Soft Real-Time GIS for Disaster Monitoring

Robert Laurini, Sylvie Servigne and Guillaume Noel

LIRIS, INSA de Lyon, F-69621 Villeurbanne Cedex, France.
Email : Robert.Laurini@insa-lyon.fr; Sylvie.Servigne@insa-lyon.fr; Noel.Guillaume@insa-lyon.fr

Abstract

The goal of this paper is to underline the importance of real-time systems for managing information during the phase of disaster monitoring. We stress the importance of soft real-time GIS, and we present a list of barriers to overcome in order to get this kind of system. Among the barriers, we present a solution for real-time indexing of spatio-temporal data based on a data structure named PO-Tree.

1 Introduction

The monitoring of natural phenomena is a key element of any situation which can provoke some disasters, such as flooding, volcanic eruptions, landslides and so on. Usually a lot of sensors are used to measure data about the phenomenon under monitoring. Those data are regularly sent to some control center and are stored in a real-time GIS, and displayed in real-time with animated cartography. So decision-makers can follow the phenomenon and be prepared to make relevant decisions. As a direct consequence, computers must always be on.

In addition, real-time characteristics of GIS can also be important during the response phase, for instance to monitor shelter and hospital capacities.

The goal of this paper will be to give a complete list of functionalities that a real-time GIS will offer, to examine them, and to identify the actual existing barriers in order to set a research program. We will continue by providing a new structure for real-time indexing of spatio-temporal data.

2 What is a Real-Time GIS?

Presently are emerging new GIS applications for which the time is a critical factor. For instance during disaster management, information must be collected in real-time, and made immediately available to a lot of potential users. In this kind of application, the characteristics of the database contents are far from the conventional view (for instance, administrative boundaries), for which the information about feature presents none or very slow evolution. In other words, in contrast, in the past, we were dealing with applications for which the date of storing or updating information was not a very critical factor.

Let us take another example in environmental monitoring such as river and flood monitoring. Several sensors are distributed along the river, regularly measuring several parameters, chemical, physical or biological, in addition to water height. After having made the measures, the sensors send the corresponding data to a control center, by using any kind of telecommunication system (could be satellite-based, could be using cellular phones attached to the sensors, etc.). In the control center, a front-end system manages the dialog with the sensors and stores the information into a database. Then, another system visualizes this information to give the decision-maker relevant information about the river. And especially, facing any crucial event, the periodicity of collecting information must be increased, for instance passing from every hour to every minute. Doing so, we are multiplying the number of data to be transferred, so increasing the importance of time: now all transactions must be committed very rapidly, and no system crashes will be permitted.

Several kinds of real-time systems can be considered according to the type of temporal constraints to follow. Usually, we speak about hard real-time when transactions must be fulfilled with hard deadline; for instance for military applications, constraints can be of millisecond magnitude. In disaster management, the phenomena under monitoring are slower, and so we speak about soft real-time systems.

With other words, three aspects are the more critical:

- when a bunch of data arrives, it must be stored very rapidly; that is to say no queue is allowed to install data in their optimal location; differently said no indices must be reorganized; as a consequence conventional indices such as based on quadtrees or any kind of R-trees are totally outside consideration;
- secondly, even if new data arrive, the database must never be saturated or completely full; a special mechanism must be provided in order to

avoid this drawback, per instance by regularly flushing elder data into a datawarehouse;

- thirdly, the computer system must be very robust, that is to say no failures must be accepted; especially the computer must always run, perhaps using electrical batteries.

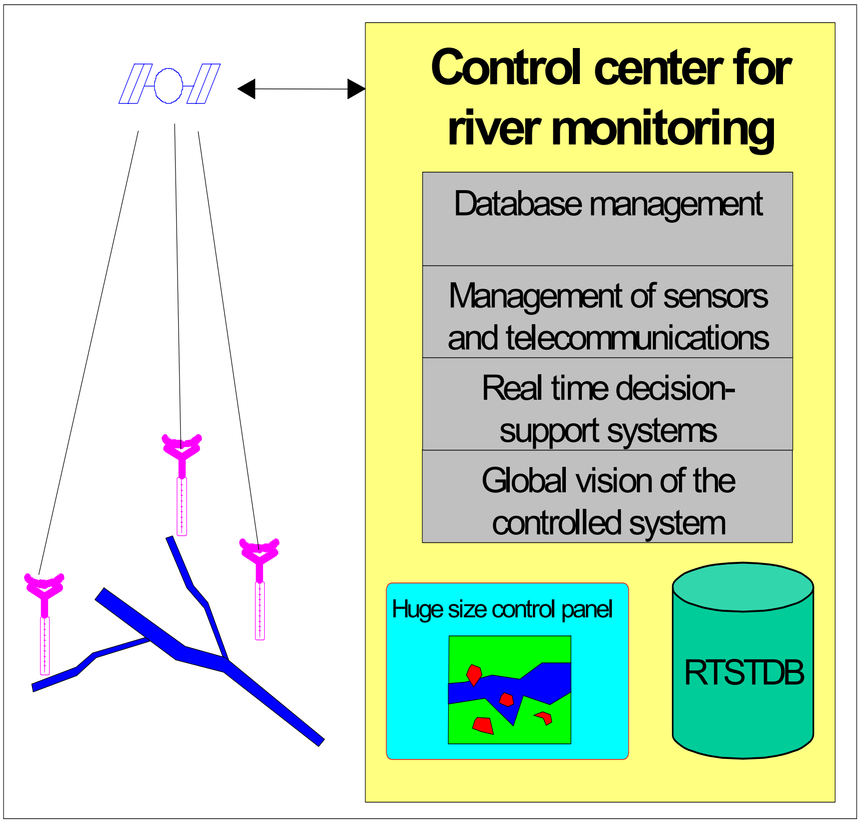

Control center for river monitoring
Database management
Management of sensors and telecommunications
Real time decision-support systems
Global vision of the controlled system
Huge size control panel
RTSTDB

Fig. 1. Architecture for river monitoring

In order to set all the specifications of a real-time GIS, several additional functionalities must be taken into account. Let us examine, or rediscover some of them.

2.1 Input: Role of Sensors

In conventional databases, the main data entry procedure is based on some kind of human-interface dialog. But in our concern, the main procedures will be based on sensors. By sensors, we mean any kind of electronic devices able to make measures on some physical phenomena, and to send this information to a control center by means of any telecommunication system. Different kinds of sensors can be distinguished:

- Passive sensors are built to send regularly their measures, for instance every 10 seconds.
- Programmable sensors for which the periodicity of the measures can be modified by using some remote control statements.
- Intelligent sensors, the program of which can be downloaded. Those sensors can present a sort of "intelligent" behavior by making local decisions, perhaps according to some predefined rules.

Whatsoever the type of the sensor is, the data which are transmitted must be immediately taken into consideration. One crucial problem is when a crisis occurs. In this content, the measuring periodicity is accelerated, and then multiplying the transmitted data: the consequence could be a sort of congestion in data arrival, and the system must be able to manage this sudden increase within critical specified times (for instance all measures must be stored in less than 10 milliseconds after acquisition was made).

2.2 Storage

As explained earlier, the key idea is to store the maximum of information into the main memory, especially more recent data, and to flush ancient data into an archive.

For real-time GIS, this key idea must be accepted: in essence, a first computer can be in charge of the system, whereas a second computer must be in charge of flushing the database and of managing the archives (See Figure 2). Those archives can be used as datawarehouses, in which some OLAP procedures can be launched for analyzing data.

Usually, the phenomena under consideration are continuous, and the sensors only measure a few points. So, some powerful mechanisms for interpolating data, either at temporal, or at spatial, or at both levels must be provided, especially for data of corresponding to spatio-temporal continuous fields such as temperatures, pressures, winds, etc. See (Laurini-Gordillo 2000) for more details.

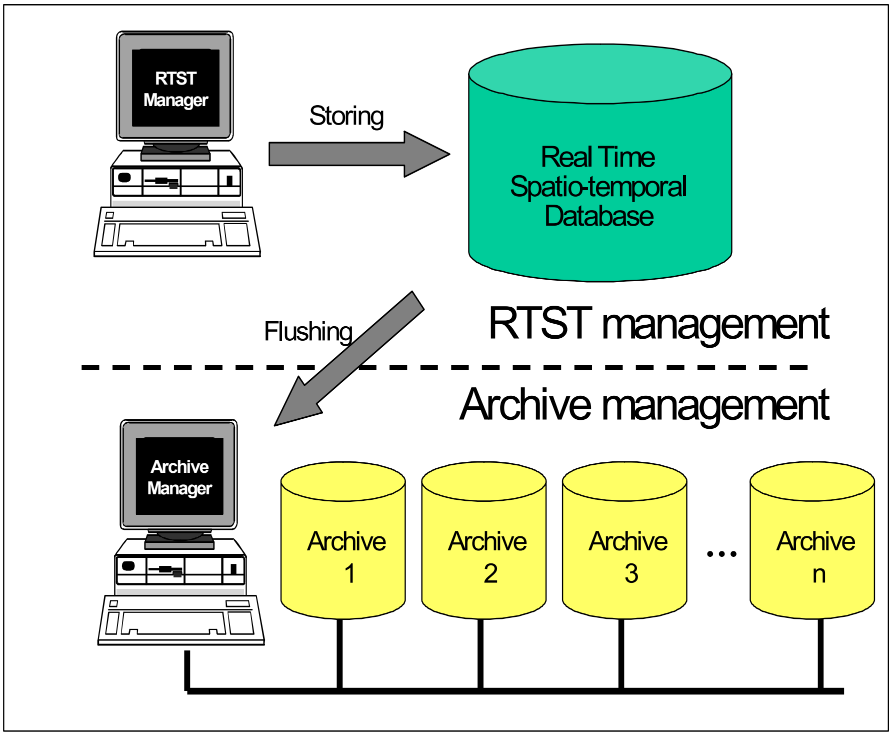


Fig. 2. Structure of a real-time system for geographic information with regular flushing of data into archive (RTST meaning real-time spatio temporal)

2.3 Output: Real-Time Animated Visualization

The privileged output is visualization, and especially real-time visualization. Among visualization systems, animated cartography is a simple way to present the evolution of an object, or a process into a territory. Especially real-time animated cartography in huge format real-time interactive panels must be studied, including:

- cognitive aspects of decision-making in real-time,
- graphic semiology for animated cartography,
- automatic selection of relevant data for synthesizing information for decision-makers.

2.4 Querying and Indexing

Extensions of SQL for real-time (Prichard-Fortier 1997) are now proposed. Those extensions must be harmonized with extension dealing with spatio-temporal issues. This must be done not only for relational DBMS, but also for object-oriented DBMS.

For indications about languages for moving objects, please refer to Güting et al. (2000). Concerning indexing, the Historical R-trees look to be a good starting point (Nascimento et al. 1998).

2.5 Interpolation and Extrapolation

In real-time GIS querying languages, automatic interpolations must be provided to retrieve not only in locations between stored times, but also amongst spatial data, especially when dealing with continuous data (Laurini-Gordillo 2000).

2.6 Integrity Constraints Correction, Sensor Failure

In addition to robustness to sensor- or system-failures, spatial and temporal integrity constraints must be checked and maintained in real-time. It is interesting to note the difference between sensor and system failures. Indeed it is important to discern failures linked to a faulty sensor and failures linked to a system error. Different studies deal with failure detection in sensor networks using redundant data, state prediction, sensor data validation through degrees on confidence in the measurements and fusing of estimated and measured values (Satnam et al. 2001). Real-time studies tend to lead to more flexible constraints so as to maintain a higher level of transaction commits. However this leads to lower the precision of the measured queries as higher flexibility comes from less accurate measurements.

2.7 Interoperability

Even so interoperability is not the goal of any real-time GIS, this requirement is nagging for users. A solution is that the data model must be compliant with standards such as those proposed by OpenGIS consortium (Buelher-McKee, 1996). Spatial models were the first goal of Open GIS, then they tackle spatio-temporal data. As far as we know, there is not yet standards for real-time spatio-temporal data. At best, we can take notice of the preliminary works done on real-time spatio-temporal data queries for the PLACE project, aiming at continuous queries (Mockbel et al., 2004).

2.8 Links to the Internet

The links for Internet must be provided. A URL must be considered as an abstract data type in order to embed hypermap potentialities (Laurini-Milleret-Raffort, 1990) in the real-time GIS.

2.9 Security, Failures

As any real-time system, robustness is a key issue for any real-time GIS. The system must provide security for any user, especially when facing computer failures.

2.10 Conceptual Data Models

For analysis and design purpose, conceptual data models must be invented taking, not only spatio-temporal aspects, which is already done (Price et al. 1999) but also real-time characteristics (Douglas, 1998).

So those tools will facilitate the user for the designing of any new GIS applications integrating real-time components.

3 A Barrier to Overcome: Spatio-Temporal Real-Time Indexing

A state of the art on the different existing methods is essential to better understand the pros and the cons, the main concepts of the situation.

3.1 Soft Real-Time

For the real-time approach, the main idea is to answer queries within time constraints (Noel, Servigne, Laurini, 2004). It is possible to separate three kinds of constraints (Lam, Kuo, 2001). The soft one, used in our case, imply that transactions should be fulfilled within time limits, yet it is understandable that some transaction can not comply with the limits. The firm constraints, more restrictive, allow some transaction not to be fulfilled within the time limits, yet in this case the whole system can be slightly impaired. The hard constraints, finally impose that under no circumstances a transaction should miss a deadline. Otherwise, the system could come to a halt. Priorities are generally used to define which transaction is more important than another, which is not equivalent to define which one should occur before another. Different techniques can be used so as to assign priorities: Earliest Deadline First, Rate Monotonic and other variants (Lam, Kuo, 2001).

Real-time computing is not similar to Fast-computing. Fast-computing does not prevent a low priority transaction to block high priority transactions (priority inversion) because they have already locked the access to some resources, data. Moreover, for databases, the current paradigm is to keep the index, and even the whole base in main memory so as to reduce the number of slow disk access. Index Consistency Control (ICC) methods can then be used to make sure no priority inversion occur while accessing the index (Haritsa, Seshadri, 2001).

3.2 Spatial Approach

The spatial approaches in indexing often tend to linearize the data so as to use known "fast" structures. Such is the case for quadtrees, kd-trees (Ooi, Tan, 1997) or other methods for spatial objects, or more accurately spatial points. Kd-trees are related to binary trees. A reference point is taken, along with a reference dimension. Every other point that falls below the reference point for this dimension shall branch to the left, all points with higher values branch to the right. At the next level, a new reference point is taken in each branch and the next dimension is used as a reference. It is relatively fast, can be updated on-line but the final shape of the tree depends on the insertion order of data, which can lead to unbalanced trees.

Another widely accepted approach is to use rectangles, bounding structures to match the position of objects. The bounding rectangles can then be regrouped within bigger rectangles so as to create a balanced tree. The R-tree, and its sibling R*-tree (Ooi, Tan, 1997) are examples of this. While the R-trees allow to work with complex objects (approximated as rectangles and not points), their higher building and querying time make the use of lighter structures appealing to index points.

3.3 Temporal Approach

For the temporal approach, it is important to note that different notions of time can be used for databases (Ooi, Tan, 1997*). The Transaction Time allows users to perform "rollbacks" so as to find past-values. It does not allow to modify previously entered values, nor to enter future values. One can only append new data issued at the present moment. The Valid Time represents the time when a fact is considered true. It allows users to modify past data, and to enter future data. However, it does not allow rollbacks. The Bi-Temporal Time is a mix between the two others, allowing rollbacks, post-modifications and future updates.

There mainly are two ways of considering temporality. The latter is to consider that time is monotonous (time goes in one direction) and to use B-trees as index structures. An interesting variation of this is to consider that the data flow constantly, therefore it becomes possible to link the root of the tree more closely the last leaf. This leaf containing the most recent data. Such an idea has been developed in AP-trees (Gunahdi, Segev, 1993).

The other way of considering temporality is to consider time just as a spatial dimension and to use R-trees, with on one dimension the time-stamps and on the other dimension the validity duration.

3.4 Spatio-Temporal Approach

Spatio-temporal approaches have to face the variety of possible types of data: points, ranges, intervals (Wang, Zhou, Lu, 2000). This leads to a distinction between three families of indexing trees. Those that work with objects in continuous movement, those for discrete changes and finally those for continuous changes of movements. Another way of differentiating the families of index has been brought by (Mockbel, Ghanem, Aref, 2003). They have focused on approaches aiming at indexing past positions, present ones and future ones.

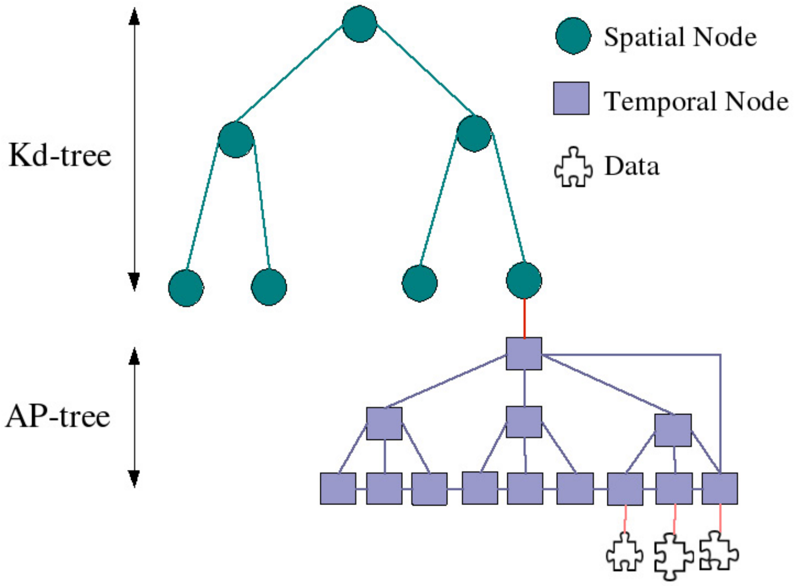
Many trees have been developed to answer specific needs. Some trees tend to consider the temporal aspect as yet another spatial dimension, which has led to 3DR-trees (Theodoridis, Vazirgiannis, Sellis, 1996). However, these trees doesn't take into account the monotonicity of time and usually need to have a previous knowledge of the data to index. Another family of trees make a difference between spatial and temporal dimensions. In HR-tree (Nascimento, 1998), typical of this case, snapshots of spatial R-trees are linked in a time indexed balanced tree. The main problem of this kind of trees is the size of the tree. While nodes that do not change between two snapshots are shared among the R-trees, only minor changes force to duplicate some of the data. Furthermore, they tend not to be optimal for interval queries.

Yet, from these aspects we can take some ideas. First of all, the differentiation of temporal and spatial data can be used to segment the tree. Then, a variation of B+-tree, the AP tree , offers the idea of a direct link between the root of the temporal data and the latest node. All of these ideas have been associated to provide our solution, the Po-tree.

3.5 Specification of a Real-Time Indexing Structure

Our first solution, the Po-tree, is based on the differentiation of temporal and spatial data, with a focus given to the latter. This way the notion of information sources is linked to a specific spatial location. It has been devised so as to deal with volcanic activity monitoring, which involves a number of different sensors with measurement frequencies going as high as 100 Hz. The spatial aspect is indexed through a Kd-tree, while the temporal aspect uses modified B+-trees (see Figure 3). As for now, mobility is not managed by the structure. However the specificities of both of these trees allow on-line and batches updates: it is possible to update the structure on real-time or using batch files.

The monitoring stations being immobile, this structure does not allow mobile sources of information. This way, every spatial location, akin to spatial object (sensor) is directly linked to a specific temporal tree. Requests shall first determine the spatial nodes concerned and later on deter-



mine the temporal nodes.

Fig. 3. Po-tree structure

3.6 Po-Tree Structure

Kd-trees are simple structures, but they are not perfect structures. One of their main problem being the fact that they rely on the order of inserted data. If the data are entered in different orders, the final trees may have different shapes. Another issue is the fact that they are not perfectly suited for mobility, which is not part of our needs so far. However, ICC methods, originally designed for B-trees can easily be adapted to cover Kd-trees. Different tests have also proven that these trees fared reasonably well compared to R-trees for small number of data (Paspalis, 2003). As each B+-tree is linked to an object, it is possible to develop a secondary structure so as to access directly the temporal data of specific objects, without the need to first determine their position. This can be useful for the notion of hierarchy of information sources.

Furthermore, it has been noticed that the most recent data are considered of higher interest than the older one. It has also been noticed that inserts are generally held at rightmost of the structure, where are found the newest nodes. Therefore, the temporal tree has been modified to add a direct link between the root and the latest node. While maintaining this link requires minimum work for the system, a simple test prevents being forced to traverse the whole tree so as to append or to find the requested data. This direct link is useful to save processing time.

As most, if not all, of the updates take place to the rightmost part of the temporal tree, the fill factor of leaf nodes can be placed higher than usual. Deletes should be somehow rare under normal conditions, and updates that does not concern the newest data should be even rarer, unless the systems experiences lag time due to network problems between the sensors and the database. Therefore the split and merge procedures can be changed so that the nodes can be filled almost at their maximum capacity.
This configuration implies that this 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 take any advantage of the specificities of the tree.

3.7 Tests

Different tests have been conducted between the Po-tree, Pas-tree and R*-tree structures (thanks to Hadjieleftheriou's implementation, Hadjieleftheriou 2004). 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 (10-100) used and the portion of the base to scan for interval queries. The tests have been conducted on a 1.6 G Hz, 128 Mo RAM computer, running Linux. The programming language used was Java.

Due to the differentiation of spatial and temporal component, and due to the fact that the data were coming from a finite set of spatial points, the Po-tree built time has been greatly reduced compared to the R*-tree. While 25 000 points stemming from 100 different locations were indexed in less than one second with the Po-tree, it took nearly 45 seconds with a R*-tree. Other tests have shown that the construction time of the Po-tree evolved linearly with the number of stations, the number of different spatial locations...

The different queries have shown interesting properties as well. The interval queries took an advantage of the linking of the temporal nodes of the Po-tree. For point-interval queries, the Po-tree can be up to 8 times faster the the R*-tree. While for interval queries the difference has shown much lighter, it still remains in favour of our solution, as shown in Figure 4. On this figure, the last 10% of the entered data where fetched. The spatial range covered the whole possible locations. It is visible than for a low number of data the R*-tree fares better, yet when the amount of data rises past 6000, it is the Po-tree that gains the advantage.

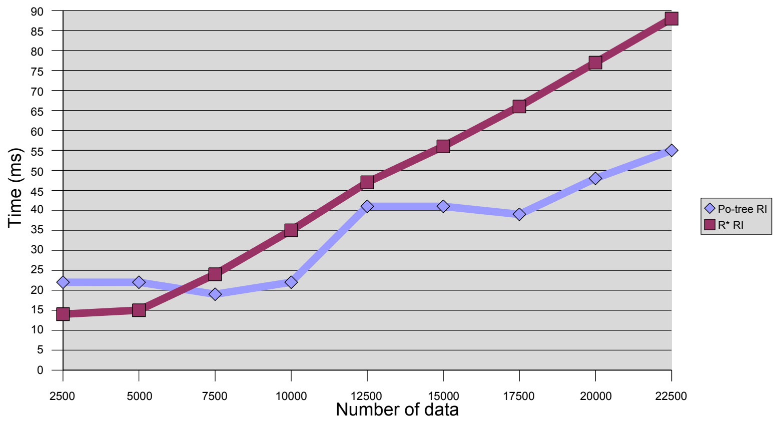


Fig. 4. Range-Interval queries

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.

3.8 Managing the Sensors Positions Changes: the Pas-Tree

While the first structure uses two structures to index spatio-temporal data, it does not allow sensor position changes. If a position was to change, a new temporal sub-tree would have to be created. This could be troublesome if a specific location becomes irrelevant, or in order to index some specific data. For example, seismic epicenters are usually determined by specialized processes but can be considered as indexable data. Those epicentres can be considered as data collected from evolving sensors. Thus the need to improve the Po-tree, so as to create the Pas-tree. This tree has to deal with position changes, yet remain focused on prioritizing the newest data and update transactions.

Sensors can move from time to time. The spatial sub-tree should be able to track these modifications. Different approaches exist, yet we shall aim at introducing a multiversion approach. A given node records the presence of past sensors (with an end-time) and of actual sensors. So as to offer other querying options, quadtrees could be used instead of Kd-trees.

The temporal sub-trees should also keep tracks of the position changes. Each of these sub-trees is related to a specific sensor. Keeping track of their position allows to follow the movement of sensors through a time interval without querying the spatial sub-tree. The temporal sub-trees of the Po-tree can be suitably adjusted to differentiate two kinds of entries: measurement data and position changes.

A tertiary structure, based on B+ tree keeps record of the sensor IDs. As a matter of fact, some queries do not need spatio-temporal properties. Scientists have grown used to sensor identifiers, and does not always rely on spatio-temporal properties. Even more so when mobility is scarce. Therefore this structure links directly to the temporal, sensor related sub-trees without using the spatial sub-tree.

The Pas-tree suffers from data duplication, yet it also allows for more request types than the Po-tree. As a matter of fact, queries can be based on the sensor ID as well as on spatio-temporal properties. It allows users to follow a specific sensor through its position changes or to have a look at different sensors passing through a region during a lapse of time. It stills focuses on update transactions and on the newest data.

4 Conclusions

As a conclusion, let us advocate for developing soft real-time GIS for disaster management, or more especially for disaster monitoring. Before reaching this goal, several barriers must be overcome such as robustness and fast storing which implies fast indexing.

References

Barbara D (1999) Mobile Computing and Databases - A Survey. In: IEEE Transactions on Knowledge and Data Engineering, vol. 11 (1), pp 108-117

Behr FJ (1995) Mobile GIS: Contributing to Corporate Benefits. DA/DSM Seminar, 20 November 1995, Rome, Italy http://www.graphservice.de/papers/mobile_g.htm

Buelher K, Mc Kee L (eds) (1996) The OpenGIS Guide, Introduction to Interoperable Geoprocessing. Open GIS Consortium. http://www.opengis.org

Gunadhi H, Segev A (1993) Efficient indexing methods for temporal relation, In IEEE Transactions on knowledge and Data Engineering, 5(3), pp 496-509

Guting RH, Bohlen MH, Erwig M, Jensen CS, Lorentzos NA, Schneider M, Vazirgiannis M (2000) A Foundation for Representing and Querying Moving Objects. ACM Transactions on Database Systems, Vol. 25 (1) pp 1-42

Hadjieleftheriou M, Spatial Index Library [Online], viewable at: http://www.cs.ucr.edu/~marioh/spatialindex/, (last consulted 01/11/04)

Haritsa JR, Seshadri S (2001) Real-time index concurrency control. In real-time Database System – Architecture and Techniques, Kluwer Academic Publishers, Boston, ISBN: 0-7923-7218-2, pp 60-74

Lam KY, Kuo TW (2001) real-time database systems: an overview of systems characteristics and issues. In real-time Database System – Architecture and Techniques, Kluwer Academic Publishers, Boston, ISBN: 0-7923-7218-2, pp 4-16

Laurini R (2001) Information Systems for Urban Planning: A hypermedia Cooperative Approach. Taylor and Francis, Forthcoming February 2001. See http://lisi.insa-lyon.fr/~laurini/isup

Laurini R (2001) real-time Spatio-Temporal Databases. In "Transactions on Geographic Information Systems", Guest Editorial, Vol 5(2), pp.87-98

Laurini R, Gordillo S (2000) Field Orientation for Continuous Spatio-temporal Phenomena. International Workshop on Emerging Technologies for Geo-based Applications, Ascona, Switzerland, May 22-26, 2000. Edited by S. Spaccapietra, published by the Swiss Federal Institute of Technology at Lausanne, pp 77-101

Laurini R, Milleret-Raffort F (1990) Principles of Geomatic Hypermaps. In: 4th International Symposium on Spatial Data Handling. Zurich, 23-27 Juillet 90. Edited by K. BRASSEL, pp 642-651

Mokbel MF, Ghanem TM, Aref WG (2003) “Spatio-temporal Access Methods”, IEEE Data Engineering Bulletin, Vol 26, n°2, pp 40-49

Mockbel MF, Xiong X, Aref WG, Hambrusch S, Prabhakar S, Hammad M (2004) PLACE: a Query Processor for Handling Real-Time Spatio-Temporal Data Streams, In VLDB 2004

Nascimento MA, Silva JRO (1998) Towards Historical R-trees, In: Proceedings of ACM Symposium on Applied Computing (ACM-SAC) Atlanta, USA, pp 235-240

Noel G, Servigne S, Laurini R (2004) "Real-time spatiotemporal data indexing structure", Proceedings of 2004 AGILE 7th conference on Geographic Information Science, Heraklion, 2004, pp. 261-268

Ooi BC, Tan KL (1997) Spatial Databases. In Indexing Techniques for Advanced Database Systems, Kluwer Academic Publishers, Boston, ISBN 0-7923-9985-4, 39-75

Paspalis N Implementation of Range searching Data-Structures and Algorithms [Online], viewable at: http://www.cs.ucsb.edu/~nearchos/cs235/cs235.html, (last consulted 20/11/03)

Prichard J, Fortier P (1997) Real-Time SQL. In: Second International Workshop on Real-Time Databases September 18-19, 1997 Burlington, Vermont, USA, pp. 289-310

Satnam A, Agogino A, Morjaria M (2001), A methodology for intelligent sensor measurement, validation, fusion and fault detection for equipment monitoring and diagnostics, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol 15, No 4

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 (ICMCS)

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

Wolfson O (1998) Moving Objects Databases: Issues and Solutions. In: the Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM98), Capri, (Italy), July 1-3, 1998, pp. 111-122.