Spatio-Temporal Functional Dependencies for Sensor Data Streams

Manel Charfi(✉), Yann Gripay, and Jean-Marc Petit

Université de Lyon, CNRS, INSA-LYON, LIRIS,
UMR5205, 69621 Villeurbanne, France
{manel.charfi,yann.gripay,jean-marc.petit}@insa-lyon.fr

Abstract. Nowadays, sensors are cheap, easy to deploy and immediate to integrate into applications. Since huge amounts of sensor data can be generated, selecting only relevant data to be saved for further usage, e.g. long-term query facilities, is still an issue. In this paper, we adapt the declarative approach developed in the seventies for database design and we apply it to sensor data streams. Given *sensor data streams*, the key idea is to consider both spatio-temporal dimensions and Spatio-Temporal Functional Dependencies as first class-citizens for designing sensor databases on top of any relational database management system. We propose an axiomatisation of these dependencies and the associated attribute closure algorithm, leading to a new normalization algorithm.

1 Introduction

Thousands and even millions of sensors can be deployed easily, generating data streams that produce cumulatively huge volumes of data. Whenever a sensor produces a data (temperature, humidity...), two dimensions are of particular interest: the *temporal* dimension to stamp the value produced at a particular time and the *spatial* dimension to identify the location of the sensor. Both dimensions have different granularities organized into hierarchies that may vary according to special needs of applications (see an example in Fig. 1).

Example 1. We consider a running example of sensor data streams from intelligent buildings. In each building different sensors (temperature, luminosity, humidity...) are deployed. At the scale of several buildings, a huge number of sensors exist, each one sending values at its own rate. Let us consider a temperature sensor data stream as given in Table 1, where *location* and *time* follow the granularity hierarchies presented in Fig. 1. Without loss of generality, we shall assume throughout the paper that the `location` attribute contains a string value containing the concatenation of all its "granules".

The stream given in Table 1 is representative of a wide variety of sensor data streams. We consider static sensors in this paper, i.e. sensors set to a particular place to sense the environment. We focus on applications requiring long term storage of data streams, in opposition to monitoring applications which have to react

M. Gertz et al. (Eds.): SSTD 2017, LNCS 10411, pp. 182–199, 2017.
DOI: 10.1007/978-3-319-64367-0_10

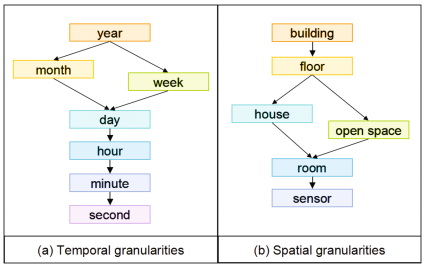

Fig. 1. Temporal and spatial dimensions

Table 1. An instance of *temperatureSensors*

Temperature	Location	Time
21	oxygen:f1:h1:livingRoom:s11	2016/03/02 11:59:00
20	oxygen:f1:h1:kitchen:s21	2016/03/02 11:59:30
20	oxygen:f1:h1:livingRoom:s12	2016/03/02 12:01:00
23	oxygen:f1:h1:kitchen:s22	2016/03/02 12:01:00
20	oxygen:f1:h1:livingRoom:s11	2016/03/02 12:01:30
24	oxygen:f1:h1:bathroom:s31	2016/03/02 12:02:00
20	oxygen:f1:h1:livingRoom:s12	2016/03/02 12:02:30
23	oxygen:f1:h1:kitchen:s21	2016/03/02 12:15:00

in quasi real-time using for example continuous query [1,2]. Many types of systems can be used for long term storage, from classical file systems (in json, flat text...) to Relational Database Management Systems (RDBMS). In the former, a file is created for a given period of time (e.g. week, month). In the latter, data streams are stored as classical tables with specific attributes to cope with spatio-temporal aspects of the stream. In both cases, the burden is let to application designers who have to pose complex queries (e.g. full-text search or SQL-like queries) to deal with temporal and spatial dimensions. Whenever the data volume is high, the query processing time can be also prohibitive. Selecting only relevant data from sensor data streams to be saved for further usage, e.g. long-term query facilities, is still an issue since huge amounts of sensors can be deployed for a specific application. The challenges of applying traditional database management systems to extract business value from such data are still there. In this setting, the problem we are interested in is the following: *Given a set of sensor data streams, how to build a relevant sensor database for long-term reporting applications?*

In our work, we focus on the implementation of a declarative approach that aims to guarantee the storage of the "relevant" sensor data at application-specific granularities. Instead of relaying on query workload or data storage budget to define an optimized database [20], we borrow the declarative approach developed in the seventies for database design using functional dependencies as constraints.

Our aim is to apply this approach in order to transform real-time sensor data streams into a sensor database. We argue that such constraints augmented with the spatial and the temporal dimensions are required to keep only the "relevant data" from incoming sensor streams. The sensor database satisfies the specified constraints while approximating data stream values. Thus, the data approximation and data reduction are "controlled" by the set of constraints. As the required results concern data over long periods of times (e.g. couple of months to several years), approximating sensor data streams should decrease storage space while allowing to express "relevant queries" more easily. We introduce Spatio-Temporal Functional Dependencies (STFDs) which extend Functional Dependencies (FDs) with the temporal and the spatial dimensions.

Example 2. Let us take back our running example. The building manager considers that the "temperature of each room remains the same over each hour", leading to the following STFD: $location^{room}, time^{hour} \rightarrow temperature$ whose syntax captures the intended meaning, i.e. it does not exist two different temperature values on the same room during a given hour.

Such constraints are straightforward to understand and convey the semantics allowing to decide what are the relevant data to keep in the database. Classical database normalization techniques can be revisited in this setting to obtain so-called *granularity-aware sensor databases*.

At given spatio-temporal granules (e.g. for a given hour and a given house), we have different alternatives to choose data, each one could be seen as an *aggregation* of values, from simple ones like *first*, *minimum*, *average within each spatio-temporal granule* to more elaborated ones. Thus, depending on the application context, we can imagine more complex aggregations that allow to define complex functions (e.g. average of the 3 first values corresponding to a given valid domain) and even to avoid noisy and incomplete data issues. We call these annotations Semantic Value Assumptions (SVA) which are based on *semantic assumptions* of temporal databases [3]. So, if we consider temperature values of each room at the scale of an hour, we have to precise which value will be associated to each couple (room,hour): it may be the average of all values from all sensors in this room at all the minutes of this hour. Once a sensor database has been built, we have a data exchange problem [7]: how to load data from the stream to the database? To do that, the main problem is to decide which values to pick up from the stream. This decision is easily made thanks to SVAs.

Contribution

We aim to establish a sensor data storage system that eases the storage of spatio-temporal data streams. To the best of our knowledge, this is the first contribution talking advantage of spatio-temporal constraints defined at database design time to automatically produce an approximated database saving "relevant" data with respect to users' constraints. Our main objective is at the end to have a reduced database that contains a summary of sensor data streams in accordance to application-predefined requirements. Thus, given a set of sensor

data streams, we propose a declarative approach to build a representative sensor database on top of any RDBMS with the following key features:

- Both spatio-temporal granularity hierarchies and STFD are considered as first class-citizens.
- A specific axiomatisation of STFD and an associated attribute closure algorithm, leading to an efficient normalization algorithm are introduced.
- A middleware to load on-the-fly relevant data from sensor streams into the granularity-aware sensor database is proposed.

We have implemented a prototype to deal with both database design and data loading. We have conducted experiments with synthetic and real-life sensor data streams coming from Intelligent Building.

Paper organization
Section 2 gathers preliminaries. In Sect. 3, we define the formalism of STFDs leading to the design of the granularity-aware sensor database. In Sect. 4, we sketch the architecture of our declarative system and the experiments conducted on intelligent building data streams. The related work is presented in Sect. 5. Finally, in Sect. 6 we conclude and expose some perspectives.

2 Preliminaries

To define spatio-temporal granularity hierarchies, we first define a general notion of granularity, borrowed from time granularity definition [3], then a partial order on a set of granularities for which we require a lattice structure.

Let T be a countably infinite set and $\leq$ a total order on T. A granularity is a mapping G from $\mathbb{N}$ to $\mathcal{P}(T)$ where $\mathcal{P}(T)$ is the powerset of T. A non-empty subset $G(i)$, $i \in \mathbb{N}$, of a granularity G is called granule. The granules in a granularity do not overlap. A granularity G is *finer than* a granularity H (H is coarser than G), denoted $G \preceq H$, if for each integer i, there exists an integer j such that $G(i) \subseteq H(j)$. Intuitively this means that each granule of H holds a set of granules of G. Let $\mathcal{G}$ be a set of granularities and $\preceq$ a partial order on $\mathcal{G}$. We assume the set $(\mathcal{G}, \preceq)$ is a lattice, meaning that each two-element subset $\{G_1, G_2\} \subseteq \mathcal{G}$ has a join (i.e. least upper bound) and a meet (i.e. greatest lower bound). For $X \subseteq \mathcal{G}$, $glb(X)$ denotes the greatest lower bound of X in $\mathcal{G}$. The greatest and least elements of $\mathcal{G}$, or the top and bottom elements, are denoted by *Top* and *Bottom*, respectively. G is *collectively finer* than $\{G_1, \ldots, G_m\}$, denoted by $G \preceq_c \{G_1, \ldots, G_m\}$, if for each positive integer i, there exist k, j such that $G(i) \subseteq G_k(j), 1 \leq k \leq m$ and j a positive integer. Intuitively this means that there exists in the set of granularities $\{G_1, \ldots, G_m\}$ at least one granularity that, for each granule $G(i)$ taken independently, is coarser than G. As a particular case, $G \preceq G'$ implies $G \preceq_c \{G'\}$.

Application to spatio-temporal dimensions
For each considered dimension, a lattice of granularities is defined accordingly. For the sake of clearness, we assume without loss of generality that a total

order exists for time instants and sensors, meaning that two time instants (resp. two sensors) can always be compared. In this setting, we define two lattices, $(\mathcal{T}, \preceq_t)$ and $(\mathcal{S}, \preceq_s)$ leading to two granularity hierarchies. $\mathcal{T}$ is a set of temporal granularities and $\mathcal{S}$ a set of sensor location granularities. An example is given in Fig. 1 where the arrows connect the coarser to the finer granularities.

3 Database Modeling for Sensor Data

3.1 Granularity Aware Sensor Database

We assume the reader is familiar with database notations, see for example [11] for details. Hereinafter we use G as a spatial granularity and H as a temporal granularity. We extend the temporal database definitions given in [3] to take into account the spatial dimension. Let $\mathcal{U}$ be a universe (set of attributes) and $\mathcal{D}$ be a countably infinite set of constant values. A *spatio-temporal module schema* over $\mathcal{U}$ is a triplet $M = (R, G, H)$, where $R \subseteq \mathcal{U}$ is a relation schema, $G \in (\mathcal{S}, \preceq_s)$ is a spatial granularity and $H \in (\mathcal{T}, \preceq_t)$ is a temporal granularity. For a relation schema R, $Tup(R)$ is the set of all possible tuples defined over $\mathcal{D}$. A *spatio-temporal module* is a quadruple $\mathcal{M} = (R, G, H, \varphi)$, where (R, G, H) is a *spatio-temporal module schema* and φ is a mapping from $\mathbb{N}\times\mathbb{N}$ to $\mathcal{P}(Tup(R))$. Actually, the mapping function $\varphi(i, j)$ gives the tuples over attributes of R that hold at each couple of granules $(G(i), H(j))$. When clear from context, we shall use the time instants and sensor locations instead of integers to describe a particular mapping function, e.g. $\varphi(building_x{:}house_y{:}room_z{:}sensor_i, 2016/03/02\ 11{:}59{:}00)$ instead of $\varphi(i, j)$ for some integer values i and j.

Example 3. Consider the "raw" data stream given in Table 1. It can be represented at different granularities: e.g. with the module $\mathcal{M}_1{=}(R, G_1, H_1, \varphi_1)$ where $R{=}\{temperature\}$, $G_1{=}room$, $H_1{=}hour$ and the windowing function φ_1 is:

φ_1(oxygen:f1:h1:livingRoom,2016/03/02 11) = {<21>}
φ_1(oxygen:f1:h1:kitchen,2016/03/02 11) = {<20>}
φ_1(oxygen:f1:h1:livingRoom,2016/03/02 12) = {<20>}
φ_1(oxygen:f1:h1:kitchen,2016/03/02 12) = {<23>}
φ_1(oxygen:f1:h1:bathroom,2016/03/02 12) = {<24>}

A granularity-aware *sensor database schema* $\mathbf{R}$ over $\mathcal{U}$ is a fixed set of *spatio-temporal module schemas* over $\mathcal{U}$. A granularity-aware *sensor database* $\mathbf{d}$ is a finite set of *spatio-temporal modules* defined over $\mathbf{R}$.

3.2 Spatio-Temporal Functional Dependency (STFD)

Dedicated FDs for sensor data streams have to take into account the temporal and spatial dimensions. Many extensions of FDs to temporal DB have been proposed but none of them extends FDs to both temporal and spatial dimensions. In the sequel, we extend temporal functional dependencies introduced in [3].

Intuitively, a STFD means that the X-values determine the Y-values within each granule of the spatio-temporal granularities.

Let $X, Y \subseteq \mathcal{U}$ and $(\mathcal{T}, \preceq_t)$, $(\mathcal{S}, \preceq_s)$ two granularity hierarchies. To express STFDs, we need to consider two special attributes, disjoint from $\mathcal{U}$, to take into consideration granularities. Let *location* and *time* be the special spatial and temporal attributes respectively. When clear from context, *location* and *time* will be abbreviated by L and T respectively.

Definition 1. *A spatio-temporal FD over $\mathcal{U}$ is an expression of the form: $X, location^G, time^H \rightarrow Y$ where $G \in (\mathcal{S}, \preceq_s)$ is a spatial granularity and $H \in (\mathcal{T}, \preceq_t)$ is a temporal granularity.*

We shall see that the case $X = \emptyset$ is meaningful for STFDs whereas the classical FD counterpart is almost useless (i.e. $\emptyset \rightarrow A$ means that in every possible relation, only one value for A is allowed).

Example 4. Regarding the temperature approximations, one may consider that the temperature of the same room does not change all along the same hour. This approximation can be represented as: $\emptyset, location^{room}, time^{hour} \rightarrow temperature$ (or simply: $location^{room}, time^{hour} \rightarrow temperature$).

The satisfaction of a STFD with respect to a module is defined as follows:

Definition 2. *Let $\mathcal{M} = (R, G, H, \varphi)$ be a spatio-temporal module, $X, Y \subseteq R$ and f: $X, location^{G'}, time^{H'} \rightarrow Y$ an STFD. f is satisfied by $\mathcal{M}$, denoted by $\mathcal{M} \models f$, if for all tuples t_1 and t_2 and positive integers i_1, i_2, j_1 and j_2, the following three conditions imply $t_1[Y] = t_2[Y]$: **(1)** $t_1[X] = t_2[X]$, **(2)** $t_1 \in \varphi(i_1, j_1)$ and $t_2 \in \varphi(i_2, j_2)$, and **(3)** $\exists\ i'$ such that $G(i_1) \cup G(i_2) \subseteq G'(i')$ and $\exists\ j'$ such that $H(j_1) \cup H(j_2) \subseteq H'(j')$.*

This definition extends classical FDs ($r \models X \rightarrow Y$) as follows:

(1) is the classical condition for FDs, i.e. the left-hand sides have to be equal on X for the two considered tuples.

(2) bounds tuples t_1, t_2 to be part of two spatio-temporal granules of $\mathcal{M}$ (equivalent to $t_1, t_2 \in r$).

(3) restricts eligible tuples t_1 and t_2 in such a way that the union of their spatial (resp. temporal) granules with respect to G (resp. H) has to be included in some granule of G' (resp. H').

Example 5. Let us consider the module $\mathcal{M}_2$=(R, G_2, H_2, φ_2) where R=$\{temperature\}$, G_2=$sensor$, H_2=$second$. We have: $\mathcal{M}_2 \models location^{room}, time^{hour} \rightarrow temperature$ and $\mathcal{M}_2 \not\models location^{house}, time^{hour} \rightarrow temperature$. As for FDs, the non-satisfaction is easier to explain since we just need to exhibit a counter example. The two first tuples given in Table 1 form a counter-example since both sensors belong to the house $h1$ and have been produced at the hour 11 whereas $20 \neq 21$.

From these examples, we argue that STFDs are quite natural to express *declarative constraints over sensor data streams* and provide a powerful abstraction mechanism towards granularity-aware sensor database design.

3.3 Reasoning on STFDs

Inference Axioms for STFDs. In order to derive all the possible STFDs logically implied by a set of STFDs, we need to define the inference axioms corresponding to STFDs. We propose the three following finite axioms:

(A1) Restricted reflexivity:
if $Y \subseteq X$ then $F \vdash X, L^{Top}, T^{Top} \rightarrow Y$

(A2) Augmentation:
if $F \vdash X, L^G, T^H \rightarrow Y$ then $F \vdash X, Z, L^G, T^H \rightarrow Y, Z$

(A3) Extended transitivity:
if $\begin{cases} F \vdash X, L^{G_1}, T^{H_1} \rightarrow Y \\ F \vdash Y, L^{G_2}, T^{H_2} \rightarrow Z \end{cases}$ then $F \vdash X, L^{G_3}, T^{H_3} \rightarrow Z$
where $G_3 = glb(\{G_1, G_2\})$ and $H_3 = glb(\{H_1, H_2\})$.

These three inference axioms for STFDs are a generalization of axioms of temporal FDs, shown to be sound and complete in [3]. The proof for STFDs is similar and is omitted in this paper.

Closure of Attributes. The closure of attributes plays a crucial role for reasoning on classical FDs and are generalized to STFDs as follows. Let $\mathbf{R}$ be a sensor database schema over $\mathcal{U}$, F a set of STFDs over $\mathcal{U}$ and $X \subseteq \mathcal{U}$. The closure of X with respect to F is denoted by X_F^+. X_F^+ contains elements of the form (B, G, H) over $\mathcal{U} \times \mathcal{S} \times \mathcal{T}$. X_F^+ is defined as follows:

$X_F^+ = \{(B, G, H) \mid F \vdash X, L^G, T^H \rightarrow B$ such that there is no $F \vdash X, L^{G'}, T^{H'} \rightarrow B$ *with* $G' \preceq_s G', H \preceq_t H'$ and $(G \neq G'$ or $H \neq H')\}$.

Algorithm 1 computes the finite closure of a set of attributes X with respect to F. This algorithm is a generalization of the classical closure algorithm for FDs [11] taking into account the granularities. Its basic idea is to compute progressively the set X_F^+. Line 1 encodes the first axiom (A1). The following procedure is repeated over all STFDs until a fix point is reached (line 13). For each STFD of line 4, if $A_1, .., A_k$ appears in the current closure X_{prev}, then $B_1, .., B_m$ are added to X_F^+ with the corresponding spatial and temporal granularities. Line 14 ensures that the closure is composed of elements with incomparable granularities for the same attribute.

Example 6. Let us consider $\mathcal{U} = \{temperature, humidity, luminosity, CO2\}$, four sensor types sending the temperature, humidity, luminosity and CO_2 values, the granularity hierarchies given in Fig. 1 and the set F of STFDs:

$F = \{location^{room}, time^{hour} \rightarrow temperature; location^{house}, time^{day} \rightarrow humidity;$
$location^{room}, time^{day} \rightarrow luminosity; location^{room}, time^{minute} \rightarrow CO2;$
$location^{room}, time^{hour} \rightarrow humidity; location^{room}, time^{minute} \rightarrow temperature;$
$location^{house}, time^{hour} \rightarrow humidity; location^{sensor}, time^{hour} \rightarrow luminosity;$
$location^{room}, time^{hour} \rightarrow CO2 \ \}$
The closure of *temperature* w.r.t. F is: $temperature_F^+ = \{(temperature, Top, Top), (humidity, house, day), (luminosity, room, day), (CO2, room, hour)\}$.

Algorithm 1. ClosureAttribute

Require:
F: a set of STFDs over $\mathcal{U}$
$X \subseteq \mathcal{U}$
Ensure:
X_F^+: the finite closure of X with respect to F

1: $X_F^+ := \{(A, Top, Top) \mid a \in X\} \cup \{(\emptyset, Top, Top)\}$
2: **repeat**
3: $X_{prev} := X_F^+$
4: **for each** $A_1, .., A_k, L^G, T^H \rightarrow B_1, .., B_m \in F$ **do**
5: **for each** $\{(A_1, G_1, H_1), .., (A_k, G_k, H_k)\} \subseteq X_{prev}$ **do**
6: $G' := glb(G_1, .., G_k, G)$
7: $H' := glb(H_1, .., H_k, H)$
8: **for each** $B \in \{B_1, .., B_m\}$ **do**
9: $X_F^+ := X_F^+ \cup \{(B, G', H')\}$
10: **end for**
11: **end for**
12: **end for**
13: **until** $X_F^+ = X_{prev}$
14: Minimize X_F^+ such that there is no two elements (A, G, H) and (A, G', H') with $G \preceq_s G'$, $H \preceq_t H'$ and $(G \neq G'$ or $H \neq H')$

The closure of attributes with respect to a set F of STFDs is polynomial and allows to decide whether or not a given STFD is implied by F, as shown in the following property.

Property 1. $F \vdash X, L^G, T^H \rightarrow B$ **iff** $\exists\ Y \subseteq X_F^+$ such that $Y = \{(B, G_{i_k}, H_{j_l}) \mid i_k \in i_1..i_n, j_l \in j_1..j_m\}$, $G \preceq_c \{G_{i_1}, \ldots, G_{i_n}\}$ and $H \preceq_c \{H_{j_1}, \ldots, H_{j_m}\}$.

Example 7. Let us consider the set F of STFDs given in Example 6. We consider the following STFDs:
$f_1 : temperature, location^{room}, time^{hour} \rightarrow luminosity$; and
$f_2 : temperature, location^{openspace}, time^{day} \rightarrow humidity$;
As $(luminosity, room, day) \in temperature_F^+$ and $hour \preceq_t day$ we have $F \vdash f_1$. But $F \nvdash f_2$ since $temperature_F^+$ only contains $(humidity, house, day)$ and $openspace \not\preceq_s house$.

Attribute closure is one of the technical contributions of the paper and is, to the best of our knowledge, a new result never addressed in related works.

3.4 Normalization

Our aim is to extend the well known *synthesis algorithm* for database design [11] from a set of classical FDs in our setting. With STFDs, we propose a normalization technique based on two main steps: first, computing a minimal cover of a set of STFDs and then producing spatio-temporal modules. Let F^+ be the closure of F. F^+ is defined by: $F^+ = \{X, L^G, T^H \rightarrow Y \mid\ F \vdash X, L^G, T^H \rightarrow Y\}$.

Definition 3. *A set F' of STFDs is a* cover *of a set F of STFDs if $F^+ = F'^+$. A cover F' of F is* minimal *if $\nexists$ a cover G of F such that $|G| < |F'|$.*

Algorithm 2. MinimalCover

Require:
F: a set of STFDs over $\mathcal{U}$
Ensure:
F': a minimal cover of F

1: $F' := \emptyset$
2: **for each** $X, L^G, T^H \rightarrow Y \in F$ **do**
3: **for each** $(A, G', H') \in X_F^+$ **do**
4: $F' := F' \cup \{X, L^{G'}, T^{H'} \rightarrow A\}$
5: **end for**
6: **end for**
7: **for each** $X, L^{G'}, T^{H'} \rightarrow A \in F'$ **do**
8: $F'' := F' \setminus \{X, L^{G'}, T^{H'} \rightarrow A\}$
9: **if** $F'' \vdash \{X, L^{G'}, T^{H'} \rightarrow A\}$ **then**
10: $F' := F' \setminus \{X, L^{G'}, T^{H'} \rightarrow A\}$
11: **end if**
12: **end for**
13: **while** there exists $f, f' \in F'$ with the same left-hand-side **do**
14: Merge f and f'
15: **end while**

Algorithm 2 generalizes the classical procedure to get a minimal cover of FD. We sketch the main steps to compute a minimal cover. First, we saturate the initial set of STFDs with STFDs induced by the closure of each set of attributes (line 2–6). Then, we apply classical minimization procedure (line 7–12) and finally, we merge STFDs whose left-hand sides are the same by taking the union of their right-hand sides (line 13–15) (not detailed in the Algorithm).

Example 8. We consider the set F of STFDs in example 6. Using Algorithm 2 we get the following minimal cover F' (details are omitted):

$$F' = \{location^{room}, time^{hour} \rightarrow temperature, CO2;$$
$$location^{house}, time^{day} \rightarrow humidity; location^{room}, time^{day} \rightarrow luminosity\}$$

The sensor database schema can be deduced from the obtained minimal cover: for each STFD, a module is generated. Algorithm 3 presents the main steps allowing to get a granularity-aware sensor database schema from a set of STFDs. Studying the properties of this decomposition is left for future work.

Algorithm 3. Normalization

Require:
F: a set of STFDs over $\mathcal{U}$
Ensure:
$\mathbf{R}$: a granularity-aware sensor database schema

1: $\mathbf{R} := \emptyset$
2: $F' := MinimalCover(F)$
3: **for each** $A_1, .., A_k, L^G, T^H \rightarrow B_1, .., B_m \in F'$ **do**
4: $R := \{A_1, .., A_k, B_1, .., B_m\}$
5: $M := (R, G, H)$
6: $\mathbf{R} := \mathbf{R} \cup M$
7: **end for**

Example 9. Continuing the previous example, we obtain a granularity-aware sensor database schema $\mathbf{R} = \{M_1, M_2, M3\}$ with:
$M_1 = (< temperature, CO2 >, room, hour)$, $M_2 = (< humidity >, house, day)$, and $M_3 = (< luminosity >, room, day)$.

It is worth noting that every module of such a database schema is easily implementable on top of any RDBMS. In the sequel, we denote the database schema obtained through the normalization process by the *abstract schema* because in our context we need to add more semantic information to this schema.

3.5 Semantic Value Assumption

Given a set of sensor data streams, constraints for long-term storage can be defined as a set of STFDs. We have seen that a granularity-aware sensor database schema can be obtained from them thanks to the proposed normalization algorithms which allow producing a database schema, i.e. abstract schema, from the user-defined inputs (dimensions, sensor stream schemas and STFDs). After that, the user annotates the abstract schema with the semantic information allowing to specify the "relevant" data at the right granularities. In fact, at given spatio-temporal granules, we have different alternatives to choose data, each one could be seen as an aggregation of values with some aggregate functions. These functions can be simple aggregations (e.g. first, max ...) or more elaborated ones (e.g. average of the 3 first values corresponding to a given valid domain) depending on the application context. To do so, we introduce the so-called "*Semantic Value Assumptions*" (SVA) allowing to declaratively define the values to be selected. Annotating an abstract schema, obtained by normalization techniques based on STFDs, with user-defined SVAs leads to a *concrete schema*, which can be implemented on top of classical RDBMS. The definition of SVA is:

Definition 4. *Let $M = (R, G, H)$ be a spatio-temporal module schema and $A \in R$. A SVA is a triplet (A, agg_fct, M) where agg_fct is an aggregation function (first,avg...) over the spatio-temporal granularities G and H.*

Example 10. Let us consider M_2 in Example 9. The SVA $(humidity, first, M_2)$ means that the *first* humidity value per house per day has to be kept in M_2 and the SVA $(humidity, avg, M_2)$ means that the average of the humidity values per house per day has to be kept in M_2.

Whenever multiple SVAs exist for a given couple $(attribute, module\ schema)$, new attributes could be created in the target schema of the underlying RDBMS. These specific annotations allow, in a declarative manner, to annotate the database schema with the semantic information required to indicate which value is representative in each granule. SVAs can be very complex aggregations that allow to define complex functions and even to avoid noisy and incomplete data issues. This semantic information is important in database design level as well as in the data stream loading procedure. In fact, as the relevant tuples definition is ensured thanks to SVAs, for each SVA the system instantiates a specific data

wrapper. It is possible to implement SVA in different manners namely using triggers or a dedicated middleware. As triggers do not scale to important data stream loads [1,8], we chose to implement SVA data wrappers in a middleware.

4 Prototype for Sensor Database

We propose a declarative system for both database design and data stream loading containing the following two levels:

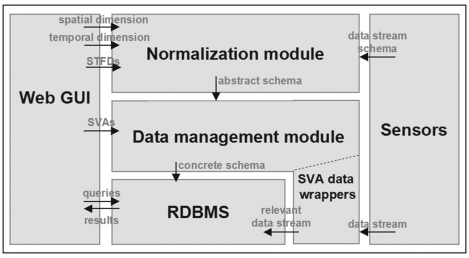

Fig. 2. An overview of our architecture

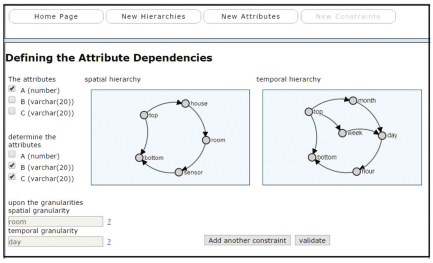

Fig. 3. STFD definition in our prototype

1. ***Sensor database* design:** Given a spatial and a temporal dimensions, the aim of this module is to determine a *granularity-aware sensor database* schema from a set of data streams, a set of STFDs and a set of SVAs.
   Once the granularity-aware sensor database schema is defined, this module allows to create the corresponding database relations in SQL, data description language implemented on top of any RDBMS.
2. **On-the-fly data loading:** Once the database is created, this level ensures the selection of the relevant data from the received sensor data streams. Thanks to STFDs and SVAs this middleware observes sensor data, chooses data to be stored and prunes the rest.

4.1 Implementation

We implemented in Java and Prolog a prototype containing the two levels presented previously. An overview of the proposed architecture is presented in Fig. 2 where we found the following main modules:

1. **Normalization module:** This module takes the user inputs (i.e. spatio-temporal dimensions, stream schemas and STFDs) and generates database schema using the algorithms presented in Sect. 3. This process leads to the database *abstract schema*. The reasoning about the spatio-temporal dimensions is proceeded through a Prolog environment. A *.pl* file, containing the *finer than* relationships between the different granularities, is generated from the input spatio-temporal dimensions. This file is checked whenever an algorithm needs to compare two granularities.

2. **Data management module:** At this stage, the user defines the SVAs corresponding to the proposed abstract database schema. This module updates the abstract schema with the corresponding semantic annotations which leads to the *concrete schema*. This module also ensures the selection of the "relevant" data (i.e. corresponding to the specification of the set of SVAs) from sensor data streams. Thus, for each SVA the system instantiates a specific *SVA data wrapper* which observes the sensor data stream concerning its attribute and identifies the accurate values. Once the user validates the obtained sensor database schema, the database relations are created. We used *Oracle 11G* for the implementation and the experiments.
3. **Sensor module:** This module is the interface between the sensors and the system. It gathers sensor data and links it to the user-defined dimensions.
4. **Web GUI:** This module allows the application manager to: **(a)** design the temporal and spatial dimensions, **(b)** from data streams at hand, define the stream schemas, **(c)** declare relevant STFDs from (a) and (b), **(d)** once an abstract schema exists, define a set of SVAs.

We believe that STFDs are natural and easy to express. Indeed, Fig. 3, contains a screen-shot from our prototype containing the user interface for STFD definition. As we can see the user just has to check the concerned attributes and select each granularity without caring about any syntax.

4.2 Ongoing Experiments

Data accuracy w.r.t. data reduction

In this section, we are mainly interested in studying the trade-off between data reduction and data accuracy with respect to some STFDs. To do so, we consider real-life sensor data. We have conducted real-life experiments in two buildings in our university. A total of around 400 heterogeneous physical sensors are deployed to measure temperature, humidity, CO2/VOC, presence, contact (for doors/windows), electricity consumption, weather conditions...

In these experiments, we consider 19 temperature sensors belonging to 7 different rooms. Each sensor sends a new value per minute. We are interested on data coming from these sensors all along one day (June, 1 2016). The idea is to compare the data reduction with respect to the considered set of STFDs. Then, to compare the considered data ("relevant data"), we observe the impact of this reduction upon the accuracy of the results. In our case, the initial sensor data stream spatio-temporal granularities are *sensor* and *minute*. All received sensor data is stored in a table called d_0 (i.e. raw data). We consider the following three sensor database tables and their corresponding STFDs:

1. sensor DB table d_1: contains the first temperature value per sensor per hour with respect to the STFD $location^{sensor}, time^{hour} \rightarrow temperature$,
2. sensor DB table d_2: contains the first temperature value per room per hour with respect to the STFD $location^{room}, time^{hour} \rightarrow temperature$, and

Table 2. Data reduction w.r.t. STFDs

Database table	Number of tuples	ratio: $\mid d' \mid / \mid d \mid$
d_0	27379 $=\mid d \mid$	100%
d_1	456	1,67%
d_2	162	0,59%
d_3	7	0,03%

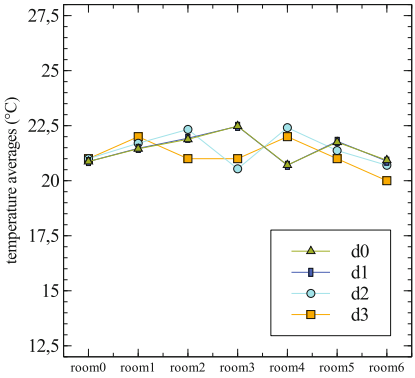

Fig. 4. Data accuracy w.r.t. data reduction

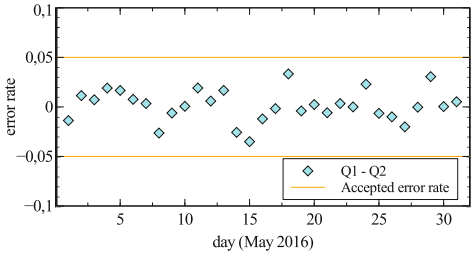

Fig. 5. Difference between the results of Q_1 and Q_2

3. sensor DB table d_3: contains the first temperature value per room per day with respect to the STFD $location^{room}, time^{day} \rightarrow temperature$.

Table 2 shows the important data reduction that may be done thanks to STFDs. As we can see, considering coarser granularities increases the data reduction. Next we are interested in data accuracy. Thus we aim to check how STFDs data approximation impacts data. To do so, we ask the different obtained database tables for the average of temperature per each room during the day. These averages are given in Fig. 4. According to this figure, approximating data with coarser granularities increases the error rate and decreases data accuracy. In order to evaluate the error rate and to check if the approximation deteriorates the evaluation of the temperature averages we are now interested in the difference between the results over *raw data* and a *sensor database*. Therefore, we executed the following queries:

1. query Q_1: computes on a *raw database* table the average of the temperature during each day of a given month for a given sensor, and
2. query Q_2: computes on a *sensor database* table (i.e. this table contains only one value per sensor per hour) the same average value.

The obtained difference values are given in Fig. 5. We can see that the difference is maintained under an error rate of 5% (most values being randomly disseminated

on the ±2% rate stripe) which could be considered acceptable in a real life settings. It also evidences the fact that even though query Q_2 considers only one value per hour for the assessment of the average temperature per day, the output of the calculation sticks to the value obtained with a finer granularity. Actually, considering coarser granularities may not have a major effect on the final results, due to the fact that sensor values may be unchangeable over some spatio-temporal granules.

Data storage

We simulate now the sensors of an intelligent building containing 10 houses, each house contains 5 rooms. In each room we consider at least 2 sensors of each type (e.g. temperature, humidity...). Thus we simulate the operation of several hundreds of sensors, each sending data at a frequency of one value per minute. We took a sensor data stream, e.g. temperature, and we stored it in different relations with different temporal and spatial granularities. For instance, if we consider the temporal granularity *hour* the concerned relation contains one temperature value per hour per sensor. And, if we consider the spatial granularity *room* the concerned relation contains one temperature value per minute per room. The experiments of this section count the number of tuples in each relation. As expected, it is clear from Fig. 6 that with finer granularities we have more tuples. However, we can note that the temporal granularities are more discriminating than the spatial granularities. So as we have seen the use of STFDs with coarser granularities may reduce the number of tuples. This leads to more efficient queries as they have less tuples to scan. We also mention that, storing relevant data according to application requirements at specific spatio-temporal granularities enables the use of simpler queries (simple SELECT queries with simple WHERE clauses instead of queries with nested SELECT and JOINS). The stored relevant data can be considered as prior-computed query results.

Sensor data loading efficiency

In this section we are interested in the total execution time over predefined duration of our prototype while storing data in the appropriate database relations. Thus, we compare some implemented SVAs with the baseline solution, i.e. storing all received sensor data. Each sensor sends one value per minute.

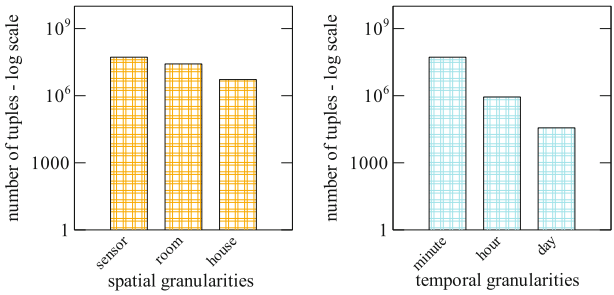

Fig. 6. Number of tuples when varying temporal and spatial granularities

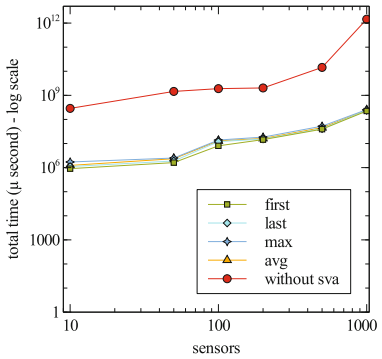

Fig. 7. Total execution time w.r.t. the number of sensors

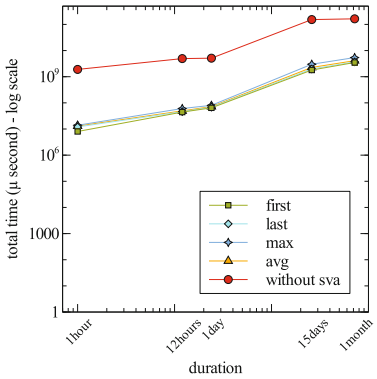

Fig. 8. Total execution time w.r.t. the duration

The stored data in the following sets of experiments is approximated upto the spatio-temporal granularities *room* and *hour*. We compare the total execution time of the baseline solution with four SVAs (*first*, *last*, *maximum* and *average*). In this section we focus on the temperature stream. First, we focus on the variation of the total execution time with respect to the sensor number. We limit the scope of these experiments on a duration of 1 h. The obtained results are given in Fig. 7. Then, we vary the duration of our experiments for a fixed number of sensors (100 sensors). Using SVAs in order to select on-the-fly the relevant data is more efficient than the baseline solution: the total time in the case of SVAs is lower by a 10^3 order of magnitude in Fig. 7 and by a 10^2 order of magnitude in Fig. 8 (in a *log* scale).

5 Related Work

As far as we know there is no many contributions aiming at using the spatial and the temporal dimension in order to retrieve the relevant data from sensor data streams. Nowadays sensor data management can be seen from two points of view: real-time, i.e. continuous queries [1,2] versus historical [6,12,13] data management. In this paper, we were interested in long-term storage of sensor data. Our aim is to decrease the storage space and increase query efficiency of long-term reporting applications. Some approaches were interested in resolving storage problems that can result from the important amount of data generated by sensors. As far as we know, there is no formal approach for dealing with spatio-temporal stream querying considering different granularities.

TFDs have been mainly introduced in order to constraint the temporal data in temporal databases. There have been an important number of articles aiming at defining and characterizing TFDs namely [9,15–18]. The three first approaches [9,15,17] handle TFD without time granularity. In [15], the author defined a temporal relation as a temporal sequence of database states and extended each tuple

with its updated version. The data model in [17] was extended with a valid time which represents a set of time points. The author presented the suitable definition of FD in the presence of valid time and defines two classes of temporal dependencies: Temporal Functional Dependencies (TFDs) and Dynamic Functional Dependencies (DFDs). In [9], data contains two time dimensions: valid time and transaction time. The authors handle the problem of expressing the functional dependencies with such data. These works do not consider granularity as a central notion as we do in this paper. Both [16,18] handled multiple time granularities. The authors in [16] and in [3] defined the time granularities and the different relationships between them. They defined the *temporal module schema* and the *temporal schema* as well as TFD. In [18], the author extended the dependencies presented in [17] using time granularity and object identity which is a time-invariant identity that relates the different versions of the same object.

Roll-up Dependencies (RUDs) [19] define dependencies with a higher abstraction level for OLAP DB. They extend TFDs to non temporal dimensions allowing each attribute to roll up through the different levels of its associated hierarchy. Algorithmic aspects such as attribute closure is not studied for RUD, as we do in this paper. We just need two dimensions, i.e. temporal and spatial. In fact, we distinct two particular attributes, for spatial and temporal dimensions, and we combine them with classical attributes (i.e. attributes without associated hierarchies). Moreover, unlike [19] which deals with schemas, we deal with attributes and we propose a normalization algorithm and a new closure algorithm of a set of attributes from a set of STFDs. Our approach is not comparable to OLAP since our main goal is to retrieve and organize sensor data and not to analyze multidimensional data from multiple perspectives.

We also defined a declarative structure, i.e. SVA, which annotates the generated database design in order to enrich it with semantic information about relevant data selection. SVAs are also useful to select on-the-fly relevant data. In fact, SVAs represent a sort of data exchange [7] mechanism inspired from interval-based semantic assumptions designed for temporal databases [3]. The point-based and interval-based semantic assumptions in [3] can be used to derive or compress temporal data.

The use of SVAs in order to choose the relevant tuples reminds us load shedding techniques. The load shedding process [14] intends to reduce the workload of the data stream management system by dropping tuples from the system. Several approaches proposed different tuples dropping strategies, e.g. random [10], with a priority order [4] or a semantic strategy [5]. The load shedding usually interferes in the physical plan of the query while our approach aims to interfere from the database design and to take into account predefined approximations. Our contribution can be thought as a "declarative load shedding process" since we allow to prune data stream from declarative constraints, instead of sampling techniques.

6 Conclusion

In this paper, we have considered the long-term storage problem of sensor data streams. We have presented a declarative approach to build a granularity-aware sensor database from sensor data streams on top of any RDBMS. Our core idea is to take into account the spatial and temporal aspects of sensor data streams thanks to Spatio-Temporal Functional Dependencies (STFDs) and to adapt the classical normalization techniques developed for relational databases to sensor databases. We have defined a dedicated normalization algorithm based on a novel closure algorithm for STFDs. The closure of attributes plays a crucial role in the generation of a minimal cover of a set of STFDs and thus in the production of normalized sensor database schemas. We have also defined Semantic Value Assumption (SVA), a declarative database schema annotation, allowing to specify the mechanism to load, on-the-fly and automatically, the relevant data into the sensor database. A prototype has been implemented in order to test both sensor database design from STFDs and data loading techniques. We have conducted experiments on real and synthetic data streams from intelligent buildings. We discussed the trade-off between the data accuracy and the data reduction.

We have highlighted our proposition in the context of intelligent buildings for domestic sensors. Nevertheless, our proposition relies upon clear theoretical foundations that enable to take both spatial and temporal dimensions into account for sensor data streams. The approach is quite versatile and could be adopted in a wide range of application contexts. Many extensions could be done, for instance to consider *mobile sensors* or to study specific properties of the decomposition algorithm for STFDs.

Acknowledgments. This work is supported by the ARC6 program of the Rhône-Alpes region, France.

References

1. Abadi, D.J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: a new model and architecture for data stream management. VLDB J. Int. J. Very Large Data Bases **12**(2), 120–139 (2003)
2. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: semantic foundations and query execution. VLDB J. Int. J. Very Large Data Bases **15**(2), 121–142 (2006)
3. Bettini, C., Jajodia, S., Wang, S.: Time Granularities in Databases, Data Mining, and Temporal Reasoning. Springer, Heidelberg (2000)
4. Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S.: Monitoring streams: a new class of data management applications. In: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB 2002, pp. 215–226. VLDB Endowment (2002)
5. Das, A., Gehrke, J., Riedewald, M.: Approximate join processing over data streams. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD 2003, pp. 40–51. ACM, New York (2003)

6. Diao, Y., Ganesan, D., Mathur, G., Shenoy, P.J.: Rethinking data management for storage-centric sensor networks. In: CIDR, vol. 7, pp. 22–31 (2007)
7. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: getting to the core. ACM Trans. Database Syst. (TODS) **30**(1), 174–210 (2005)
8. Golab, L., Özsu, M.T.: Data stream management. Synth. Lect. Data Manage. **2**(1), 1–73 (2010)
9. Jensen, C.S., Snodgrass, R.T., Soo, M.D.: Extending existing dependency theory to temporal databases. IEEE Trans. Knowl. Data Eng. **8**(4), 563–582 (1996)
10. Kang, J., Naughton, J.F., Viglas, S.D.: Evaluating window joins over unbounded streams. In: Proceedings of 19th International Conference on Data Engineering, pp. 341–352. IEEE (2003)
11. Levene, M., Loizou, G.: A Guided Tour of Relational Databases and Beyond. Springer, London (2012)
12. Lewis, M., Cameron, D., Xie, S., Arpinar, B.: ES3N: a semantic approach to data management in sensor networks. In: Semantic Sensor Networks Workshop (2006)
13. Petit, L., Nafaa, A., Jurdak, R.: Historical data storage for large scale sensor networks. In: Proceedings of the 5th French-Speaking Conference on Mobility and Ubiquity Computing, pp. 45–52. ACM (2009)
14. Tatbul, N., Çetintemel, U., Zdonik, S., Cherniack, M., Stonebraker, M.: Load shedding in a data stream manager. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29, pp. 309–320. VLDB Endowment (2003)
15. Vianu, V.: Dynamic functional dependencies and database aging. J. ACM (JACM) **34**(1), 28–59 (1987)
16. Wang, X.S., Bettini, C., Brodsky, A., Jajodia, S.: Logical design for temporal databases with multiple granularities. ACM Trans. Database Syst. (TODS) **22**(2), 115–170 (1997)
17. Wijsen, J.: Design of temporal relational databases based on dynamic and temporal functional dependencies. In: Clifford, J., Tuzhilin, A. (eds.) Recent Advances in Temporal Databases. Workshops in Computing, pp. 61–76. Springer, London (1995). doi:10.1007/978-1-4471-3033-8_4
18. Wijsen, J.: Temporal FDS on complex objects. ACM Trans. Database Syst. (TODS) **24**(1), 127–176 (1999)
19. Wijsen, J., Ng, R.T.: Temporal dependencies generalized for spatial and other dimensions. In: Böhlen, M.H., Jensen, C.S., Scholl, M.O. (eds.) STDBM 1999. LNCS, vol. 1678, pp. 189–203. Springer, Heidelberg (1999). doi:10.1007/3-540-48344-6_11
20. Zilio, D.C., Rao, J., Lightstone, S., Lohman, G., Storm, A., Garcia-Arellano, C., Fadden, S.: DB2 design advisor: integrated automatic physical database design. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30, pp. 1087–1097. VLDB Endowment (2004)