The Po-tree, a Real-time Spatiotemporal Data Indexing Structure

Guillaume Noël, Sylvie Servigne, Robert Laurini

Liris, INSA-Lyon,
Bat B. Pascal,20 av. A. Einstein,
69622 Villeurbanne Cedex FRANCE
{noel.guillaume, sylvie.servigne, laurini}@insa-lyon.fr

KEYWORDS: Spatio-temporal, Database indexing, Soft Real-time, Natural disaster prevention

ABSTRACT

This document describes the Po-tree, a new indexing structure for spatio-temporal databases with real-time constraints. Natural risks management and other system can use arrays of spatially referenced sensors. Each on them sending their measurements to a central database. Our solution tries to facilitate the indexing of these data, while favoring the newer ones. It does so by combining two sub-structures for the spatial and temporal components. While Mobility is not yet supported, evolutions of the structures shall be able to deal with it.

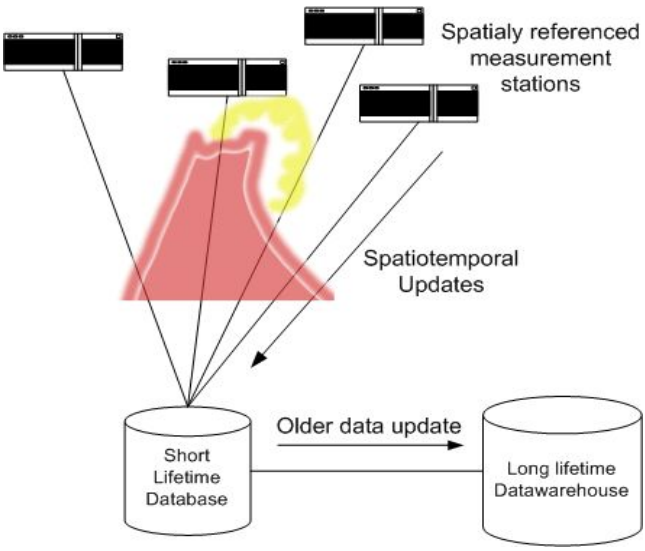
INTRODUCTION

Geographic Information Systems provide solutions to a wide panel of problems, from agronomy to urban planning or natural risk management. The databases linked to such systems usually are very large and cumbersome. They have to keep tracks of numerous heterogeneous data. Solutions exist to face this kind of needs. Yet, a particular aspect remains partially untouched: spatio-temporal indexing with real time constraints. While our application case is linked to natural disasters prevention, we shall show that the structure we propose can be extended to cover different needs. We currently propose a new database spatio temporal access method for spatially referenced sensors, the Po-tree.

This paper is divided into three chapters. First we define more precisely the problem we intend to address, then we follow by an introduction to our application case. Next comes a brief state of the art. Finally, we introduce our solution, the Po-tree, and some test results to study its usefulness.

1 Application case

Our study fields is linked with the work of volcanologists, trying to monitor a Mexican volcano, the Popocatepetl (CENAPRED,2003). An array of fixed measurement stations, hosting various sensors has been set up around the volcano. 15 stations record data within 1.5km from the crater. Others are located further down the slopes. Each sensor, spatially referenced as a fixed point, sends measurement datum toward a central database. This database is later on replicated to a data warehouse. See Figure 1 for a visual description. The Po-tree aims at indexing the database, while keeping in mind some recommendation for environmental



data warehousing (Adam & al, 2002).

Fig. 1. application case

Real time constraints stems from the number of data to process. Updates occur in a periodic - chronological order. Measurement frequency for a sensor can be up to 100Hz, as for seismographs. On this aspect, update transactions are more important than lookup queries.

The volcanologists tend to consider the most recent data as the more valuable, as they help understanding the actual activity of the volcano. Users usually query the database so as to fetch data coming from a specific sensor (spatial location) for given amount of time (temporal interval). Volcanologists generally use reference sensors so as to determine the global state of the volcano. Later on they query complementary sensors to confirm their analysis. Therefore, most lookups are spatial-point / time-interval. They are followed by range / interval queries, as defined by Erwig (Erwig, & al, 1999).

The specific needs of this kind of application are now quite well understood. The spatio-temporal access method used should focus on two aspects: the update transactions cost and the priority given to the newest data. Different studies have already opened the path for real-time, spatial, temporal and spatio-temporal databases.

2 Other studies

Real-Time approaches are based on the respect of time constraints, as defined in (Lam & Kuo, 2001). For databases, it means that transaction are to commit before a deadline. In the case of an array of sensors, the deadline is the arrival time of a new measurement.

Spatial approaches, as defined in (Ooi & Tan, 1997) can be divided into three categories.

- The first one tend to linearize the data, represented as points, in order to use well known indexing structures, such as the B+-Tree.
- Another approach intends to use a non-overlapping native space. The space is divided into non-overlapping sub-spaces. The objects are referenced within these subspaces. An object spanning across two sub-spaces may be duplicated or clipped.
- The last approach is to use an overlapping native space. The space is divided into overlapping sub-spaces.

Among the main indexes, two can be noted. The R-tree (Guttman, 1984) family uses minimum bounding structure as sub-spaces to create a hierarchy accessible through a B-tree. In the Kd-tree (Bentley, 1975), a binary tree, points are sorted according to reference points and reference dimensions.

In temporal approaches, the notion of time can lead to the use of balanced trees, such as the AP-tree (Gunadhi & Segev, 1993). Another

approach is simply to consider the time dimension as another spatial dimension.

This distinction has lead to different spatio-temporal approaches, as defined in (Wang, & al, 2000). Objects can be considered as:

- Objects that continuously move
- Objects that discretely change, such as in our case
- Continuous change of movements

Different families of access method can be defined.

- Time can be considered as another spatial dimension, as in the 3D-R-Tree case (Theodoridis, Vazirgiannis & Sellis, 1996)
- Multiversion tables can be kept to track the data, as for MVLQ (Tzouramanis T., Vassilakopoulos M. & Manolopoulos Y, 2000)
- Overlapping snapshots can be used as an alternative to multiversioning. HR-trees are a good example (Nascimento,M. & Silva J., 1998).
- A dimension can be prioritized over another, as the spatial priority given in TR-trees (Xu, Han & Lu, 1990).

None of these approaches are completely compatible with our case study. R-trees require lengthy Construction times. Approaches usually do not focus on the most recent data. However some ideas have lead to the development of a new tree: the Po-tree.

3 Po-Tree

Our solution, the Po-tree, is based on the differentiation of temporal and spatial data, with a focus given to the latter. The spatial aspect is indexed through a Kd-tree, while the temporal aspect uses modified B+-trees (see figure 2). The measurement stations being immobile, the current structure does not allow mobile sources of information, mobile sensors. Every spatial location, similar to a spatial object (sensor) is directly linked to a specific temporal tree. Queries shall first determine the spatial nodes concerned by a transaction and later on determine the temporal nodes.

3.1 Po-tree Structure

The Po-tree can be divided in to parts. A first sub-structure covers the spatial aspect. Each location, each sensor is linked through this spatial sub-tree to a temporal sub-tree.

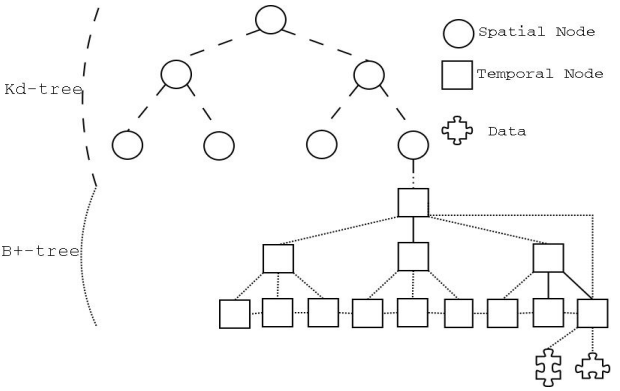


Fig. 2. Po-tree structure

3.2 Spatial Component

Kd-trees are simple spatial structures but not perfect ones. Their main problem is the fact that their final shape relies on the data insertion order. If data are entered in different orders, the final trees may have different shapes.

However, Index Concurrency Control methods, originally designed for B-trees with real-time constraints can easily be adapted to cover Kd-trees. Latches, 'fast locks', can be converted to be used on binary trees. Different tests have also proven that Kd-trees fared reasonably well compared to R-trees for small number of data (Paspalis, 2002).

Within a Kd-tree, entries take the from <left-pointer; reference-point; right-pointer>. In order to create a Kd- tree, a reference dimension D and a reference point P_i are taken. Then, all points with a greater value for the dimension D than P_i shall fall to the right part of the branch, and those with a lesser value to the left. At the next level of the tree, other reference points P_{i+1} and P_{i+2} are taken, and the reference dimension becomes D_{+1}.

In the Po-tree, each spatial point is composed of a spatial definition of the point and a link to a temporal sub-tree. It does not directly record the temporal components. Therefore, a whole definition of a spatial sub-tree entry would be <left-pointer; point-position; link-to-temporal-tree; right-pointer>. As spatial deletes should not occur, there shall not be empty spatial nodes.

3.3 Temporal Component

The temporal components to index are held within modified B+-trees. The tree records the measurement times and links to the actual measured values. Each spatial object is linked to a different temporal sub-tree.

As for the temporal aspect, it has been noticed that the most recent data are considered of higher interest than older data. It has also been noticed that data insertions are generally held at rightmost leaf of the temporal sub-structure, where are found the newest nodes.

Therefore, the temporal sub-tree has been modified to add a direct link between the root and the last node. While maintaining this link requires minimum computation from the system, a simple test during query processing prevents being forced to traverse the whole tree so as to append or to find the requested data. This direct link, updated during the data insertion procedure, is useful to save processing time.

As each temporal sub-tree is linked to an object, it is possible to develop a secondary structure in order to directly access the data of specific objects, without the need to search through their spatial position. This would however incur the addition of an Information Source identifier to the temporal sub-structure. However, this could be useful for the notion of hierarchy of information sources.

As most, if not all, of the updates take place to the rightmost part of the temporal sub-tree, the fill factor of leaf nodes can be placed higher than usual. Delete transactions should be somehow rare under normal conditions, and a posteriori updates should be as rare. The exception being when the transmitting systems experience lag time due to network problems between the sensors and the database, or when the database has to restart update transactions, which can occur because of real-time constraints and node access concurrency control. Therefore split and merge procedures can be changed so that the nodes can be filled almost at their maximum capacity.

Within the temporal sub-tree, the data are indexed according to their start-time. The periodicity of the sensors, and the assumption that they shall not go down unnoticed makes us considering the start-time only. For lower frequencies the end-time should be included as well in order to define the data lifetime. So far, entries take the shape of <pointer-0; key-1; pointer-1; key-2...; pointer-n>. Furthermore, a direct link between the root and the last node accelerate the query processing of the newest data. A simple test between the value to process and the first key of the last node determines if the query is related to the most recent data or to older one. If

the value is greater than the key, the last node is directly returned, searched or updated. This greatly helps in lowering the processing time by preventing lengthy tree traversals.

3.4 Spatio-temporal linking

Requests on the Po-tree can be divided in three different parts: spatial localization, Information Source linking and temporal localization.

Lookups start by searching the spatial tree. For point lookups, it directly fetches the Information Source and its temporal sub-tree. From here on, the lookup query searches the temporal sub-tree.

For range lookups, the query starts by determining the different Information Sources within the spatial range. Then, for each of them, it then starts a temporal lookup. The queries are answered by giving the specific results of Information Sources one by one.

For insertions, the transaction starts by defining the spatial position of the inserted data. If the position is already defined, the transaction can directly proceed with its temporal part. If the position is not defined, the transaction starts by inserting the spatial point in the sub-tree. After this first step, it creates a new temporal sub-tree and links it to a new Information Source.

Information Source linking references the change between the spatial and the temporal sub-trees as each of them holds a part of the indexed data.

Temporal queries start by comparing the timestamp or the end of the interval with the first temporal key of the last node. This node is directly accessible through the root of the tree. If the query deals with recent data, the query can act directly on the last node. On other cases, the query then proceeds with a B+-tree lookup.

Temporal intervals are dealt with by first finding the end of the interval in the tree and by using the node-links to cover the entire interval length.

Queries use the B+-tree rules. For updates however, if a new leaf-node is created to the right of the tree the link between the root and the last node is accordingly updated.

This configuration implies the Po-tree is more specifically designed for queries on the most recent data. Spatial range / Temporal interval requests that does not ends at the present time does not fully take the advantages of the specificities of the tree.

4 Tests

Different tests have been conducted between the Po-tree, R and R*-tree structures (Hadjieleftheriou, 2003). Randomly generated data have been generated and sequentially issued to a fixed number of random points acting as Information Sources. Tests have been conducted changing the total number of data to index (1000-200000), the number of information sources (10-5000) used and the portion of the base to scan for interval queries (the 5 – 30 last percents). The tests have been conducted on a 1.6 G Hz, 128 Mo RAM computer, running Linux. The programming phase was done under Java SDK 1.2

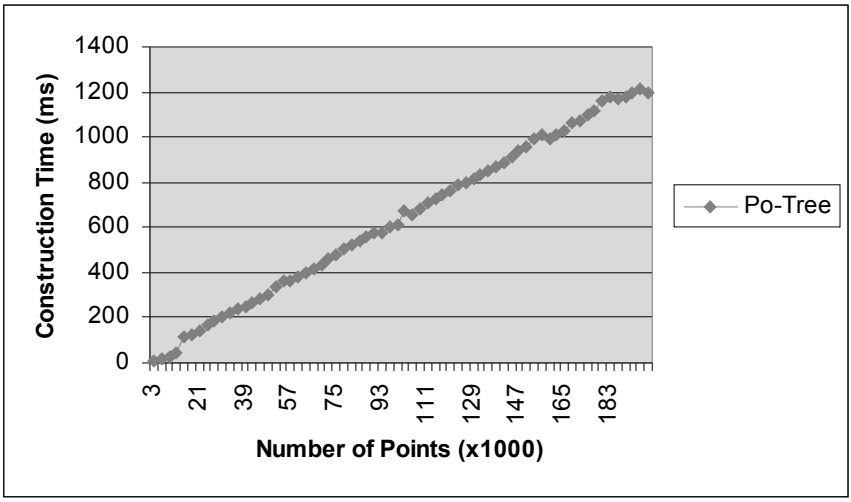


Fig. 3. Po-tree Construction Time

The first notable fact about the Po-tree comes from its construction time (see Figure 3). Tests with a fixed set of 30 sensor, compatible with our test case, along with 200 000 updates (batch) show a linear building time.

From this point, it is interesting to examine the effect of the number of different positions for queries, as we use two complementing structures so as to obtain our Po-tree (see Figure 4). The tests have dealt with a fixed total set of data and a varying number of spatial positions, from 50 to 5000. The actual variation of the construction time (similar results have been found for lookups) is steady, increasing by steps. Note that the Kd-tree performances are linked to the order of insertion of data, as the shape of this tree is not deterministic.

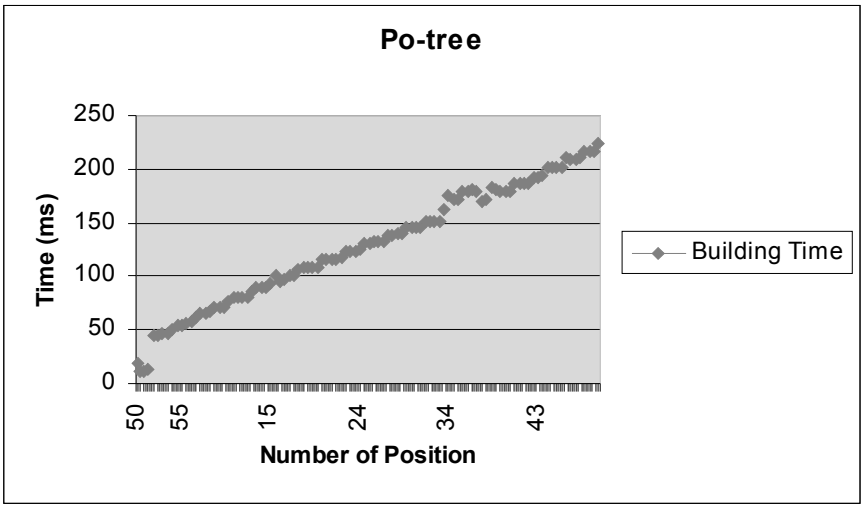


Fig. 4. Influence of the number of sources

Complementary tests between the Po-tree and the R*-tree have shown some interesting properties of the new structure. First of all, the construction time of the Po-tree is by far lesser for the Po-tree. To index (batch) 25 000 points from 30 information sources, the R*-tree takes some 45 seconds, while the Po-tree takes less than one.

This can be partly explained by the fact that the R*-tree has to deal with Minimum Bounding Structures, while the Po-tree simply has to determine which B+-tree to append. While the R*-tree must consider the whole set of data for answering queries, the Po-tree segments the dataset in spatially different sub-parts.

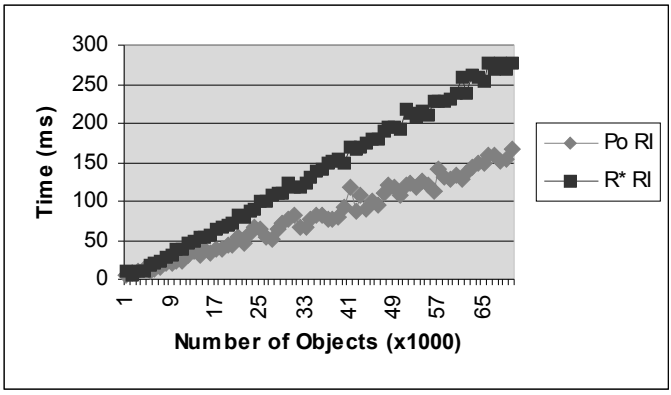


Fig. 5. Range-Interval Search Time

As for lookups, the R*-tree performs slightly better for small number of objects, but this trend soon changes when there are more than 5000 objects (see Figure 5). This can be explained by the fact that the R*-tree must work with the whole set of data while the Po-tree, with a first spatial filtering later on only has to consider a part of the set. The more data are processed, the bigger the differences are. This has been verified for spatial Point / temporal Point, Point / Interval and Range / Interval queries.

The results obtained have shown that the Po-tree was compatible with the constraints set by our application case: favoring the newest data, processing of big quantities of data in a given time, fixed set of spatial sources, possibility to use in a real-time system. Even though the mobility is not yet easily managed, the Po-tree meets the initial specifications.

CONCLUSION

The Po-tree aims at indexing spatio-temporal data issued from a network of spatially referenced sensors, with a focus given to the newest data. Our goal was to accelerate answering time to real-time queries. Our application case is linked to volcanic monitoring, yet it can be extended to include other natural disaster prevention scenarios, or scenarios where a set of fixed spatially referenced sensors sends huge quantities of data to a central database. The structure of the Po-tree uses two parts. The main difference with the existing solutions stems from this division. The Po-tree uses the spatial dimension to divide the dataset into temporal sub-trees. The spatial sub-tree references the positions of fixed sensors, Information Sources. Each position, each sensor, is then linked to a temporal sub-tree pointing to the actual data. The spatial sub-tree is based on the Kd-tree, compatible with Index Concurrency Control protocols, yet sensible to the insertion order while the temporal tree is a modified B+-tree, akin to an AP-tree. Different tests have shown that this solution can be used with ease when updates from a given set of sensor are frequent. The lookup strategy has been designed to favor the newest data thanks to a direct link between the root of the temporal sub-tree and last node. The notion of Information Source allow fast interval queries as the data from a given source are linked through the temporal sub-tree.

Further developments of the structure will include mobility and the use of Quadtrees to replace the actual spatial sub-tree. Another point that shall be studied will be the linkage of the database to the data warehouse. The Po-tree has been designed with specific needs in mind, but it can be easily

adapted to cover a majority of natural disaster cases where fixed sensors are the main source of information.

Acknowledgments should be made to the Universidad de las Americas, Puebla, for their work on the Popocatepetl monitoring. They coordinate and greatly help the different researches based on this volcano.

BIBLIOGRAPHY

Adam N., Atluri V., Yu S. & al, 2002, Efficient Storage and Management of Environmental Information, *in Proceedings of the 19th IEEE Symposium on Mass Storage Systems*, (USA, Maryland)

Bentley J.L., 1975, Multidimensional binary search trees in database application, *IEEE Transaction on software engineering*, 5(4), 333-340.

Bliujute R., Jensen C.S., Saltenis S. & al., 2000, Light-Weight Indexing of Bitemporal Data, *in Proceedings of the 12th International Conference on Scientific and Statistical Database Management* (Germany, Berlin), pp. 125-138.

CENAPRED, 2003, Monitoreo y Vigilancia del Volcan Popocatepetl, http://tornado.cenapred.unam.mx/mvolcan.html

Erwif M., Güting R.H., Schenider M. & al, 1999, Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases, *GeoInformatica*, 3(3), 269-296

Guttman A., 1984, R-trees: a dynamic index structure for spatial searching. *In Proceedings 1984 ACM SIGMOD International Conference on Management of Data*, (USA, Boston) pp. 47-57

Hadjieleftheriou M., 2003, Spatial Index Library, http://www.cs.ucr.edu/~marioh/spatialindex/

Haritsa J.R., Seshadri S., 2001, Real- time index concurrency control. In *Real Time Database System – Architecture and Techniques*, edited by K.Y. Lam and T.W. Kuo (Boston : KluwerAcademic Publishers) ISBN: 0-7923-7218-2, pp. 60-74

Lam K.Y., Kuo T.W., 2001, Real time database systems: an overview of systems characteristics and issues. IIn *Real Time Database System –*

Architecture and Techniques, edited by K.Y. Lam and T.W. Kuo (Boston : KluwerAcademic Publishers) ISBN: 0-7923-7218-2 , pp. 4-16

Lam K.Y., Kuo T.W., Tsang N.W.H., & al, 2000, The reduced ceiling Protocol for concurrency control in real-time database with mixed transactions, *The computer journal*, 43(1), 65-80

Mokbel M., Ghanem T.M. & Aref W.G., 2003, Spatio-temporal Access Methods, *IEEE Data Engineering Bulletin*, 26(2), pp. 40-49

Nascimento,M. & Silva J., 1998, Towards Historical R-trees. *In Proceedings of 1998 ACM Symposium on Applied Computing*, (USA, Atlanta) pp. 235—240

Ooi B.C., Tan K.L., 1997, temporal databases. *In Indexing Techniques for Advanced Database Systems*, edited by E. BertinoB.C. Ooi, R. Sack-Davies & al (Boston, Kluwer Academic Publishers), ISBN 0-7923-9985-4, 113-150

Paspalis N., 2003, Implementation of Range searching Data-Structures and Algorithms, http://www.cs.ucsb.edu/~nearchos/cs235/cs235.html

Theodoridis Y., Vazirgiannis M. & Sellis T., 1996, Spatio-temporal indexing for large multimedia application, *In Proceedings of the 3rd IEEE conference on multimedia computing and systems*, (Japan, Hiroshima)

Tzouramanis T., Vassilakopoulos M. & Manolopoulos Y, 2000, *Multiversion Linear Quadtrees for Spatio-temporal Data*, Proceedings 4th East-European Conference on Advanced Databases and Information Systems, (Czec Republic, Prague) pp.279-292

Xu X., Han J. & Lu W.,1990, RT-Tree: an improved R-tree indexing structure for temporal spatial databases, *in Proceedings of the 4th International Symposium on Spatial Data Handling*, (Switzerland, Zurich) pp. 1040-1049

Wang X., Zhou X., Lu S., 2000, Spatiotemporal Data Modeling and Management: A Survey, *In Proceedings of the 36th International Conference on Technology of Object-Oriented languages and Systems*, (China, Xi'an), PP. 202-221