR-tree; (2) number of node accesses for processing timestamp query on MVB-tree. Some equations for number of nodes at R-tree levels, node sizes in R-tree and others were taken from literature (Theodoridis, 1996).

7.1 AR-tree Selectivity

In the case of STAH-tree aR-tree selectivity is a sum of spatial objects and nodes that were returned by spatial query. It is assumed that size of a spatial area is bounded with (0,1) range in every dimension, query and object sizes in every dimensions are the equal. That simplifies computation with little loss of generality. Following symbols are defined (table 1).

Table 1: Symbols that are used in computation.

Symbol	Description
J	Tree level. j=0 for spatial objects; j=1 for leaves level
I	Dimension
N	Number of spatial objects
N_j	Number of objects at j level
F	'fanout' – number of objects in node
Q	Spatial query range
s_j	Average objects size at j level
P_z	Probability that object is contained in query range
P_p	Probability that object is crossing query
A	Factor used in describing how large part of objects falls in query range when leaf is crossing with query
B	MVB-tree node capacity
NA	Number of node accesses
SEL	Selectivity
n	Number of dimensions

7.1.1 Dependencies between the Object and the Query

Computations are based on a paper (Theodoridis, 1996) and basic probabilistic model derived for dependencies between the query and the aR-tree nodes. This model presents query-object containment probability as an equation 1 and query-object crossing probability as an equation 2.

$$P_z = \prod_{i=1}^{n} (q_i - s_i) \tag{1}$$

$$P_p = \prod_{k=1}^{n} (q_k + s_k) - \prod_{k=1}^{n} (q_k - s_k) \tag{2}$$

7.1.2 Common Computations

Number of nodes at level j+1 can be computed based on equation 3.

$$N_{j+1} = \frac{N_j}{f} \tag{3}$$

Node size at level j+1 (Theodoridis, 1996) can be computed from equation 5.

$$s_{j+1} = \left(f^{\frac{1}{n}} - 1 \right) \cdot \frac{1}{(N_j)^{\frac{1}{n}}} + s_j \tag{4}$$

7.1.3 Selectivity Among Tree Nodes

The number of nodes that are returned is equals the number of nodes that are contained in a query range. It is assumed that 'containment' is computed only at leaves level (always overvalued). The formula for selectivity among aR-tree nodes (leaves) is derived based on equations (1,3, 4). This formula is presented in equation 7.

$$SEL_1 = P_z \cdot N_1 = (q - s)^n \cdot \frac{N}{f} = \left(q - \left(\left(f^{\frac{1}{n}} - 1 \right) \cdot \frac{1}{N^{\frac{1}{n}}} + s_0 \right) \right)^n \cdot \frac{N}{f} \tag{5}$$

7.1.4 Selectivity Among Spatial Objects

Selectivity for spatial objects takes into consideration that a selection of objects are returned in aggregates (query stops on higher R-tree levels). It can be approximated by the number of leaves that cross query range multiplied by a number of objects in a single leaf and constant 'a' (within (0,1) range) that describes how many objects in a leaf are contained in a query. This constant was experimentally set to 0.5. The formula for selectivity among data objects is presented in equation 8.

$$SEL_o = P_p \cdot N_1 \cdot a \cdot f \tag{6}$$

Using equations (2,6) and assumption of same size among spatial objects in every dimension this formula can be rewritten as shown in equation 7.

$$SEL_o = N \cdot a \cdot \left(\left(q + \left(f^{\frac{1}{n}} - 1 \right) \cdot \frac{1}{N^{\frac{1}{n}}} + s_0 \right)^n - \left(q - \left(f^{\frac{1}{n}} - 1 \right) \cdot \frac{1}{N^{\frac{1}{n}}} - s_0 \right)^n \right) \tag{7}$$

7.2 MVB-tree

The paper investigates only one query for MVB-tree, it is range aggregate query. This query is spitted into two queries at extreme timestamps of time range. Retrieval time for each of this queries may be computed with equation 8 (Becker, 1996). The number of node accesses for the whole time range query is equal to this value multiplied by 2.

$$NA_{single} = \lceil \log_b N \rceil \tag{8}$$

7.3 Final Model

The final model is a simple combination of formulas introduced above. The number of queries posed on temporal index equals the spatial result set size multiplied by the doubled height of the tree (8). The number of node accesses to spatial index is relatively small (at least order of magnitude). The number of node accesses in temporal index of STAH-tree is presented in equation 9. This formula may be derived from equations (5,7,8).

$$NA_{MVB} = 2NA_{\sin gle}(SEL_0 + SEL_1) \qquad (9)$$

8 EXPERIMENTS

We performed multiple which proved correctness of all assumptions presented in the paper. The experiments prove that all presented modifications (AGG, BATCH, MAP) result in high acceleration. Objects selectivity prediction was highly accurate (about 5-10% and less). The selectivity for leaves was correct with the same precision when aggregation was performed only on the leaves level. When aggregation was allowed on all aR-tree levels – the error reached 30% for large values (when more than 50% of space was covered by a query). For smaller queries the error did not rise above 10%. Detailed results description is not included as paper page count is limited.

9 CONCLUSIONS AND FUTURE WORK DIRECTIONS

STAH-tree is a complete system that allows to pose various queries over spatio-temporal data. The system performance is highly data parameter-independent. Even for parameters that have large impact on performance in standard solutions (for example relational database).

STAH-tree is based on widely recognized and valued solutions from spatial and temporal data processing domain (R-tree, aR-tree, MVB-tree,…). These ideas are combined and adapted is order to work together and assure required features. The original solution was extended with additional accelerating techniques (BATCH, AGG, MAP). The presented solution is more robust than the one using the components separately. Disadvantage is higher storage space consumption.

Selectivity prediction model for aggregate R-tree was introduced. Its accuracy was proved by experiments. The selectivity model along with node accesses formula for MVB-tree (available in literature) are merged in order to achieve full performance model for STAH-tree (with respect to node accesses).

Several main directions of future fork have been recognized: (1) selectivity model for non uniform data distribution; (2) extension of presented selectivity model for aR-tree nodes with the possibility of aggregating on higher aR-tree levels; (3) selectivity model for different types of spatial queries (KNN, Spatial Join etc.); (4) replace spatial and temporal indexes with other techniques (quad trees, OVB-trees, …). (5) use more sophisticated way to aggregate spatial objects (not only point aggregates) as it is done in (Zhang, 2002). Work on some of these issues has already started.

REFERENCES

Manolopoulos, Y., Nanopoulos, A., Papadopoulos A. N., Theodoridis Y. 2006. R- Trees: Theory and Applications. Springer

Beckerman N., Kriegel H.P., Schneider R., Seeger B. 1990. The R*-tree: An efficient and robust access method for points and rectangles. Proc. SIGMOD International Conference on Management of Data, pages 322-331

Guttman A. 1984 R-trees: A dynamic index structure for spatial searching Proc. SIGMOD International Conference on Management of Data, pages 47-57

Kamel I. Faloutsos. 1994. Hilbert R-tree: An improved R-tree using fractals. Proc. International Conference on Very Large Databases, pages 500-509

Kamel I. Faloutsos. 1993. On packing R-trees. Proc. International Conference on Information and Knowledge Management, pages 490-499

Becker B., Gschwind S., Ohler T., Seeger B., Windmayer O. 1996 An Asymptotically Optimal Multiversion B-tree VLDB Journal, 5(4), pages 264-275

Tao Y., Papadias D., Zhang J. 2002 Efficient Cost Models for Overlapping and Multi-Version Structures. TODS, 27(3), pages 299-342

Tao Y., Papadias D. 2004 Range Aggregate Processing in Spatial Databases. IEEE Trans. Knowl. Data Eng. 16(12), pages 1555-1570

Theodoridis Y., Sellis T., 1996. A model for the Prediction of R-tree Performance. Proc. Symp. Principles of Database Systems

Comer D., 1979. The Ubiquitous B-tree. ACM Computing Surveys, Vol. 11, No 2, pages 121-137

Zhang D., Tsotras V. J., Gunopulos D. Efficient aggregation over Objects with Extent. PODS 2002, pages 121-132

Zhang D., Markowetz A., Tsotras V. J., Gunopulos D., Seeger B. Efficient Computation of Temporal Aggregates with Range Predicates. PODS 2001

Bercken J., Seeger B. Query Processing Techniques for Multiversion Access Methods. VLDB 1996