Indexation multidimensionnelle de bases de données capteur temps-réel et spatiotemporelles

G. Noël, S. Servigne

LIRIS, INSA Lyon
Bat. B. Pascal, 20 av. A. Einstein 69622 Villeurbanne Cedex
e-mail: {Noel.guillaume; Sylvie.servigne}@insa-lyon.fr

RÉSUMÉ. *Les systèmes de base de données capteurs sont de plus en plus fréquemment utilisés pour la surveillances de milieux à risques. Ces systèmes sont usuellement composés d'un ensemble de capteurs envoyant les mesures effectuées vers une base de données centralisée. La fréquence des mesures aussi bien que les besoins des utilisateurs imposent à la base des contraintes temps-réel douces, tout en valorisant les données les plus récentes. Quant au besoin semi-généralisé d'accéder aux données en fonction de critères spatiaux, il impose à son tour des caractéristiques spatiotemporelles. Afin de répondre aux spécificités de ces systèmes, cet article propose deux méthodes d'indexation de données. La première, dédiée à l'indexation d'un grand nombre de données issues de capteurs fixes se nomme le PoTree. Son évolution, le PasTree, privilégie la gestion de l'agilité des capteurs et l'augmentation des méthodes d'interrogation.*

ABSTRACT. *More and more risk monitoring systems use sensor databases. These systems usually consist of an array of sensors sending their measurements toward a central database. The measurement frequency as well as the user requirements impose soft real-time constraints to the database, and tend to set the focus on the newest data. As for the need to access the data through spatial criterium, it adds spatiotemporal specificities. So as to meet these requirements, this paper propose two data access methods. The first one, dedicated to systems with a high number of data from fixed sensors is named the PoTree. Its evolution, the PasTree, focuses on sensor agility management and adding new querying patterns.*

MOTS-CLÉS : *Système d'indexation, Base de données capteur, Spatiotemporel, Temps-réel*

KEYWORDS: *Indexing system, Sensor database, Spatiotemporal, Real-time*

1. Introduction

Les systèmes de base de données capteurs sont de plus en plus fréquemment utilisés pour la surveillances de milieux à risques aussi bien que pour la prévention de catastrophes naturelles. Ces systèmes sont usuellement composés d'un ensemble de capteurs envoyant les mesures effectuées vers une base de données centralisée en mémoire vive. La fréquence des mesures aussi bien que les besoins des utilisateurs imposent à la base des contraintes temps-réel douces, tout en valorisant les données les plus récentes. Quant au besoin semi-généralisé d'accéder aux données non plus en fonction d'un identifiant de capteur mais en fonction de critères spatiaux, il impose à son tour des caractéristiques spatiotemporelles. Afin de répondre aux spécificités de ces systèmes, cet article propose deux méthodes d'indexation de données en mémoire vive. La première, dédiée à l'indexation d'un grand nombre de données issues de capteurs fixes se nomme le PoTree. Son évolution, le PasTree, privilégie la gestion de l'agilité des capteurs et l'augmentation des méthodes d'interrogation au détriment du nombre de capteurs indexables.

Cet article se compose de quatre ~~grandes~~ parties. La première présente les particularités des bases de données capteurs et se conclut par une présentation de notre cas d'application. La seconde partie offre un état de l'art des domaines temps-réel et spatiotemporels. La troisième entre dans le détail du PoTree, offrant une description et des tests de la structure. La partie suivante s'attache à son évolution, le PasTree, en suivant le même schéma.

2. Présentation générale

Les bases de données capteurs sont de plus en plus régulièrement utilisées dans des domaines tels que la surveillance de phénomènes naturels. Bien que chacun de ces cas apporte son lot de spécificités propres, il reste cependant possible de noter certaines similitudes. Ces similitudes donnant à leur tour naissance à des particularités rencontrées dans la majorité des systèmes de ce type.

En premier lieu, on retrouve le plus souvent un ensemble de capteurs hétérogènes, mesurant différents paramètres: température, pression, clinométrie,~~...~~ etc. Alors que certains capteurs sont purement périodiques (sismographes), d'autres n'envoient des mises à jours que lorsqu'un certain seuil de mesure est dépassé ou

lorsqu'une variation de valeur est constatée. D'autres politiques combinent ces différents modes de fonctionnement. De là notre première idée: les données peuvent être hétérogènes.

Un autre point à noter vient de la représentation des capteurs. Alors que historiquement les capteurs sont représentés de par leur identifiant de capteur, avec l'apparition de technologies permettant une mise à jour des informations de positionnement, il devient possible de représenter les capteurs en fonction d'informations spatiales déterminées via GPS ou triangulations diverses.

Une fois que les capteurs ont effectué leurs mesures, le problème du stockage de données se pose. Bien que certaines recherches préconisent l'utilisation de techniques de stockage au niveau des capteurs eux-mêmes, ou au niveau de capteurs intermédiaires, force est de constater que pour l'heure l'immense majorité des réseaux de capteurs utilisés envoient les données vers une base de donnée centralisée. Ce n'est que par la suite qu'il peut y avoir réplication de données vers d'autres bases ou des entrepôts de données en dehors du réseau.

Les données en elles-même sont référencées en fonction de l'instant de la mesure et d'informations relatives à l'identification du capteur, que ce soit spatialement ou au travers d'un identifiant. Par la même, il apparaît~~appert~~ que les données sont multidimensionnelles.

De plus, les prises de mesures imposent à la base centrale des contraintes temps-réel. En effet, il est nécessaire qu'une mesure provenant d'un capteur donné soit intégré à la base avant qu'une mesure plus récente n'apparaisse dans celle-ci. Ou encore, il est parfois nécessaire de consulter les données dans des délais impartis.

Enfin, un autre point important vient du fait que ce genre de système met souvent l'accent sur l'importance des données les plus récentes. Bien qu'il soit souhaitable de conserver un long historique des mesures passées (afin de pouvoir comparer, retrouver des schémas similaires ou autre), les spécialistes sont le plus souvent principalement intéressés par les données les plus récentes. Ainsi, au sein de l'immense quantité de données recueillies (certains capteurs peuvent avoir des fréquences s'exprimant en centaines de hertz, voire plus), il est généralement souhaité de pouvoir mettre en avant les mesures reflétant l'état actuel du système.

Finalement, on retrouve des spécificités communes aux différentes bases de données capteur dans deux domaines: le spatiotemporel et le temps-réel.

2.1 Spécificités spatiotemporelles

L'utilisation de données multidimensionnelles a diverses implications spatiales et temporelles autant que sémantiques.

On considère dans cet article les capteurs prenant des mesures et où les valeurs de ces mesures ont une valeur aussi si ce n'est plus importante que la position du capteur aux yeux des spécialistes. Il est possible de diviser les systèmes spatiotemporels en trois catégories: les systèmes basés sur des valeurs discrètes, des systèmes basés sur des valeurs continues et des systèmes basés sur les variations de valeurs continues (Wang, Zhou & Lu, 2000). Dans le cas de capteurs prenant des mesures à période déterminées, on se ~~fixe~~situe dans le premier cas. Les problématiques de la gestion de flotte, non spécifiques à notre cas, relèveraient de la troisième catégorie.

Dans la mesure où les capteurs peuvent être référencés spatialement, il devient envisageable de lier chaque capteur à un emplacement précis, correspondant à sa position ou à celle de la station qui l'habrite. Dans de nombreux réseaux de détection actuels il existe des listes associant des sismographes ou d'autres capteurs fixes à une position précise (ex: « PPJ, sismographe triaxial, latitude 19.0342N, longitude 98.6446O »). Ceci suppose que les capteurs soient fixes. Dès lors, il est possible de représenter le système étudié par une représentation bidimensionnelle~~2D~~ ou ~~3D~~tridimensionnelle, avec les capteurs représentés par des emplacements précis de cet espace.

Historiquement, bon nombre de réseaux de capteurs utilisaient des capteurs imposants et immobiles. Avec l'apparition de technologies sans-fils, des capteurs plus légers et mobiles ont commencé à apparaître. Notre étude se concentre principalement sur les capteurs effectuant des mesures depuis des positions fixes dans un premier temps. Dans un second temps, il a été décidé d'aborder la notion d'*agilité* des capteurs. C'est à dire leur propension à changer de position entre deux prises de mesures. Cependant, l'accent n'a pas été mis sur la gestion de flotte. Alors que l'information apportée par le changement de position peut être important, la valeur mesurée est considérée ici comme prépondérante. Ainsi donc, si un capteur fixe possède une agilité nulle et un capteur mobile une agilité de l'ordre de 1 (constamment en mouvement), un capteur agile peut avoir une agilité de l'ordre de 0,01 à 0,5. Cela correspond alors à des stations portables qui sont déployées le temps d'effectuer quelques relevés avant d'être déplacés.

Puisque les mesures sont temporellement définies par l'instant de la prise de mesure (*timestamp*), les données sont également temporelles. Les données d'un capteur particulier peuvent être ordonnées en fonction de ces marqueurs temporels. Ceci permet d'obtenir un historique des mesures prises, utile pour une analyse permettant de pronostiquer l'évolution à venir du système, à condition que celui-ci suive certaines lois.

Un dernier point vient de la nature des données. Alors que la grande majorité des données provient des mesures de capteurs, d'autres peuvent être extrapolées à partir de celles-ci. Ainsi, un processus défini peut utiliser un mécanisme de triangulation pour déterminer l'épicentre d'un phénomène à partir de mesures prises par ~~deux~~différents capteurs. Ces valeurs calculées peuvent être considérées comme étant indépendantes les unes des autres ou bien comme étant issues d'un même capteur *agile*, c'est à dire pouvant changer de position entre deux mesures. De même, la différenciation effectuée entre les différentes sources d'information (types de capteurs) ou de données (mesures, calculs) permet de déterminer des propriétés sémantiques. Ces propriétés se couplent aux propriétés spatiotemporelles.

En plus de ces caractéristiques liées aux particularités multidimensionnelles des données viennent se greffer des aspects temps-réels, du fait des~~de par les~~ fréquences de mise à jour ou de par des impératifs formulés par les utilisateurs.

2.2 Spécificités temps-réel

Les contraintes temps-réel auxquelles doit répondre la base de donnée sont de deux types. En premier lieu, elle doit être à même d'indexer et de stocker une mesure avant qu'une version plus récente de celle-ci n'arrive. On obtient ici des contraintes temps-réel au niveau de la gestion des données entrantes, pour éviter une saturation de la base. Généralement, pour les mesures périodiques ou pseudo-périodiques, les spécialistes s'accordent à admettre qu'il est permis d'écarter certaines mesures si elles n'apportent que peu d'information par rapport aux mesures précédentes et si elles ne sont pas caractéristiques d'une activité particulière. On aborde dès lors la notion d'epsilon-données (si le taux de variation entre deux mesures est inférieur à un seuil déterminé, on abandonne la seconde mesure pour gagner du temps de calcul et des ressources). Certaines mises à jours peuvent être écartées si la variation de données par rapport à la mesure précédente est inférieure à un seuil.

Il existe également un autre facteur temps-réel. En fait, dans les systèmes de surveillance basés sur des capteurs, l'un des principaux buts visés est l'aide à la décision. En période de crise, ou tout du moins d'activité anormale, les données idoines doivent être fournies aux décideurs selon des délais impartis (en cas d'incendie: force des vents, taux d'humidité dans l'air, etc.). La notion de délais impartis nous amène donc naturellement vers la notion de requêtes temps-réel. Ainsi, pour ces mêmes raisons de délais il convient de différencier d'une part la base de données qui collecte les mesures issues des capteurs des systèmes d'entrepôts de données qui sont chargés de collecter ces mesures sur de plus longues durées. Alors que les entrepôts de données peuvent utiliser un ensemble de méthodes de stockage de données variées, la base doit minimiser les coûts d'accès aux informations. Par conséquent, le consensus actuel privilégie l'utilisation de bases de données contenues en mémoire vive (pour limiter les coûts des accès disques). Nos travaux se sont axés sur cette base de données.

Ces différentes caractéristiques se retrouvent dans nombre de système d'observation de milieux particuliers, ou de surveillance de phénomènes. Plus particulièrement, nos travaux se sont concentrés sur un exemple concret, la surveillance volcanique.

2.3 Un exemple de cas d'application: la surveillance volcanique

Le Popocatépetl est un volcan mexicain, à 60 kilomètres de Mexico, 45 km de Puebla. Il s'étend sur trois états mexicains, chacun ayant sa propre politique de gestion des risques. Afin de prévenir les catastrophes naturels, le volcan a été placé sous la surveillance du Cenapred (Cenapred, 2005), centre national de prévention des désastres. Le Cenapred utilise un ensemble de capteurs répartis dans 25 stations (séismographes, clinomètres, etc.~~...~~) permettent de qualifier et quantifier l'activité du volcan (figure 1). Ces données sont collectées au sein d'une base de données centralisées. Pour l'heure, le système se base sur les spécifications Earthworm, définis par l'USGS (Usgs, 2005), ~~le pendant américain~~collaborateur historique du Cenapred. Cependant, Earthworm ne met pas l'accent sur les données spatiales et classifie les données par l'identifiant du capteur les ayant prises. Il a été décidé de changer d'optique pour se rapprocher de problématiques spatiotemporelles.

Fig. 1 : *Stations de mesures du volcan Popocateptel*

Usuellement, les spécialistes (chimistes, physiciens, preneurs de décision) admettent que les données les plus récentes sont également pour eux les plus intéressantes, hormis les données caractéristiques de phases spécifiques d'activité. En outre, en raison de la saisie continue des mesures, la base de données se doit d'utiliser des mécanismes mettant au premier plan l'importance des nouvelles données. Cette remarque est toute particulièrement vraie pour les méthodes d'indexation. Alors que le mode de requêtage habituel de la base se fonde sur les identificateurs des différents capteurs, les scientifiques demandent à présent la capacité d'utiliser des requêtes spatio-temporelles. Ils souhaitent être à même de retrouver les données émises pendant un intervalle de temps spécifique, dans une certaine zone spatiale.

3. Etat de l'art

Les aspects spatiotemporels et temps-réel ont été explorés, chacun de leur côtés par différentes équipes. De ces travaux découlent certaines caractéristiques. Aussi, après avoir abordé les aspects temps-réel il convient de s'attarder sur les indexation spatiales puis temporelles et enfin spatiotemporelles. Ces différentes approches devant enfin amener à des travaux liés à la localisation des données dans un système basé sur un réseau de capteurs.

3.1 Temps-réel

Le principal but des systèmes temps réel est de respecter des impératifs temporels. Pour une base de données, cela signifie que les transactions doivent s'effectuer dans un certain délai imparti (Lam, 2001). Si ces contraintes ne peuvent être respectées, les conséquences peuvent s'avérer désastreuses pour le système. Toutes les transactions n'ont cependant pas les mêmes limitations temporelles (*deadlines*).

Il en existe principalement trois types, selon (Lam, 2001). Il est possible de définir une courbe de valeur pour déterminer l'effet de la transaction sur le système global (tant que la valeur est positive, l'effet est positif ou sans conséquence notable, si elle passe négative, l'effet est néfaste). Tant que la transaction se trouve avant la fin du délai qui lui a été imparti, la courbe prend une valeur constante positive non nulle. C'est lorsqu'une transaction ne peut pas respecter ses contraintes temporelles que le comportement de la courbe varie.

Les limites douces (*soft deadlines*) sont les moins restrictives. Il est permis qu'une transaction ne respecte pas ces limitations. Dans ce cas, à la fin du temps imparti à la transaction, la courbe de valeur de celle ci décroît plus ou moins doucement jusqu'à zéro.

Les limites fermes (*firm deadlines*) sont plus restrictives. Il est éventuellement possible à une transaction de dépasser sa limite, mais ce n'est absolument pas souhaitable. Dans ce cas, la fonction de valeur devient nulle dès la fin du temps imparti.

Les limites dures (*hard deadlines*) enfin ne doivent pas être ratées. Tout retard deviendrait pénalisant pour le système. La fonction de valeur tend vers l'infini négatif dès la fin du temps alloué à la transaction.

Les principaux travaux dans ce domaine, proposent de donner des priorités aux transactions afin de les ordonnancer avant l'exécution. On pourra utiliser pour cela des algorithmes comme *Earliest Deadline First* ou *Rate Monotonic*, en fonction des contraintes du système (Lam, 2001).

Pour ce qui est de l'exécution, les problèmes peuvent surgir si une transaction de faible priorité s'exécute quand une transaction de haute priorité arrive (« inversion de priorité »). Dès lors, on peut soit mettre en attente la transaction de faible priorité, soit l'annuler et la faire recommencer pour laisser la place à celle de haute

priorité (Haritsa, 2001). D'autres techniques permettent de trouver un compromis entre ces deux solutions.

3.2 Indexation spatiale

Il existe plusieurs grandes familles d'indexation spatiale ou temporelle. Un aperçu permet d'en mettre en avant certaines.

3.2.1 Indexation spatiale basée sur les binary tree

Le binary-tree est utilisé pour indexer des données de manière à ce que les valeurs de l'index soient ordonnées de manière linéaire. Pour cela, on va subdiviser l'espace de données afin de créer l'arbre. Cette idée a été reprise dans de nombreux types d'index. Entre autre, on peut citer les kd-tree, kdB-tree, hB-tree, BD & GBD-tree, LSDh-tree, skd-tree, etc.... (Bertino, 1997) On s'intéressera plus précisément au kd-tree (Bentley, 1975).

Le kd-tree permet la recherche binaire d'ordre k. Il représente une subdivision récursive de tout l'espace via des plans iso-orientés. Chaque séparation doit contenir au moins un point, utilisé dans la représentation de l'arbre. Ce point devient alors le point de référence pour subdiviser l'espace, en fonction d'un critère (un axe, alternativement vertical et horizontal pour un espace deux dimensions). Les points inférieurs à celui de référence iront à gauche dans l'arbre, ceux supérieurs, iront à droite. Les nœuds internes peuvent avoir un ou deux descendants.

La recherche est simple, mais la suppression compliquée du fait de la réorganisation de l'arbre. De plus, on notera que la structure de l'arbre dépend de l'ordre d'insertion des données.

Le coût de construction de l'arbre est de l'ordre de O(n log n), pour n le nombre de points, mais la recherche d'une fenêtre spatiale est de l'ordre de O (sqrt(n)+k) avec k le nombre de points retournés.

3.2.2 Indexation spatiale basée sur les B-tree

Ici, on utilise généralement la notion de boite de recouvrement minimum (MBR en anglais). Un objet est contenu dans un MBR qui est lui-même contenu dans un autre MBR, etc.. On obtient alors une hiérarchie. Au sein de ces arbres, on peut citer les R-tree (Gutman, 1984), R* & R+-trees, BV-tree (Bertino, 1997), DR-tree

(Lee, 2001), Pyramid-tree, SS & SR-trees, TV-tree, X-tree (Berchtold, ~~2001~~1996), XBR-tree (Vassilakopoulos, 1999), etc.~~...~~

3.2.3 Autres types d'indexation spatiale

Les architectures vues ci-dessus ne sont pas les seules existantes. En effet, on peut également citer les méthodes de hachage dynamique (Grid File, Plop Hashing, etc.~~...~~), ou encore celles basées sur les quadtrees, ou octrees pour un espace en trois dimensions (Bertino, 1997).

Le principe du quadtree est de diviser récursivement l'espace en quatre quadrants, appelés le plus souvent Nord-Est, Sud-Est, Sud-Ouest et Nord-Ouest. Les points viennent s'insérer dans ces quadrants qui sont récursivement re-décomposés en sous-quadrants si le nombre de points du quadrant est trop important. Cette structure est appliquée avec un certain succès aux images *raster* et est également assez bien adaptée à l'indexation de points. Le coût de construction du quadtree est de l'ordre de O(n log n).

3.3 Indexation temporelle

On peut diviser les index temporels en deux grandes familles. La première tend à considérer la dimension temporelle comme une autre dimension spatiale, et par conséquent utilise les index spatiaux. L'autre famille se base sur des B et B+-tree. Au sein de cette dernière on pourra citer les Append-Only-tree , B+-tree avec ordre linéaire, Interval B-tree, Multi Version B-tree, NST-tree, Time Index, TSB-tree. (Bertino, 1997) On s'intéressera plus particulièrement à l'Append Only-tree.

Dans cet arbre, les nœuds feuille conservent les temps de début des relations temporelles. Les nœuds non-feuille conservent des pointeurs vers des valeurs temporelles pointant vers des enfants où cette valeur est la plus petite valeur (sauf pour les premiers enfants).

Tous les ajouts se font à droite de l'arbre. Tous les sous-arbres sont pleins, sauf le dernier. Un *split* amène à un nouveau nœud à remplir. Mais l'arbre n'est pas équilibré car le nouveau nœud est mis où on peut le mettre…

La recherche peut être assez longue. Pour trouver un intervalle, on prend le temps de fin, et on remonte vers la gauche. Les ajouts sont monotones, avec des valeurs temporelles incrémentielles, ce qui peut limiter les mises à jour. Cet arbre trouve son efficacité dans les opérations sur les données les plus récentes.

3.4 Indexation Spatio-temporelle

Wang, Zu et Lu (Wang, 2000) définissent 3 types d'applications spatio-temporelles :

-aApplications avec des objets en mouvement continu

-aApplications avec des changements discrets

-aApplications avec des changements continus de mouvement.

Dans notre cas d'étude, nous nous concentrerons sur le second type d'application.

Historiquement, le domaine de l'indexation spatio-temporelle a principalement été poussé par la recherche dans les bases de données spatiales, alors plus avancées. Les arbres les plus connus sont les 3D R-tree (GutmanThéodoridis, 1996, 1984), HR-tree (Nascimento , 1998), TR-tree et MV3R-tree (Tao, 2001). Mais on pourra aussi noter les HR+-tree (Tao, 2001), MultiVersion Linear Quad-tree (Tao, 2000), SEB-tree (Song, 2003), etc.... Une présentation plus complète a été fournie par (Mockbel, 2003). On s'intéressera principalement au 3D R-tree et au HR-tree, illustrant deux grandes familles d'arbres..

Dans un 3D R-tree, les intervalles de temps et les clefs (spatiales) forment un point dans un espace en trois dimensions3D (clef ;Tdébut ;Tfin). Ceci permet de gérer les requêtes de temps, de clef et temps, et de clef.

Cette approche souffre cependant de différents problèmes. En effet, la gestion des données les plus récentes peut poser problème car on doit alors définir une zone de requête non fermée (non-encore finie). De même, cette approche n'est pas optimale pour gérer les objets en mouvements (création de boites trop grandes dans les approches continues, perte d'information dans les approches discrètes).

Enfin, on notera que les recherches dans le 3D R-tree prennent en compte l'intégralité des entrées, étaient aussi d'autant plus longues que le nombre de données est important. Il en ressort donc que le coût de construction de l'arbre augmente avec le nombre de dimensions. D'une manière globale cependant, si cet index est efficace pour les requêtes d'intervalle, il ne l'est pas pour les requêtes de *timestamp*, de points temporels.

Le principe de fonctionnement du HR-tree est de conserver les différents états du système (via les R-trees) à des temps différents. Les branches qui n'ont pas évolué dans les différents R-tree sont communes.

Le HR-tree est meilleur que le 3D R-tree pour les recherches de *timestamp* et les petits intervalles. On notera cependant qu'il implique la duplication d'objets, et a de relativement mauvaises performances dans les requêtes d'intervalle. De plus, il n'est pas recommandé de l'utiliser dans des cas de mises à jour importantes, comme dans notre cas. La taille de l'arbre devient très vite importante, ce qui est une gèêne pour le respect des contraintes temps-réel rencontrées.

Un dernier arbre mérite qu'une attention particulière lui soit accordée: le Q+R tree (Xia, 2003). Cette structure vise à indexer les données issues d'objets mobiles. Il se différencie d'autres approches par la constatation que bon nombre d'objets mobiles alternent des étâts quasi-statiques et d'autres de grande mobilité. Ainsi les auteurs de cette structure proposent d'utiliser conjointement un R*-tree pour indexer les données quasi-statiques et un quadtree pour les données fortement mobile. Il convient alors de pouvoir déterminer précisément le seuil de mobilité idoine afin de bénéficier des capacités optimales de cette structure. De plus, une fois encore cette structure se focalise sur la mobilité d'un objet et sur la topologie. Elle n'offre pas une solution centrée sur les objets eux-mêmes, ni sur leur infrastructure.

3.5 Infrastructure de données et réseau de capteurs

Du fait du~~De par le~~ nombre de capteurs et leurs capacités de calcul, de stockage et de transmission réduite, les problèmes de localisation et de mise à jour des données en temps réel ne sont pas à négliger. Pour les réseaux se composant de micro-capteurs, la transmission de données en continu est généralement irréalisable, pour des considérations de coût de transmission. Diverses études tendent à montrer à ce sujet que l'agrégation locale des données avant transfert est généralement plus économique, malgré les faibles ressources disponibles au niveau des capteurs (Intanagonwiwat, 2001) (Ratnasamy, 2002). Cependant, se pose alors le problème de la validité des mesures disponibles, puisque l'utilisation d'agrégats sous-entend un décalage temporel entre le moment de collecte de l'information et celui où celle-ci sera disponible pour les utilisateurs, sur la base de données.

Pour les mêmes raisons de coût de l'information, plusieurs études se sont concentrées sur la localisation des données. On distingue principalement trois types d'approches (Eiman, 2003).

D'un coté les **approches locales** proposent de conserver le maximum de données au niveau local. Cette approche minimise le coût de transmission des mises à jour.

Cependant, pour les requêtes sur les données, il devient alors souvent nécessaire de propager la requête au travers les différents éléments du réseau, augmentant ainsi le coût des requêtes. De plus, dans des systèmes demandant de conserver un historique des données recueillies, il devient tôt ou tard nécessaire de transmettre les données sous une forme ou sous une autre.

A l'opposé, les **approches externes**, souvent retrouvées dans les systèmes d'information géographiques, proposent de centraliser les données dans une base à l'extérieur du réseau de capteurs. Dans ce cas de figure, le coût de transmission des mises à jour est bien évidemment largement supérieur. Cependant, cette approche permet de minimiser le coût de requêtage. Dans les systèmes d'aide à la décision, ou d'analyse de phénomènes comportant plusieurs variables interagissant entre-elles, le consensus actuel se veut favorable à la centralisation des données. De plus, cette approche offre une relative indépendance face aux problèmes de pannes de capteurs (Satnam, 2001). Si un capteur tombe soudainement en panne pendant une période critique, les données collectées par celui-ci qui ont été transmises à la base centralisée sont toujours accessibles. Enfin~~En point final~~, force est de constater que l'approche externe est pour l'heure actuelle l'architecture la plus répandue en matière de réseaux de capteur, malgré le nombre d'études théoriques sur d'autres modèles.

Entre ces deux extrêmes se trouvent diverses approches, offrant des systèmes de réplication des données entre différents nœuds du réseau. Il s'agit d'un compromis entre les deux approches. Cependant, cela pose différents problèmes. L'utilisation de techniques de localisation spatiale des données, le **stockage centré sur les données** permet de définir des nœuds de réplication dans le réseau où il devient possible d'accéder aux données. En répartissant les données entre différents points du réseau, il devient possible de chercher un compromis entre les différents coûts de requêtes (mise à jour / accès aux données). Pourtant la prise en compte de la mobilité impose de régler des problèmes de re-localisation de données, de détermination de nœuds de réplication, d'accès au réseau, etc.~~...~~ Et ce, en prenant en compte des restrictions temps-réelles.

3.6 Synthèse de l'état de l'art

Il ressort de cet état de l'art différents points notables. En premier lieu, il convient de rappeler que le temps-réel n'implique pas nécessairement la rapidité du calcul mais le respect de contraintes temporelles et que dans certains cas il est donc

préférable d'abandonner certaines transactions plutôt que d'induire une latence en cascade pour résoudre les transactions. Le temps-réel est également lié à la notion de priorité, des transactions jugées plus importantes recevant une priorité plus importante.

Pour ce qui est des aspects spatiaux, temporels ou spatiotemporels, on peut observer plusieurs tendances. En premier lieu, les travaux dans le domaine spatial, plus anciens ont utilisés des techniques permettant la réutilisation de structures déjà connues dans les bases de données classiques, via entre autre la linéarisation de l'espace. L'ajout majeur apporté par Guttman et ses R-tree a par la suite fortement contribué à l'utilisation de structures similaires dans les domaines temporels et spatiotemporels. Bien souvent, l'aspect temporel est considéré comme un aspect spatial supplémentaire. Seules quelques solutions permettent de différencier efficacement les deux. On notera cependant que plusieurs travaux permettent d'aborder une notion de source d'information, permettant de subdiviser l'espace d'origine en parcelles en fonction d'une zone précise ou bien d'un capteur déterminé.

Finalement, la structuration des données dans les réseaux de capteurs montre qu'il est souvent nécessaire de trouver un compromis concernant la localisation des données. Plus elles sont proches des capteurs, plus les mises à jours seront allégées. Cependant, dans ce cas, la mise en panne d'un capteur (ce qui est une éventualité forte dans le cadre de la surveillance de phénomènes naturels) risque d'interdire l'accès aux données récoltées. A contrario, plus les données sont proches d'une base centrale, externe au réseau lui-même et plus les mises à jours sont onéreuses, mais plus l'accès au données se trouve facilité.

En somme, des travaux déjà effectués dans ces différents domaines, il en ressort certaines idées qui peuvent être réutilisées pour répondre aux problématiques posées par les bases de données capteur, et par notre cas d'étude en particulier. Ainsi, en se fondant sur une approche centralisée de stockage de données, il sera possible de reprendre les idées de structure d'indexation combinant plusieurs sous-arbres. Chaque capteur peut se voir associé un sous-arbre particulier afin de limiter les plages de recherches lors des requêtes. De plus, la structure de l'AP-tree propose un lien direct permettant de valoriser les données les plus récentes. De toutes ces idées est né un arbre d'indexation pour des données issues de capteurs fixes, le PoTree.

4. PoTree

Comme sous-entendu précédemment, le PoTree propose de séparer les aspects temporels et spatiaux (Noël, 2004). Il vise à indexer des données spatio-temporelles, en minimisant la taille de la structure, et en valorisant les recherches de points spatiaux et d'intervalles temporels. Il cherche également à faciliter les mises à jours de données concernant des points spatiaux déjà connus (capteurs fixes).

En utilisant d'abord des paramètres spatiaux pour restreindre le nombre d'accès à la structure, le PoTree se distingue des autres index.

4.1Structure

Contrairement aux approches basées sur les R-tree, où la dimension temporelle est considérée comme une dimension spatiale, il est permis dans le cas présent de séparer temporel et spatial. N'ayant pas dans un premier temps de mouvements à gérer, les mises à jours s'effectueront principalement sur un plan temporel.

Les requêtes portant principalement sur des points spatialement connus (ou des groupes de points connus) et sur des intervalles de temps, les facteurs spatiaux peuvent être utilisés pour délimiter les recherches. Celles-ci ne s'effectuent alors plus sur l'ensemble des données, mais sur la partie relative à une station.

Les données issues d'un même capteur viennent s'ajouter de manière incrémentale. Il n'y a pas de mise à jour a posteriori, ni a priori. Les données utilisent le temps transactionnel. Ceci est propice à l'utilisation de techniques de linéarisation des données.

L'approche retenue propose d'utiliser une forme de kd-tree pour la gestion de l'aspect spatial et des dérivés de B+-trees pour la gestion de l'aspect temporel (cf. figure 2). On notera cependant que les entrées se feront toujours à droite de l'arbre, et qu'il est alors possible de remplir les nœuds feuilles au fur et à mesure de leur création.

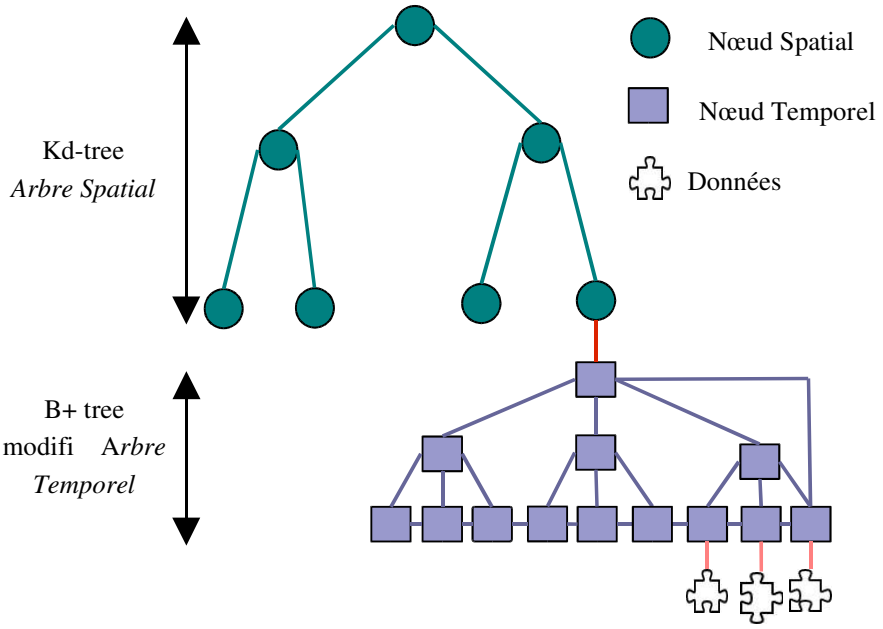


Fig. 2 *Structure du PoTree*

Le choix du kd-tree est motivé par des comparaisons entre cette structure et les R-tree. En résultats de ces comparaisons, il est apparu que les kd-tree, offrent des performances~~une flexibilité~~ intéressantes par rapport au R-tree. Si les temps de recherche peuvent s'avérer comparables, les temps de création d'index et la taille de l'arbre utilisé sont en faveur du kd-tree. La sensibilité de cette structure à l'ordre d'entrée des données est ici limitée par le faible nombre de données spatiales (positions des capteurs) à ajouter, et par le fait qu'elles le soient lors de la création de la structure.

L'utilisation des B+-tree est justifiée par les possibilités offertes en matière de mise à jour et de recherche. De plus, cette structure pourrait être développée par la suite afin de faciliter l'intégration de la base de donnée à un système d'entrepôt de données. Dans notre cas, les ajouts « à droite » de la structure lors des mises à jour auraient pu faire prévaloir un Append-Only-tree, cependant les faibles performances de cet arbre pour des recherches sur des intervalles longs et anciens ~~permettent de~~forcent à l'écarter pour l'instant. En effet, dans certains cas de figure, les spécialistes utilisateurs du système peuvent être intéressés par des données s'étalant sur plus d'un an, ou par des données caractéristiques vieilles de plusieurs années.

L'utilisation d'une structure duale permet d'associer chaque localisation spatiale à un B-tree. De la sorte, chacun de ces arbres représente les données issues d'une source d'informations, d'un objet spatial donné. Ceci permettrait de développer une structure parallèle permettant d'indexer directement ces arbres, en fonction de leurs OIDs (identificateurs d'objets), sans avoir à effectuer une nouvelle requête spatiale.

Comme il apparaît sur la figure 2, les feuilles du kd-tree pointent vers des B-trees. Ce sont les feuilles des B-trees qui pointent vers les données. Les nœuds du kd-tree ne comportent que des données relatives à la position spatiale des objets. Ils ne portent pas d'informations relatives au temps.

Parallèlement, les nœuds des B-trees ne portent pas d'informations relatives à la position géographique des objets, mais uniquement les informations relatives au temps. De ce fait la gestion de la mobilité n'est pas encore assurée.

Implémentation & tests

Différents essais ont été effectués entre le PoTree et le R*-tree. Pour ce dernier, le R*-tree, nous avons repris l'implémentation de (Hadjieleftheriou 2005~~4~~). Des données aléatoires ont été produites et séquentiellement attribuées à un nombre fixe de points spatiaux agissant en tant que sources d'informations. Des séries de tests ont été effectuées en changeant le nombre de données à indexer (de 1000~~-~~ à 200000), le nombre de sources d'informations (10-100) employées et la partie de la base à balayer pour les requêtes d'intervalle. Les tests ont été effectués sur un ordinateur Athlon XP cadencé à 1,3 G Hz~~hertz,~~ avec 256 Mo de RAM, sous Linux. Le langage de programmation utilisé était Java.

Puisque le R*-tree ne différencie pas les dimensions spatiales et temporelles, et comme les requêtes de mises à jour doivent tenir compte de l'ensemble de données, le coût de construction pour un tel arbre peut mener à une situation où il devient impossible de répondre aux contraintes en temps réel. De son coté, le PoTree subdivise cet ensemble de données. Il emploie différents arbres temporels pour classer les données, selon les capteurs dont ils dépendent. La figure 3 montre les temps de construction pour les deux structures. Ces résultats ont été obtenus à partir de données réelles concernant 68 capteurs fixes. Il s'avère que le temps de construction de PoTree est bien inférieur car le nombre de points spatiaux est limité et les sous-arbres temporels doivent seulement accéder à la racine et aux derniers nœuds, excepté lorsque l'arbre B+ se restructure quand un nœud est rempli.

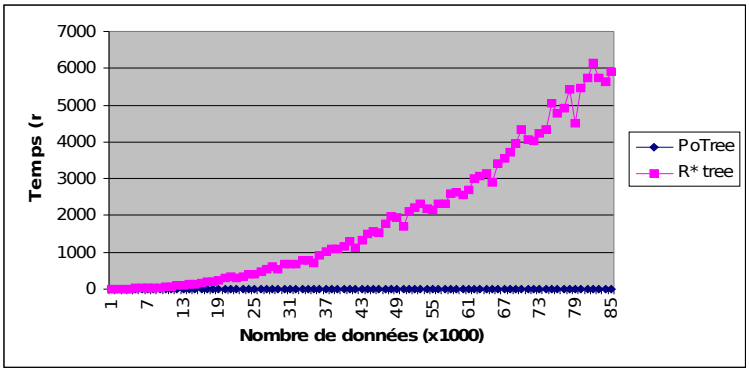


Fig. 3 : *Evolution du temps de construction des structures en fonction du nombre de données à indexer.*

D'autres essais ont montré que la durée de construction du PoTree a augmenté logarithmiquement~~semi-linéairement~~ avec le nombre de stations, le nombre de différentes ~~endroits spatiaux~~ localisations spatiales. Le temps de mise à jour de l'arbre, hors ajout de nouveau capteur, étant alors égal au temps de parcours de l'arbre spatial ajouté au temps de mise à jour du dernier noeud de l'arbre temporel et en cas de débordement de ce noeud de la mise à jour de cette partie de l'arbre (selon les techniques des B+-tree). En utilisant des requêtes de mise à jour, nous avons pu classer des données venant de 1200 sondes simulées (sismographes à 100 hertz), comme montré sur la figure 4.

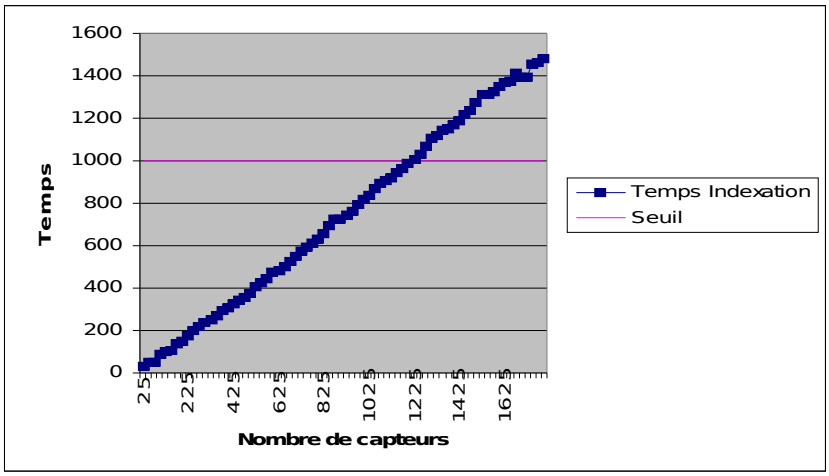


Fig 4 : *Influence du nombre de capteurs fixes sur le temps de création de l'arbre (indexation de une seconde de données à 100 Hz)*

D'autres requêtes ont montré des propriétés intéressantes. Les requêtes d'intervalle ont profité de l'enchaînement des nœuds temporels du PoTree. Pour des requêtes de point spatial / intervalle temporel, le PoTree peut être jusqu'à 8 fois plus rapide que le R*-tree. Pour des requêtes de fenêtre spatiale / intervalle temporel la différence s'est montrée plus légère; elle demeure cependant toujours en faveur de notre solution, comme le montre la figure 5. Sur cette figure, les 10 derniers pourcents des données saisies furent cherchés; la fenêtre spatiale couvrant l'ensemble de l'espace. Pour un faible nombre de données le R*-tree se comporte mieux, pourtant quand la quantité de données dépasse 6000, c'est le PoTree qui se montre le plus rapide~~gagne l'avantage~~. Ces résultats peuvent être expliqués par le fait que les R*-tree doivent considérer l'ensemble des données afin d'exécuter une requête. Par conséquent, plus il existe de données, plus les requêtes sont lentes.

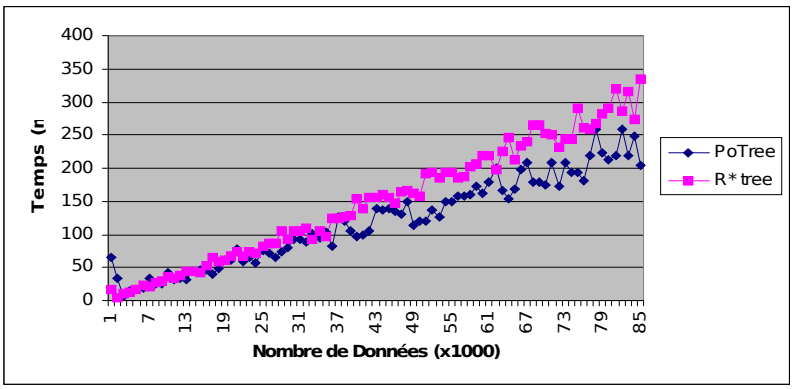


Fig. 5 : *Temps de résolution d'une requête de Fenêtre spatiale / Intervalle temporel en fonction du nombre de données présentes dans la base.*

Un deuxième ensemble de tests a été effectué, en utilisant des données venant des sismographes du K-Net, institut de recherche en science de la terre et pour la prévention de désastre (K-Net, 200~~5~~4). Les données ont été indexées par le PoTree avec des résultats semblables aux autres obtenus avec des données aléatoires.

Les résultats obtenus montrent l'adéquation du PoTree avec les contraintes associées à une base de données liées à un ensemble~~s~~ de capteurs fixes, traitant de grandes quantités de données. Cette structure permet une valorisation des données les plus récentes. Cependant, elle reste limitée de par sa gestion non optimale de la mobilité des capteurs. De plus, les modes d'interrogation de la base sont basés sur des caractéristiques spatiotemporelles, limitant la compatibilité avec les modes

d'utilisation classiques, associés à des identifiants de capteurs. Afin de palier ces manques, une nouvelles structure est présentée : le PasTree.

5. Le PasTree

5.1 Idées de base

Le PasTree se base sur différentes idées (Noël, 2005). En tout premier lieu, cet arbre d'indexation pour base de données spatiotemporelles cherche à conserver la possibilité d'accéder aux données par les identifiants de capteurs tout en ajoutant la possibilité d'y accéder par des particularités spatiotemporelles. Le but étant de proposer plusieurs méthodes d'accès aux données, afin de permettre à différents spécialistes d'utiliser les données récoltées selon les critères exploités dans leur processus métier habituel. De plus, les systèmes basés sur les réseaux de capteurs, et plus particulièrement ceux chargés de la gestion des risques naturels se doivent d'offrir des interfaces et un accès simplifiés aux informations recherchées en phase critique ou lors de la prise de décisions.

Pour cela, le PasTree pourra utiliser plusieurs sous-structures combinées entre elles; la redondance occasionnée se justifiant par les besoins d'accès variés aux données.

Les données provenant d'un certain capteur sont regroupées au sein d'une même sous-structure temporelle, valorisant les données les plus récentes. Cette idée, reprise du PoTree, permet un chaînage rapide des données issues d'une même source. Dès lors, les autres sous-structures auront pour but de déterminer lequel de ces sous-arbres temporels interroger.

La première de ces sous-structures permet de déterminer quel sous-arbre temporel, lié à un capteur, doit être interrogé par le biais d'un identifiant d'objet (le capteur). Il s'agit là d'un arbre B+ classique, indexant les identifiants de capteurs. Il est raisonnable de penser que le schéma d'attribution des noms des capteurs les regroupe d'une manière ou d'une autre, en fonction d'une zone géographique ou d'un type de capteur par exemple. Il est alors envisageable que des requêtes s'effectuant sur les données issues de plusieurs capteurs pourront bénéficier des liens existant entre les nœuds fils de cette sous-structure.

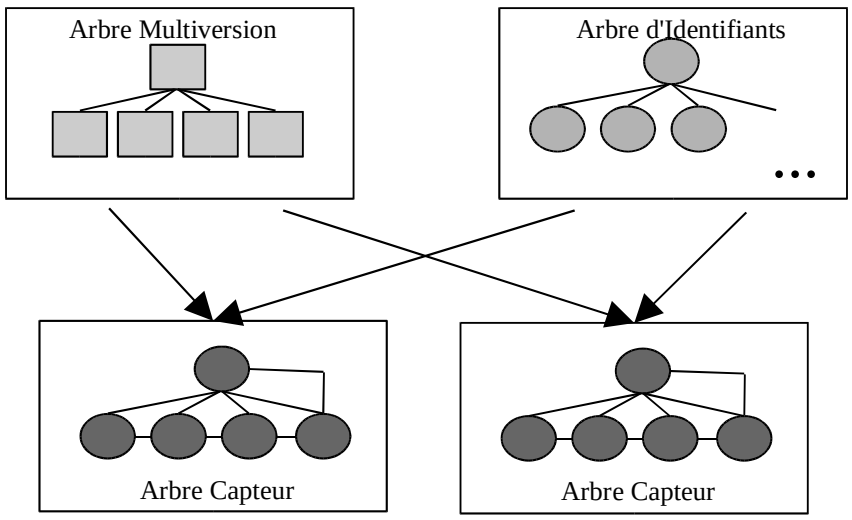
La seconde de ces sous-structures, établie en parallèle de celle référençant les identifiants travaille de son coté sur les données spatiales. En effet, on désire intégrer dès le niveau de l'index des données spatiotemporelles. Via cet index, un quadtree (Ooi, 1997), les utilisateurs pourront émettre des requêtes en fonction des zones spatiales jugées intéressantes, et non plus en fonction des capteurs.

Cependant, l'utilisation d'une structure spatiale seule ne permet pas de prendre en compte tous les cas de figure possibles. En premier lieu, certaines stations peuvent comporter plusieurs capteurs différents. Dans un autre cas de figure, un capteur se déplaçant peut passer plusieurs fois par une même position, pendant des intervalles de temps différents... Une solution pour prendre en compte ce comportement est d'utiliser une approche multiversions (Tzouramanis, 2000).

Dernier point à prendre en considération: les types de données. En effet, il est possible de décomposer les données en deux catégories. Tout d'abord celles issues des mises à jour « fixes ». Dans ce cas, il n'y a pas de changement de position, donc il n'est pas nécessaire de noter une modification spatiale du capteur. En second lieu, on note des mises à jour avec changement de position. Dès lors, il devient nécessaire de conserver la mesure récoltée ainsi que le changement de position. Ceci suppose bien sûr que les capteurs soient capables de déterminer de manière fiable leur position. Une position peut être entrée manuellement à la création du capteur si il ne dispose pas de système de détection de position particulier.

5.2 Description du PasTree

En fonction des critères retenus, il a été décidé d'élaborer le PasTree comme indiqué dans la figure 6. On peut donc remarquer que la structure globale comprend trois types de sous-structures.



5.2.1 Description de la structure

En premier lieu, la sous-structure relative capteurs (figure 7) est reprise sur celle développée pour le PoTree. Il s'agit d'un arbre B+ modifié. On a pu y rajouter un lien direct entre la racine de l'arbre et le dernier nœud feuille. Ainsi, tous les accès à cette sous-structure, en écriture comme en lecture commencent par tester si les données recherchées ne se trouvent pas dans le dernier nœud.

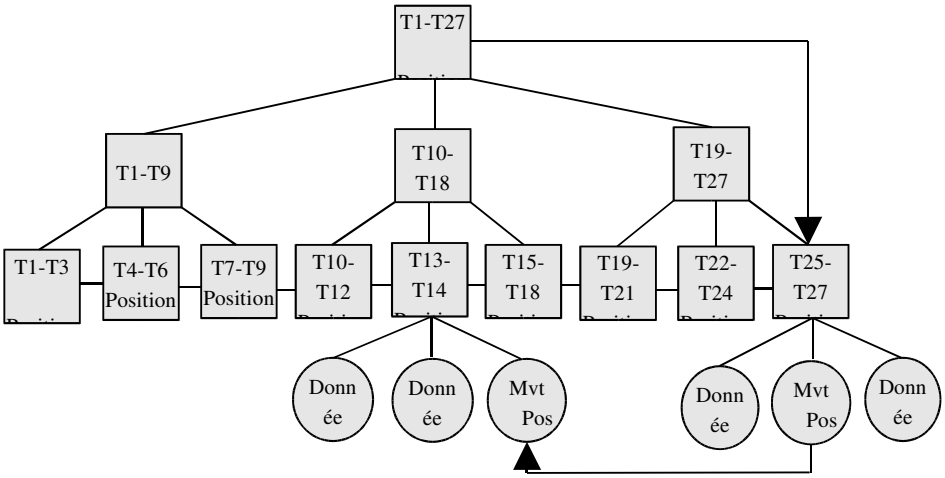


Fig. 7 : *Structure du sous-arbre Capteur*

En écriture, cela facilite les mises à jour, en épargnant au processus de devoir parcourir l'ensemble de l'arbre.

En lecture, puisque les scientifiques autant que les preneurs de décisions sont généralement plus intéressés par les données les plus récentes, cela permet également de retrouver immédiatement les informations voulues. On n'utilise cependant pas un simple enregistrement séquentiel dans la mesure où les systèmes de surveillance de phénomènes naturels font régulièrement appels à des données de période critiques passées pour déterminer le comportement actuel du système. Dès lors, il est préférable de pouvoir utiliser une structure d'arbre B+ pour y accéder.

Ce sous-arbre a été enrichi par rapport à la version utilisée par le PoTree dans la mesure où il est à présent permis de prendre en compte une différence entre des données issues de capteurs fixes et des données issues de capteurs venant de se déplacer.

En d'autres termes, c'est à ce niveau qu'apparaît la différenciation entre les types de données. Les feuilles de l'arbre possèdent donc deux types d'enregistrements. Le premier est de type <Tm; pointeur>, avec Tm le temps de la mesure (on utilise alors le temps envoyé par les capteurs que l'on suppose coordonnés entre eux) et pointeur, un pointeur vers les données. Le second type de données, plus complexe est de type <Tm; ptr_pos; ptr_pos_preced; pointeur>. Dans ce cas, ptr_pos est un pointeur vers la partie position de la donnée, ptr_pos_preced est un pointeur vers la position précédemment occupée par le capteur. De plus, au niveau des nœuds feuille, on conserve un pointeur vers la position courante en fin de nœud. Ce rajout à la structure originelle augmente le coût de mise à jour et diminue le nombre d'entrée dans chaque nœud en fonction du taux de mobilité des capteurs. Cependant, il permet de suivre les évolutions d'un capteur particulier. En ne connaissant que l'identifiant de celui-ci, il est alors possible de le suivre dans une modélisation spatiotemporelle, sans avoir pour cela à interroger la base selon des critères spatiaux.

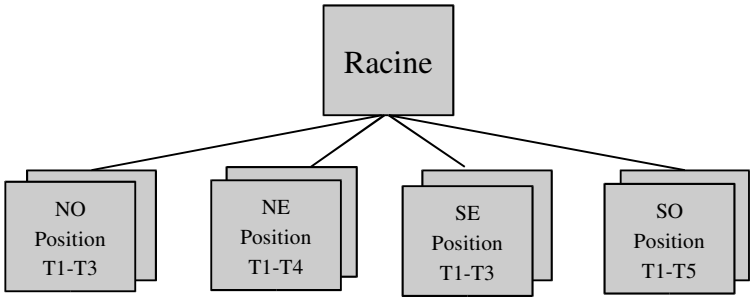


Fig. 8 : S*tructure du sous-arbre spatial, un quadtree multiversions*

L'arbre spatial, basé sur un quadtree multiversions, est de son coté basé sur le quadtree, auquel on rajoute un aspect multiversions (figure 8). Les quadtrees décomposent récursivement l'espace en quatre quadrants égaux (Nord-Ouest, Nord-Est,...) Les entrées du quadtree, à l'intérieur de chaque quadrant, sont ~~donc~~ de la forme <position; Td; Tf> avec *position* la position courante du point dans le quadrant concerné, Td le début de l'intervalle pendant lequel le capteur est considéré dans la zone et Tf la fin de cet intervalle. On notera que cette approche se base sur une notation discrète des positions, des fonctions d'extrapolation pouvant par la suite être utilisées pour estimer la position du capteur entre deux mesures. Pour ce

qui est des positions courantes, on établit Tf à la valeur *NOW* pour marquer que le capteur n'a pas changé d'emplacement. Les travaux de Tao (Tao, 2002) montrent l'influence de l'agilité (le taux de changement de positions des capteurs entre deux mesures) du système sur les performances de l'approche multiversions. Face à l'*overlapping*, ils favorisent cependant cette approche dans le cadre d'une agilité faible avec un taux de charge du système moyen.

Le dernier arbre référence les identificateurs de capteurs dans un arbre B+ classique; les feuilles de cet arbre pointant vers des sous-arbres de capteur. Ce sous-arbre a pour but de permettre la compatibilité avec l'utilisation classique des systèmes de surveillance de phénomènes naturels, via des références de capteurs.

L'utilisation de plusieurs sous arbres entraîne la mise en place de mécanismes particuliers afin de prendre en charge les différents types de mise à jour ou d'interrogation de la base. En d'autres termes, il est nécessaire de développer des algorithmes pour les accès passant par le Quadtree et des algorithmes pour les accès passant par les identificateurs de capteurs.

5.2.2 Accès à la structure

D'une manière globale, le processus d'accès aux données peut se décomposer en deux phases. Dans un premier temps, via l'arbre spatial ou l'arbre des identifiants de capteurs~~SIDs~~, on détermine quel arbre de capteurs parcourir. De là, on exécute une recherche dans ce sous-arbre.

On notera que pour les recherches sur des intervalles temporels, la principale différence avec cet algorithme vient de la présence de tests complémentaires afin de vérifier que les intervalles recherchés et de présence du capteur sont compatibles. On effectue par la suite les recherches d'intervalles dans les sous-arbres Capteur avec des bornes égales à l'intersection des deux intervalles temporels.

5.3 Tests effectués

Différentes séries de tests on été effectuées sur la structure du PasTree. La première série de tests portait principalement sur des données générées aléatoirement tandis que la seconde série reprenait des données *batch* de sismographes du réseau de surveillance japonais K-Net.

Le langage Java (VM Sun 1.4.2) a été utilisé pour coder la structure de l'arbre, fonctionnant sur un PC disposant de 256 Mo de RAM et d'un processeur Athlon XP de 1,36 GHz.

Parmi les facteurs testés, on a pu s'intéresser plus particulièrement à l'influence du nombre total de capteurs ainsi qu'à leur agilité, le taux de changements de positions entre deux prises de mesures. On a également étudié l'influence du nombre de positions différentes utilisées.

Des comparatifs ont été effectués avec d'un coté le PasTree et de l'autre le PoTree et le R*-Tree. Le code source de (Hadjieleftheriou, 2005) a été utilisé pour le R*-Tree.

Plusieurs séries de tests ont été menées. La première utilisait des données générées aléatoirement afin de correspondre à différents cas d'utilisation. On y a fait varié le nombre de capteurs, leurs positions et leur agilité. La seconde série de tests utilisait des données issues des sismographes du K-Net (K-Net, 2005), correspondant à différents séismes enregistrés par ces capteurs. Dans ce cas particulier, l'agilité des sismographes était nulle; ils ne bougeaient pas. Les résultats obtenus avec les données réelles ont confirmé ceux issues de données aléatoires.

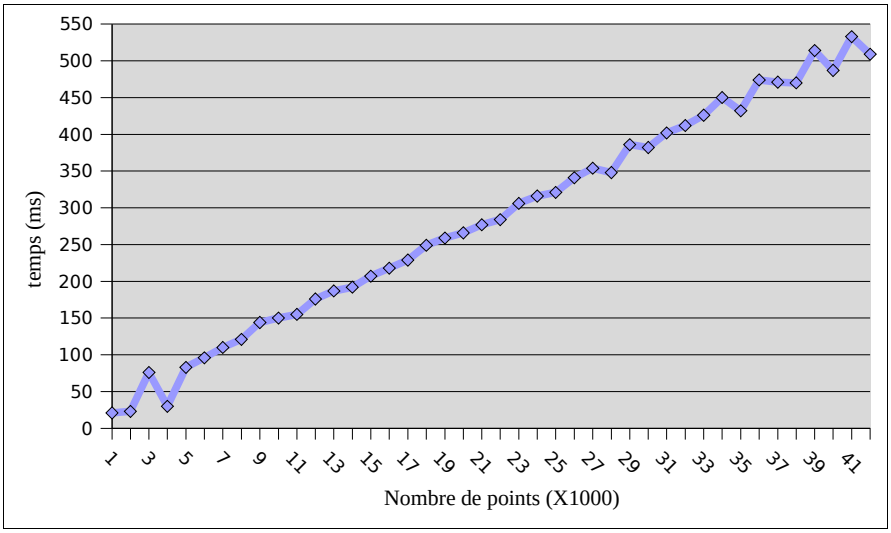


Fig. 9 : *Temps de construction du PasTree en fonction du nombre de données à indexer pour 68 stations fixes.*

La figure 9 nous montre l'influence du nombre de points à indexer sur le temps de construction de la structure. Le test a été effectué avec les données

sismographiques réelles provenant de 68 capteurs fixes. On peut remarquer une augmentation semi-linéaire du temps de construction. Dans cet exemple, il n'y a pas de changement de position des capteurs. Par conséquent les capteurs sont placés dans le sous-arbre spatial basé sur un quadtree. Par la suite, lorsque le quadtree est créé, l'ajout de données à indexer ne fait que traverser cette sous-structure pour vérifier qu'il n'y a pas eu de changement de position. Au niveau des sous-structures de capteurs, la première opération effectuée consiste à vérifier si la mise à jour concerne les données les plus récentes. C'est ici toujours le cas. On obtient donc un temps de traitement linéaire correspondant au test avec les dernières données de l'arbre et à la mise à jour de la dernière feuille. Il n'y a que lorsque la feuille est remplie que la mise à jour doit procéder à une étape de partage de noeud et de réagencement de la sous-structure.

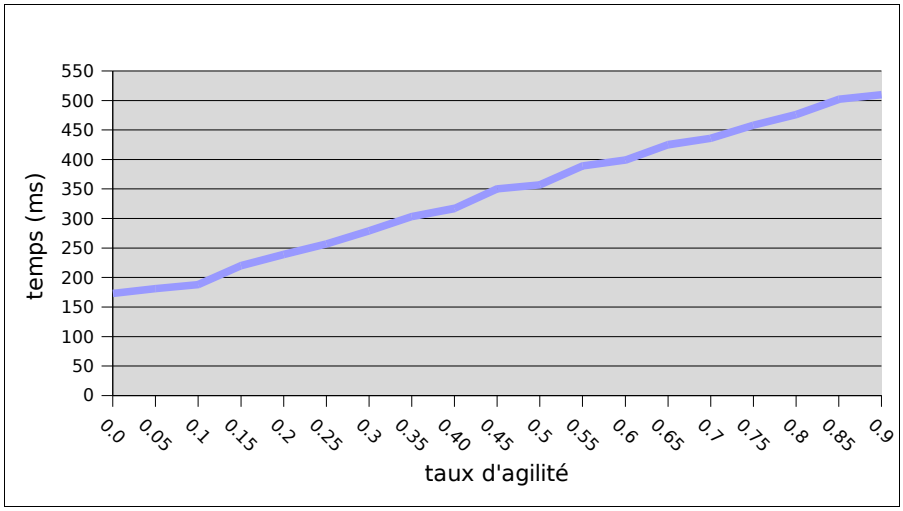


Fig. 10 : *Influence du taux d'agilité sur le temps de construction du PasTree, avec un nombre de capteurs et de données fixes.*

Des tests complémentaires ont été effectués pour connaître l'influence des changements de positions sur le temps de construction (figure 10). Ces tests ont été éffectués sur des données générées aléatoirement. On a utilisé 100 capteurs et 10000 données en tout, soit l'équivalent de une seconde de mesure pour un réseau de capteurs à 100 Hz. On y voit que plus l'agilité des capteurs augmente, plus le temps de construction de l'index augmente également. L'agilité s'évalue sur une échelle de 0 à 1. Ce facteur est donc à prendre en compte pour déterminer le nombre maximal

de capteurs indexables dans le système. On a ainsi pu constater que le PasTree peut indexer jusqu'à 600 capteurs immobiles et 300 capteurs avec une mobilité supérieure à 50%. Par comparaison, le PoTree pouvait gérer dans des conditions analogues 1200 capteurs immobiles. Cependant cette structure ne permettait pas la gestion des changements de position des capteurs.

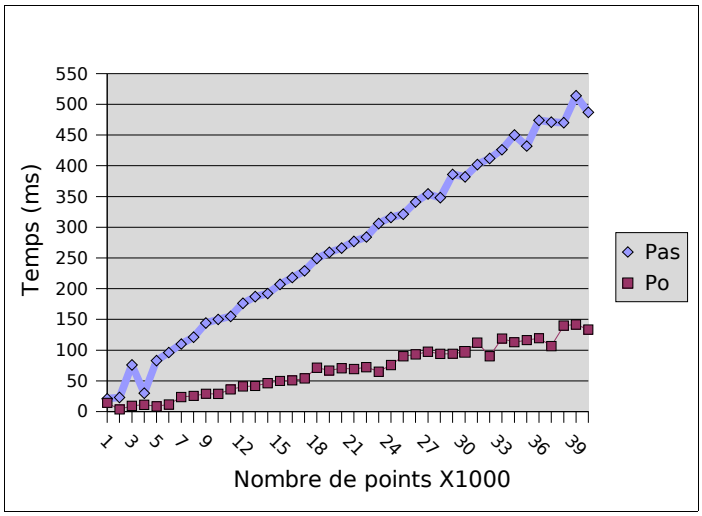


Fig. 11 : *Temps de construction PoTree / PasTree en fonction du nombre de points à indexer pour 68 capteurs fixes*

En continuant la comparaison entre le PoTree et le PasTree, on en arrive au temps de construction. Comme l'indique la figure 11, le PoTree est 3 à 4 fois plus rapide à construire que le PasTree. Cependant il est à noter que les fonctionnalités du PoTree sont très limitées par rapport à celles du PasTree. Le PasTree prend en charge la gestion d'un certain niveau de mobilité des capteurs, les requêtes spatiotemporelles ou basées sur les identifiants de capteurs, etc.~~...~~ La structure du PasTree offre plus de variété au niveau de son mode d'utilisation. Le coût s'en fait ressentir au niveau de la création et de la mise à jour de la structure. Cependant le temps de construction du PasTree reste largement inférieur à celui du R*Tree. En effet, le PasTree utilise la notion de source d'information, réunissant les données issues d'un même capteur dans une sous-structure commune. Contrairement au R*Tree, les données ne sont pas ici considérées comme indépendantes les unes des autres mais liées par leurs sources. Il ne devient donc plus nécessaire de devoir

placer les données dans un espace à 4 dimensions (3 spatiales et 1 temporelle). Il suffit de placer d'un coté les données spatiales, puis de l'autre les données temporelles dans des sous-structures différentes, sans avoir à reconstruire l'arbre global.

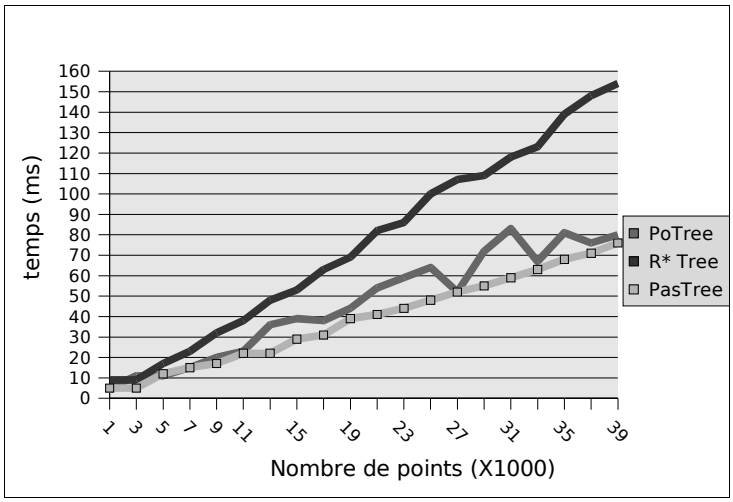


Fig. 12 : *Temps de résolution d'une requête de recherche sur une fenêtre spatiale / intervalle temporel en fonction du nombre de données contenues dans la base.*

La figure 12 montre les temps de recherche pour une requête de type fenêtre / intervalle. On a pris ici des données réelles issues de 68 capteurs fixes. La requête récupère les derniers 10% des données entrées sur l'ensemble du domaine spatial. On notera que le PoTree et le PasTree offrent des performances assez proches et plus intéressantes que le R*Tree. Une fois de plus ceci peut s'expliquer par l'utilisation qui est faite des sous-structures de capteurs, réunissant les données issues d'une même source au sein d'un même arbre. Alors qu'un R*Tree doit prendre en compte l'ensemble des données pour répondre à une requête, le PasTree se contente de sélectionner les sous-arbres correspondant et d'interroger ceux-ci. De plus, puisque les données recherchées étaient les plus récentes, le PasTree a pu tirer profit du lien direct entre la racine et la dernière feuille de ses sous-arbres, économisant ainsi du temps.

En somme, le PasTree offre une solution d'indexation de données permettant la gestion de critères spatiotemporaux. Il permet également une compatibilité avec les modes d'utilisation de la base basés sur des identifiants de capteurs. Le PasTree gère

un certain niveau d'agilité des capteurs. Tous ces apports au PoTree ont cependant un coût. Bien que cette structure soit encore plus rapide que le R*-tree, le PasTree est dans l'ensemble plus lent que le PoTree, du fait de sa structure plus imposante, basée sur une certaines redondance d'informations.

6. Travaux futurs

Alors que les deux solutions proposées répondent à un certain nombre d'attentes, différents axes d'étude se sont dévoilés. En premier lieu, l'utilisation d'une base centralisée pose les problèmes de la réplication des données, mais surtout de la saturation de la base. De par la quantité de données arrivant pour être indexées, la saturation mémoire de la base est à prendre en compte. Les prochains travaux porteront donc sur l'étude de systèmes prenant en compte cet élément.

De plus, d'autres développements pourront avoir lieu. La tendance actuelle étant aux systèmes distribués, l'étude d'un système de base de données capteur distribué pourrait à son tour être prometteuse.

7. Conclusion

Nos travaux proposent deux solutions d'indexation pour des bases de données capteurs, en mémoire vive, soumises à des contraintes spatiotemporelles temps-réel. Les système de surveillance basés sur des réseaux de capteurs envoient traditionnellement leurs données vers une base centralisée. Les données sont alors marquées spatialement par la position du capteur émetteur et temporellement par le marqueur temporel de la mesure. De plus, du fait du~~de par le~~ grand nombre de données à traiter on voit apparaître des contraintes temps-réel. Ces caractéristiques sont communes à de nombreux domaines, en particulier à celui de la surveillance volcanique, la gestion de risques naturels.

La première solution présentée est un index permettant de traiter les données de capteurs fixes indexés spatialement. Le PoTree différencie les dimensions spatiales et temporelles. Il utilise un premier sous-arbre spatial pointant vers des sous-arbres temporels relatifs aux différents capteurs. Ces derniers sous-arbre offrant un lien direct entre la racine et le dernier noeud afin de valoriser les données les plus récentes. Alors que cette structure permet d'obtenir des temps de mise à jour performants, elle montre ses limites de par la difficulté à prendre en charge le

changement de position de capteurs et ~~de~~ par le manque de souplesse dans le mode d'interrogation de la base.

Une seconde structure, dérivée du PoTree, le PasTree comble ces manques en modifiant l'arbre spatial et en lui ajoutant un aspect multiversions. En différenciant les données de mise à jour de mesure et les données de changement de position de capteur, le PasTree permet de prendre en charge une certaine agilité des capteurs. De plus, en incorporant une nouvelle sous-structure permettant l'accès via l'identifiant des capteurs, le PasTree élargit les possibilités d'accès aux données offertes, au prix d'un coût de mise à jour légèrement supérieur.

Des travaux à venir s'intéresseront au phénomène de saturation de ces bases de données capteurs en mémoire vive, et pourra également étudier le contexte des technologies distribuées afin d'élargir le champs d'application.

Bibliographie

Bentley J.L., Multidimensional binary search trees in database application, 1975, IEEE Transaction on software engineering, vol 5, n°4, pp 333-340.

Berchtold S., Keim D.A., Kriegel H-P, 1996, The X-tree: an Index Structure for High Dimensional Data, *in proceedings of the 22nd International Conference on Very Large Database*, pp 28-39

Bertino E., Ooi B.C., Sacks-Davis R., 1997, Indexing techniques for advanced database systems, Kluwer Academic Press, London, ISBN 0-7923-9985-4

CENAPRED, 2005, Centro Nacional de Prevencion de Desastres [En ligne], http://tornado.cenapred.unam.mx/mvolcan.html , vu le 01/02/05

Eiman E., Research Directions in Sensor Data Streams: Solutions and Challenges, 2003, technical report DCIS-TR-527, Rutgers University

Hadjieleftheriou M. Spatial Index Library [En ligne], viewable at: http://www.cs.ucr.edu/~marioh/spatialindex/, vu le 01/02/05

Haritsa J.R., Seshadri S., Real-Time Index Concurrency Control, 2000, IEEE Transactions on Knowledge and Data Engineering, vol 13, n°3, pp 429-447

Intanagonwiwat C., Estrin D., Govindam R. et al., 2001, Impact of Network Density on Data Aggregation in Wireless Sensor Networks, *in proceedings of the International Conference on Distributed Computing Systems*, Vienne~~,~~

K-NET, 20054, Kyoshin Network [En ligne], http://www.k-net.bosai.go.jp/k-net/index_en.shtml, vu le 01/12/0401/06/05

Lam K.Y., Kuo T.W., 2001, Real-Time Database Systems: Architecture and Techniques, Kluwer Academic publishers, London, ISBN 0-7923-7218-2

Mokbel M., Ghanem T.M., Aref W.G., 2003, Spatio-temporal Access Methods, IEEE Data Engineering Bulletin, Vol 26, n°2, pp. 40-49

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

Noël G., Servigne S., Laurini R.,20Bentley J.L., 1975, Multidimensional binary search trees in database application, *IEEE Transaction on software engineering*, 5 (4), 333-340.04, The Po-tree, a Real-Time Spatiotemporal Data Indexing Structure, *in proceedings of development in Spatial Data Handling SDH2004*, UK 2004, pp259-270

Noël G, Servigne S., Laurini R., 2005, Spatial and Temporal Information Structuring for Natural Risk Monitoring, *in proceedings of GISPlanet 2005*, Lisbonne

Ooi B.C., Tan K.L., 1997, spatial databases. In Indexing Techniques for Advanced Database Systems, Kluwer Academic Publishers, Boston, ISBN 0-7923-9985-4, 39-75

Satnam A., Agogino A.M., Morjaria M., 2001, A Methodology for Intelligent Sensor Measurement, Validation, Fusion, and Fault Detection for Equipment Monitoring and Diagnostics, Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing, , vol 15, n°4, pp307-320

Song Z., Roussopoulos N., 2003, SEB-tree: An Approach to Index Continuously Moving Objects, *in Mobile Data Management*, pp. 340-344

Tao Y., Papadias D., Zhang J., 2002, Cost Models for Overlapping and Multi-Version Structures, *in proceedings of 18th International Conference on Data Engineering (ICDE'02)*, Califronie, pp 191-200

Theodoridis Y., Vazirgiannis M., Sellis T.K., 1996, Spatio-temporal Indexing for Large Multimedia Applications, *in proceedings of IEEE ICMCS*, pp 441-448

Tzouramanis T., Vassilakopoulos M., and ManolopoulosY..2000, Multiversion linear quadtree for spatio-temporal data. *In proceedingsof ADBIS*, pp. 279-292

USGS, 20054, U.S. Geological Survey Volcano Hazards Program [En ligne], http://volcanoes.usgs.gov/, vu le 01/06/0501/12/04

Vassilakopoulos M., Manolopoulos Y., 1999, External Balanced Regular (x-BR) Trees: new Structures for for Very Large Databases, *in proceedings of 7th Panhellenic Conference on Informatics*, Grèce, pp III.61-III.68

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

Xia Y., Prabhakar S., 2003, Q+Rtree: Efficient Indexing for Moving Object Database, *in proceedings of 8th Conference on Database Systems for Advanced Applications DASFAA 2003*, Kyoto