



Conception d'une base de données capteurs à l'aide de contraintes spatio-temporelles

Manel Charfi, Yann Gripay, Jean-Marc Petit

► To cite this version:

Manel Charfi, Yann Gripay, Jean-Marc Petit. Conception d'une base de données capteurs à l'aide de contraintes spatio-temporelles. BDA 2016, Nov 2016, Poitiers, France. hal-01520716

HAL Id: hal-01520716

<https://hal.science/hal-01520716>

Submitted on 10 May 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Conception d'une base de données capteurs à l'aide de contraintes spatio-temporelles

Manel Charfi
Université de Lyon, CNRS
INSA-LYON, LIRIS, UMR5205
F-69621, Villeurbanne, France
manel.charfi@insa-lyon.fr

Yann Gripay
Université de Lyon, CNRS
INSA-LYON, LIRIS, UMR5205
F-69621, Villeurbanne, France
yann.gripay@insa-lyon.fr

Jean-Marc Petit
Université de Lyon, CNRS
INSA-LYON, LIRIS, UMR5205
F-69621, Villeurbanne, France
jean-marc.petit@insa-lyon.fr

1. CONTEXTE & PROBLÉMATIQUE

De nos jours on a de plus en plus de capteurs qui ont tendance à apporter confort et facilité dans notre vie quotidienne. Ces capteurs sont faciles à déployer et à intégrer dans une variété d'applications (monitoring de bâtiments intelligents, aide à la personne, ...). Ces milliers (voire millions) de capteurs sont de plus en plus envahissants et génèrent sans arrêt des masses énormes de données qu'on doit stocker et gérer pour le bon fonctionnement des applications qui en dépendent. A chaque fois qu'un capteur génère une donnée, deux dimensions sont d'un intérêt particulier : la dimension temporelle et la dimension spatiale. Ces deux dimensions permettent d'identifier l'instant de réception et la source émettrice de chaque donnée. Chaque dimension peut se voir associée à une hiérarchie de granularités qui peut varier selon le contexte d'application.

Dans notre travail on se focalise sur les capteurs dans le cadre des bâtiments intelligents et les applications qui requièrent le stockage à long terme des flux de données issues de ces capteurs. Notre objectif est de mettre en œuvre une approche qui vise à contrôler la stockage des données capteurs en ne gardant que les données jugées pertinentes selon la spécification des granularités spatio-temporelles représentatives des besoins applicatifs. Nous empruntons l'approche déclarative développée dans les années soixante-dix pour la conception de bases de données et on l'étend au flux de données des capteurs. Nous pensons que des contraintes temporelles augmentées avec la dimension spatiale sont nécessaires dans notre contexte pour saisir les contraintes sous-jacentes.

Ayant un ensemble de capteurs émettant des flux de données, notre problématique est :
Comment construire une base de données capteurs pour des applications qui requièrent le stockage à long terme ?

2. PROPOSITION

Nous nous basons sur l'intuition que les granularités spatio-temporelles combinées aux Dépendances Fonctionnelles (DFs)

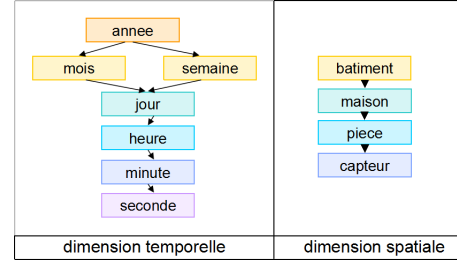


Figure 1: Dimensions spatiale et temporelle

[3] classiques peuvent jouer un rôle majeur pour exprimer quelles sont les données pertinentes à conserver pour une utilisation à long terme. De nombreuses extensions des DFs aux bases de données temporelles ont été proposées [4, 2, 5, 6]. Mais aucun d'entre elles n'étend complètement les DFs aux dimensions temporelles et spatiales. Dans [6], la notion de *Roll-Up Dependencies* (RUD) généralise les DFs temporelles à d'autres dimensions. Les aspects algorithmiques, comme la fermeture des attributs par rapport à un ensemble de contraintes, ne sont pas étudiés pour RUD, comme nous le faisons dans notre travail. A notre connaissance, ce problème n'a pas été étudié en profondeur dans la littérature.

On introduit alors les Dépendances Fonctionnelles Spatio-Temporelles (DFSTs) qui étendent les DFs aux dimensions spatiale et temporelle. Les DFSTs servent à exprimer les relations entre les données en prenant en compte des granularités spatio-temporelles. Les DFSTs sont inspirées des Dépendances Fonctionnelles Temporelles (DFT) [1].

Par exemple, considérons les dimensions spatio-temporelles présentées dans la figure 1 et le schéma de flux de capteurs : *capteursTemperature(temperature, localisation, temps)*.

On peut exprimer le fait qu'un utilisateur considère que « la température de chaque pièce reste la même sur chaque heure » en utilisant la DFST :

$localisation^{piece}, temps^{heure} \rightarrow temperature.$

En fait, comme les résultats requis concernent les données sur une longue période de temps (par exemple quelques mois, plusieurs années), les flux de capteurs peuvent être approximatés grâce aux DFSTs pour réduire l'espace de stockage et augmenter l'efficacité des requêtes applicatives.

Nous proposons une technique de normalisation basée sur deux étapes principales : d'abord, le calcul d'une couverture minimale de DFSTs, ensuite, la production de schémas spatio-temporels pour la base de données capteurs. À l'issue

de la phase de normalisation, on obtient un premier schéma de base de données qu'on appellera dans la suite *schéma abstrait*. A ce stade, la question qui se pose est : quelle valeur choisir (quelle minute de l'heure et quel capteur parmi ceux de la pièce) ?

Afin de choisir la valeur représentative de chaque couple de granules spatio-temporelle, on a défini la notion d'Affectation Sémantique de Valeur (ASV), inspirée des hypothèses sémantiques [1] dans les bases de données temporelles. En effet, une ASV est une sorte d'annotation de schéma qui permet de préciser avec une fonction d'agrégation pour chaque attribut la valeur à garder. Par exemple, on peut préciser que pour l'attribut *temperature* si on doit garder une seule valeur par pièce par heure, il faut choisir la première (ou la moyenne, le maximum ...). Cela veut dire que pour une pièce i à l'heure j on gardera la première valeur de température reçue à une minute appartenant à l'heure j par un capteur qui appartient à la pièce i . Les ASVs dépendent du contexte d'application. Elles peuvent présenter des fonctions d'agrégation plus complexe (par exemple la moyenne des 3 premières valeurs appartenant à un domaine de validité). Ces fonctions peuvent être définies afin d'éviter les problèmes de données incomplètes et bruitées. L'ajout des ASVs au schéma abstrait mène à l'obtention du *schéma concret* qui peut être intégré dans des SGBDR classiques.

Dans notre travail, nous nous intéressons aussi au chargement des données à la volée à partir des flux de capteurs. Pour ce faire, on réutilise l'information sémantique relative au choix de la donnée dans chaque intervalle spatio-temporel contenue dans les ASVs. Notre système crée alors pour chaque ASV un *data wrapper*. Chaque *data wrapper* observe le flux de données du capteur concernant son attribut. Ensuite, il identifie les valeurs pertinentes en fonction de ses granularités et de sa fonction d'agrégation.

3. PROTOTYPE & EXPÉRIMENTATIONS

Nous avons implémenté un prototype qui gère les deux niveaux de prise en compte des flux de données des capteurs : la conception du schéma de la base de données et le chargement à la volée des données à partir des flux des capteurs. Ce qui nous a permis de mener des expériences avec des flux de données, synthétiques et réelles, provenant de bâtiments intelligents. Nous avons comparé notre solution avec la solution de référence et nous avons obtenu des résultats prometteurs en termes de performance de requêtes et d'utilisation de mémoire.

Figure 2 contient une capture d'écran de la phase de création d'une DFST par l'utilisateur. Comme nous pouvons le voir, les DFSTs peuvent être naturelles et simples à définir. En effet, l'utilisateur n'a qu'à cocher les attributs concernés et sélectionner les granularités sans se soucier de la syntaxe.

Parmi les expérimentations que nous avons déroulées afin de vérifier notre approche, nous avons évalué le taux d'erreur et vérifié si l'approximation dégrade les résultats de l'évaluation des moyennes de température. Dans cette expérimentation nous avons utilisé des données réelles issues de capteurs installés dans notre campus. Nous nous sommes intéressés à la différence entre les résultats sur les données brutes (base de données contenant toutes les valeurs reçues par le flux de capteurs) et une base de données capteurs (base de données créée et maintenue en suivant notre approche). Les résultats de la comparaison des moyennes obtenues à partir d'une requête Q_1 sur les données brutes et des moyennes obtenues

Figure 2: Création d'une DFST

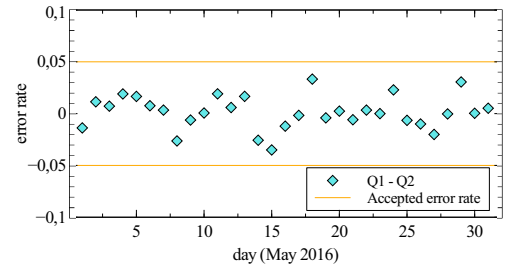


Figure 3: Différence entre les résultats de Q_1 et Q_2

à partir d'une requête Q_2 sur les données pertinentes sont donnés dans Figure 3. On peut voir que la différence est maintenue sous un taux d'erreur de 5%, ce qui peut être considéré comme acceptable dans un contexte réel.

4. REMERCIEMENTS

Ce travail a été financé par une allocation doctorale de recherche de la Région Rhône-Alpes.

5. REFERENCES

- [1] C. Bettini, S. Jajodia, and S. Wang. *Time granularities in databases, data mining, and temporal reasoning*. Springer, 2000.
- [2] C. S. Jensen, R. T. Snodgrass, and M. D. Soo. Extending existing dependency theory to temporal databases. *Knowledge and Data Engineering, IEEE Transactions on*, 8(4) :563–582, 1996.
- [3] M. Levene and G. Loizou. *A guided tour of relational databases and beyond*. Springer Science & Business Media, 2012.
- [4] V. Vianu. Dynamic functional dependencies and database aging. *Journal of the ACM (JACM)*, 34(1) :28–59, 1987.
- [5] X. S. Wang, C. Bettini, A. Brodsky, and S. Jajodia. Logical design for temporal databases with multiple granularities. *ACM Transactions on Database Systems (TODS)*, 22(2) :115–170, 1997.
- [6] J. Wijnen. Temporal fds on complex objects. *ACM Transactions on Database Systems (TODS)*, 24(1) :127–176, 1999.