A Model for a Multiresolution Time Series Database System

A. LLUSÀ-SERRA T. ESCOBET-CANAL S. VILA-MARTA
Universitat Politècnica de Catalunya
Department of Electronic Systems Design and Programming
Av. Bases de Manresa 61–73, 08242 Manresa
ES-CT
aleix@dipse.upc.edu teresa.escobet@upc.edu sebas@dipse.upc.edu

Abstract: In this paper we propose a model for multiresolution time series database management systems. This model stores compactly a time series and manages consistently its temporal dimension. This is achieved by extracting different resolutions and attributes summaries from the original time series. Our work is concerned in putting together two areas of study: time series analysis and database management systems. Time series analysis offers a great deal of methodologies and algorithms to process time series data. Database systems field provides software expertise in managing data. Therefore, for many applications it is of primary relevance that database systems support time series.

Key–Words: time series, data model, database systems, monitoring systems.

1 Introduction

The emergence of embedded systems and sensor networks has made possible the collection of large amounts of data for monitoring and control of complex systems. Nevertheless, before using the collected signals, it is of primary importance to detect the eventual sensor failures or malfunctions and to reconstruct the incorrect signals. This avoids to process misleading information which may lead to unsafe or inefficient actions. Acquired data is associated with a time stamp, which implies that the correctness of those data depends not only on the measured value but also on the time as it is collected. When observations are collected at specific time intervals, large data sets in the form of time series are generated, [2].

Time series are defined as a collection of observations made chronologically, [7], accordingly they are also called time sequences, [8]. Time series are usually stored in a database. Usually the managing software to store this data are relational database management systems (RDBMS). However, using a RDBMS as a time series back-end suffers of some drawbacks, [6, 14, 15, 17]. Time series come from a continuous nature phenomenon. They are recorded at regular intervals, say hourly, or at irregular intervals, such as recording when a pump starts running.

There are two main problems when managing time series. The first results from these data being voluminous, [7]. Because of this, storing and accessing them can be difficult. Moreover, this is critical when developing small embedded systems, whose resources (capacity, energy, processing, and communications) suffer a genuine restriction, [16]. The second problem concerns the procedure of processing and synthesising information from the time series data, that becomes challenging when data is not equi-time spaced.

This paper focuses on Data Base Management Systems (DBMS) that store and treat data as time series. These are usually known as Time Series Data Base Management Systems (TSMS), [6]. We introduce a new data model named multiresolution TSMS (MTSMS). This model organises data in an aggregated way and it allows to store time series using different resolutions. It is designed to cope well with bounded storage computers such as sensor systems.

This manuscript is organised as follows. In Section 2 some related work concerning TSMS are presented and a summary of its features is shown in Section 3. The MTSMS model is formalised as follows: the nomenclature preliminaries are introduced in Section 4, the data structure is formulated in Section 5, and attribute aggregation is summarised in Section 6. Section 7 is devoted to a real data multiresolution database example. Finally, Section 8 offers some conclusions.

2 Related work

There are some prior works concerning TSMS. RRDtool from Oetiker, [13], is a free software database management system. It is designed to be used for monitoring systems. Because of this, it is focused to a particular kind of data, gauges and counters, and it lacks general time series operations. RRDtool can store multiple time resolution data. The work in this paper is partially inspired in RRDtool.

Cougar, [3], is a sensor database system. It has two main structures: one for sensor properties stored into

ISBN: 978-1-61804-162-3

relational tables and another for time series stored into data sequences from sensors. Time series have specific operations and can combine relations and sequences. Cougar target field is sensor networks, where data is stored distributed in sensors. Queries are resolved combining sensor data using a data stream abstraction that improves processing performance.

SciDB, [15], and SciQL, [17], are array database systems. These systems are intended for science applications, in which time series play a principal role. They structure time series into arrays in order to achieve multidimensional analysis and allow tables to store other data. SciDB is based on arrays which, according to the authors, allow to represent time series. However, it does not consider time series special needs: it does not care for managing continuously voluminous data neither for achieving temporal coherence. In contrast, SciQL exhibits some time series managing characteristics that include time series regularities, interpolation or correlation queries. However, difference between tables and arrays seems too physical and leads to ambiguity when representing time series.

Bitemporal DBMS is another database field related with time. Bitemporal data target is to keep historical events in the database by associating time intervals to data. Bitemporal data and time series data are not exactly the same and so can not be treated interchangeably, [14]. However, there are some similarities that can be considered. First, extending a relational model to manage bitemporal data illustrate the extension of RDMBS with new types and how to model them. Second, bitemporal data modelling settles some time-related concepts that can be extended to time series.

The recent bitemporal data research in relational DBMS model terms, [5], marks a promising foundation. It models bitemporal data as relations extended with time intervals attributes and extends relational operations in order to deal with related time aspects.

3 TSMS features

A TSMS is a special purpose DBMS devoted to store and manage time series. The main objective of TSMS is to put together two areas of study: time series analysis and DBMS. Time series analysis formalises a great amount of algorithms and methodologies that apply to time series, with a main focus on improving efficiency. DBMS theory formalises systems that store and operate with data. Currently the relational model, [4], is the referent.

In time series analysis there are some common generic operations. Most of these operations deal with the time given the nature of data. Usual operations include the query of time intervals, to find time correlations, or to calculate distances between two time series. In all these operations TSMS must consider the temporal coherence of the time series. In the context of statistics, aggregation of time series is also a common operation. Aggregate means to summarise a time series subset by a smaller set of measures. Statistic indicators like the mean, the maximum, or the mode, for instance, summarise time series into a only measure.

A time series is defined discrete as a set of value and time pairs. Furthermore, a time series has a continuous nature as it comes from a phenomena evolution along time. As a result, TSMS operations may deal with this time series nature by methods of interpolation or approximation.

A MTSMS is a TSMS with multiresolution capabilities. A MTSMS schema represents a time series using a set of table like structures each of them representing the series at a different resolution.

4 Preliminaries

In this section we introduce some background concepts and the nomenclature which we will use later. First we define the main objects of a MTSMS which are measures and time series.

A *measure* is a value measured in a time instant. More formally it is a tuple (v,t) where v is the value of the measure and $t \in \mathbb{R}$ is the time instant of measurement. The values of a time series can be of any type. For simplicity examples are presented with integers or real numbers but can also be strings or vectors. Let $m = (v,t)$ be a measure, v is written as $V(m)$ and t is written as $T(m)$.

The time value defines the canonical order between measures. Let $m = (v_m, t_m)$ and $n = (v_n, t_n)$ be two measures, then $m \geq n$ if and only if $t_m \geq t_n$.

A *time series* is sequence of measures of the same phenomena that are ordered in time.

Definition 1 (Time series) *A* time series *S is a a set of measures of the same phenomena $S = \{m_0, \ldots, m_k\}$ without repeated time values $\forall i, j : i \leq k, j \leq k, i \neq j : T(m_i) \neq T(m_j)$. Given a time series $|S|$, we note its size by $|S| = k+1$. Observe that, because measures in S are of the same phenomena, the type of S values is homogeneous.*

The order defined by measures implies a total order in a time series. As a time series is a finite set, if it is not empty it has a maximum and a minimum. Let $S = \{m_0, \ldots, m_k\}$ be a time series and $n \in S$ be a measure. The time series' maximum is $n = \max(S)$ if and only if $\forall m \in S : n \geq m$. Similarly, the time series' minimum is $n = \min(S)$ if and only if $\forall m \in S : n \leq m$.

Given the order defined by time, in a time series we define the sequence interval following [8, 10]. Let $S = \{m_0, \ldots, m_k\}$ be a time series. We define the subset $S(r,t] \subseteq S$ as the time series $S(r,t] = \{m \in S | r <$

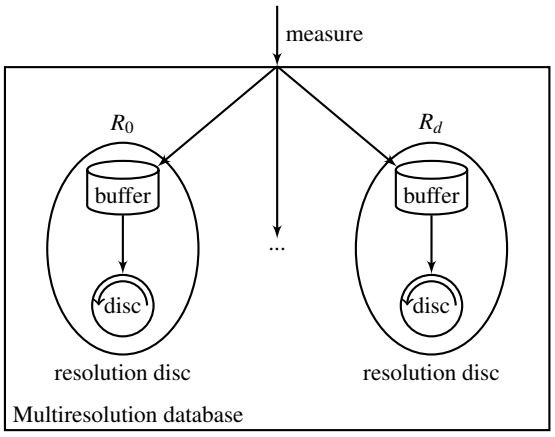

Figure 1: Architecture of MTSMS model

$T(m) \leq t\}$, where r and t are two instants in time. We also define the subset $S(r,+\infty) \subseteq S$ as the time series $S(r,+\infty) = \{m \in S | r < T(m) \leq T(\max(S))\}$ and the subset $S(-\infty,t) \subseteq S$ as the time series $S(-\infty,t) = \{m \in S | T(\min(S)) \leq T(m) < t\}$.

The time order in time series also implies the sequence concept of next and previous measure. Let $S = \{m_0,\ldots,m_k\}$ be a time series and $l \in S$ and n be two measures. We define the next measure of n in S as $l = \text{next}_S(n)$ where $l = \min(S(T(n),+\infty))$. We define the previous measure of n in S as $l = \text{prev}_S(n)$ where $l = \max(S(-\infty,T(n)))$.

Let S be a time series, t be a time instant and δ be a time duration, then the time series' measures can be located in the time interval $i_0 = [t,t+\delta]$ and its multiples $i_j = [t+j\delta,t+(j+1)\delta]$ for $j = 0,1,2,\ldots$. When time series' measures are equally spaced we say it to be regular.

Definition 2 (Regular time series) *Let $S = \{m_0$, $ldots$, $m_k\}$ be a time series and δ a time duration. S is regular if and only if $\forall m \in S(T(\min(S),+\infty)$: $T(m) - T(\text{prev}_S(m)) = \delta$.*

5 The proposed data model

The MTSMS model is an storage solution for a time series where, in short, the information is spread in different time resolutions. The objects of MTSMS are measures and time series as defined in Section 4 and each MTSMS database contains only one time series.

The general schema of the MTSMS model can be seen in the Figure 1. A multiresolution database is a collection of resolution discs, which temporarily accumulate the measures in a buffer where they are processed and finally stored in a disc. The data process is mainly intended to change the time intervals between measures in order to compact the time series information. In this way, the time series gets stored in different time resolutions spread in the discs.

Discs are size bounded so they only contain a fixed amount of measures. When a disc becomes full it discards a measure. Thus, multiresolution database is bounded in size and the time series gets stored in pieces, that is time subseries.

Regarding to operations, MTSMS structure needs operators to change the time intervals between measures. Most of these operators are attribute aggregate functions and consolidation actions.

In what follows we describe the basic MTSMS model centered in: (i) the four basic data model elements —buffer, disc, resolution disc, and multiresolution database—, and (ii) the operations to create a multiresolution database, to add measures, and to consolidate time series. Attribute aggregate functions are required but not linked to the model. They are defined in the Section 6.

A *buffer* is a container for a regular or a no-regular time series. The buffer objective is to regularise the time series using a predetermined step and an attribute function. We name *consolidation* to this action.

Definition 3 (Buffer) *A* buffer *is defined as the tuple (S,τ,δ,f) where S is a time series, τ is the last consolidation time, δ is the duration of the consolidation step and f is an attribute aggregate function.*

An empty buffer $B_\emptyset = (\emptyset,t_0,\delta,f)$ has an empty time series, an initial consolidation time t_0 and predetermined δ and f. From the $B_\emptyset$ all the consolidation time instants can be calculated as $t_0 + i\delta, i \in \mathbb{N}$.

Operator *addBuffer* adds a measure to its time series: addBuffer : $B = (S,\tau,\delta,f) \times m \mapsto (S',\tau,\delta,f)$ where $S' = S \cup \{m\}$.

A buffer is ready to consolidate when the time of some measure is bigger than the buffer's next consolidation time. Let $B = (S,\tau,\delta,f)$ be a buffer and $m = \max(S)$ the maximum measure, B is ready to consolidate if and only if $T(m) \geq \tau + \delta$. The consolidation of B in the time interval $i = [\tau,\tau+\delta]$ results in a measure $m' = (v,\tau+\delta)$ where $m' = f(S,i)$ and f is an attribute aggregate function f. Operator *consolidateBuffer* consolidates a set of measures and removes the consolidated part of the time series from the buffer. Usually consolidateBuffer is only applied to the present consolidation interval and it is defined as follows: consolidateBuffer : $B = (S,\tau,\delta,f) \mapsto B' \times m'$ where $B' = (S',\tau+\delta,\delta,f)$, $S' = S$ and $m' = f(S,[\tau,\tau+\delta])$. When historic data is not needed anymore the consolidated buffer measures can be removed applying $S' = S(\tau+\delta,\infty)$.

A *disc* is a finite capacity measures container. A time series stored in a disc has its cardinal bounded. When the cardinal of the time series is to overcome the limit, some measures need to be discarded.

ISBN: 978-1-61804-162-3

Definition 4 (Disc) *A* disc *is a tuple* (S,k) *where* S *is a time series and* $k \in \mathbb{N}$ *is the maximum allowed cardinal of* S*. An empty disc* $D_\emptyset = (\emptyset,k)$ *has an empty time series and the* k *maximum cardinal allowed.*

The cardinal of the times series is kept under control by the add operator, addDisc : $D = (S,k) \times m \mapsto (S',k)$ where

$$S' = \begin{cases} S \cup \{m\} & \text{if } |S| < k \\ (S - \{\min(S)\}) \cup \{m\} & \text{otherwise} \end{cases}$$

A *resolution disc* is a disc which stores a regular time series. It is composed of a buffer, that contains the partial time series to be regularised, and a disc, that contains the regularised time series.

Definition 5 (Resolution disc) *A* resolution disc *is a tuple* (B,D) *where* B *is a buffer and* D *is a disc. An empty buffer and empty disc imply an empty resolution disc* $R_\emptyset = (B_\emptyset, D_\emptyset)$.

The operators of a resolution disc extend the buffer and disc ones: (i) The addition of a measure to the buffer of the resolution disc, addRD : $R = (B,D) \times m \mapsto R'$ where $R' = (B',D)$, and $B' = \text{addBuffer}(B,m)$; (ii) The consolidation of the resolution disc by consolidating its buffer and adding the consolidation measure to its disc, consolidateRD : $R = (B,D) \mapsto R'$ where $R' = (B',D')$ and $(B',m') = \text{consolidateBuffer}(B)$ and $D' = \text{addDisc}(B,m')$.

A *multiresolution database* is a set of resolution discs which share the input of measures, that is they store the same time series. A time series is stored regularised and distributed with different resolutions in the various resolution discs, as it was shown in the Figure 1.

Definition 6 (Multiresolution Database) *A* Multiresolution Database *is a set of resolution discs* $M = \{R_0,\ldots,R_d\}$*. An empty multiresolution database has empty resolution discs* $M_\emptyset = \{R_{0_\emptyset},\ldots,R_{d_\emptyset}\}$.

We define the addition of a measure to every resolution disc as addMD : $M = \{R_0,\ldots,R_d\} \times m \mapsto \{R'_0,\ldots,R'_d\}$ where $R'_i = \text{addRD}(R_i,m)$.

The consolidation of all resolution discs can be defined as follows: consolidateMD : $M = \{R_0,\ldots,R_d\} \mapsto \{R'_0,\ldots,R'_d\}$ where

$$R'_i = \begin{cases} \text{consolidateRD}(R_i) & \text{if } R_i \text{ ready to consolidate} \\ R_i & \text{otherwise} \end{cases}$$

.

6 Attribute aggregate function

When a buffer is consolidated we summarise the time series information using an attribute aggregate function. Let S be a time series and t_0 and t_f two time instants, an attribute aggregate function f calculates a measure that summarises the measures of S included in the time interval $i = [T_0,T_f]$:

$$f : S = \{m_0,\ldots,m_k\} \times [T_0,T_f] \mapsto m'$$

To summarise a time series we can use different attribute aggregate functions. For instance, we can calculate an statistic indicator of the time series such as the average or we can apply a more complex digital signal processing operation, [17].

Below there are some examples. Let $S' = S(T_0,T_f]$. Then:

- maximum^d: $S \times i \mapsto m'$ where $V(m') = \max_{\forall m \in S'}(V(m))$. It summarises S' with the maximum of the measure values.
- last^d: $S \times i \mapsto m'$ where $V(m') = \max(S')$. It summarises S' with the maximum measure.
- arithmetic mean^d: $S \times i \mapsto m'$ where $V(m') = \frac{1}{|S'|} \sum_{\forall m \in S'} V(m)$. It summarises S' with the mean of the measure values.

In the design of the attribute aggregate function we can interpret a time series in different ways, that is what we call the representation of a time series. Keogh et al., [10], cite some possible representations for time series such as Fourier transforms, wavelets, symbolic mappings or piecewise linear representation. The last one is very usual due to its simplicity, [9].

Time series representations can be taken into account when computing with the measures of the time series. For example, a maximum attribute aggregate function may give different values if we consider a linear or a constant piecewise representation.

Following we show a possible family of attribute aggregate functions for time series represented by a staircase function, that is with a piecewise constant representation. We define a new representation for time series named *zero-order hold backwards* (zohe). This representation holds back each value until the preceding value. RRDtool, [12], has a similar aggregate function.

Let $S = \{m_0,\ldots,m_k\}$ be a time series, we define $S(t)^{\text{zohe}}$ as its continuous representation along time t: $\forall t \in \mathbb{R}, \forall m \in S$:

$$S(t)^{\text{zohe}} = \begin{cases} \infty & \text{if } t > T(\max S) \\ V(m) & \text{if } t \in (T(\text{prev}_S m), T(m)] \end{cases} \quad (1)$$

In conclusion, we can define many attribute aggregate functions and thus no global assumptions can

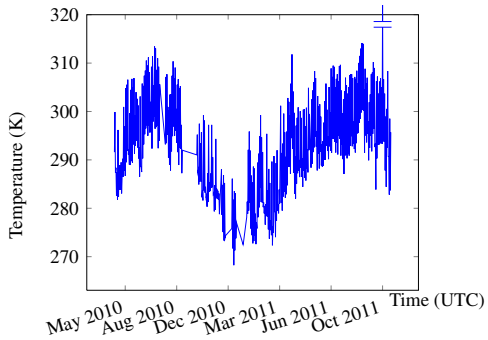

Figure 2: Example of a temperature time series data

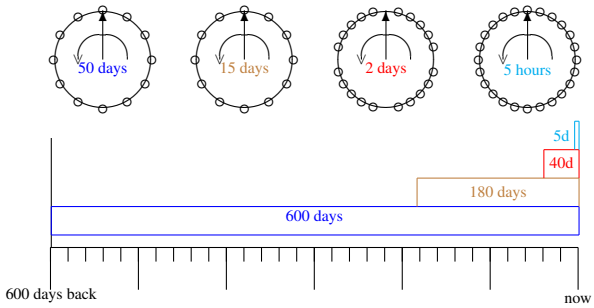

Figure 3: Schema of resolutions in a MTSDB

be made about them. Each user has to decide which combination of aggregation and representation fits better with the measured phenomena. Therefore, MTSMS must allow to define user aggregate functions.

7 Example

Next we show a real example database for a time series data. Actual data comes from a temperature distributed sensor monitoring system, [1]. We focus on one sensor data.

The Figure 2 shows the original data for one year and a half. The plot interpolates linearly the measures. In this plot we can see that there is missing data and some outlying observations. There are 146 709 stored values.

Schema. We design a multiresolution time series database that stores a time series with high resolution at recent times and with low resolution at older times. The schema is illustrated in the Figure 3. At the top there are four discs with different number of measures and at the bottom there is a timeline showing the time series chopped along time. Going from most to least granularity disks are configures as follows: (i) a measure every 5 h in the fourth disc which has a capacity of 24 measures and thus it spans 5 days; (ii) a measure every 2 days in the third disc, with a capacity of 20 thus spanning 40 days; (iii) a measure every 15 days in the second disc, with a capacity of 12 thus spanning 180 days and; (iv) a measure every 50 days in the first disc that, with a capacity of 12 results in a span of 600 days.

Attribute aggregate functions. In order to illustrate

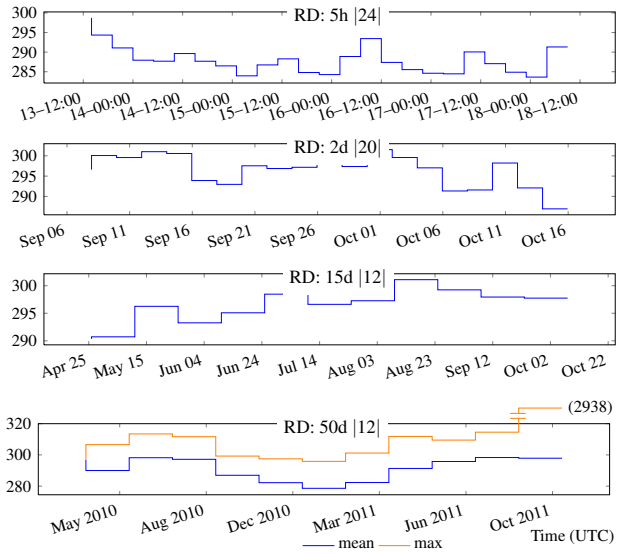

Figure 4: Resolution discs' time series in a MTSDB

this example we consolidate all the resolution discs using the zohe arithmetic mean aggregate function and the highest resolution disc using the zohe maximum aggregate function. Next, we show the process in designing both aggregate functions.

In accordance to the zohe Equation 1 defined using left-continuous step functions, we define the zohe attribute aggregate function family as the one interpreting the consolidation time interval left-continuous $i = (T_0, T_f]$ and the one aggregating on the subset $S' = S(T_0, T_f] \cup \{\min(S - S(-\infty, T_f))\}$:

- maximum^{zohe}: $S \times i \mapsto m'$ where $V(m') = \max_{\forall m \in S'}(V(m))$ and $T(m') = T_f$.
- arithmetic mean^{zohe}: $S \times i \mapsto m'$ where $V(m') = \frac{1}{|S'|} \sum_{\forall m \in S'} V(m)$ and $T(m') = T_f$.

The time series after consolidating the MTSDB are shown in the Figure 4, where each graphic corresponds to a resolution disc time series. Each title shows the disc resolution and its cardinal, and each attribute aggregate function has different colour. Each time series is plotted with zohe continuous representation. Time axis has UTC units rounded to nearest time points and temperature axis has Kelvin units. Outlayers are marked as discontinuities, for instance see fourth plot's 2938 K maximum.

In all the four plots, we can see that mean aggregate function has filled missing data and filtered outlayer observations. This is due to the aggregate function coming from a zohe interpretation.

Data can be queried. For example, one query would be the union of the four time subseries choosing the one with the highest resolution as shown in the Figure 5. Each time series is plotted interpolating linearly its measures, note that this linearly visualisation seems right time displaced as time series comes from a zohe

ISBN: 978-1-61804-162-3

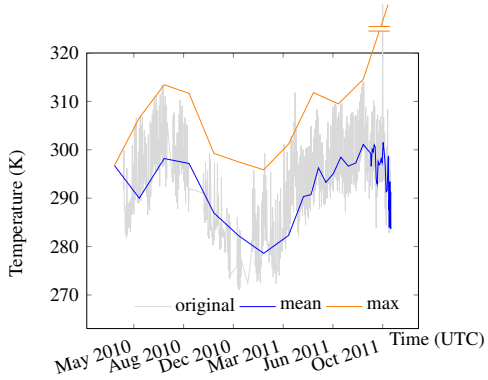

Figure 5: All time series united from the MTSDB

aggregation. Comparing this figure with the original series, see Figure 2, we observe that it resembles an incremental low-pass filter because we applied mean aggregation while the maximum aggregation resembles an envelope function.

Note that this MTSDB example schema does not store the complete original data but an approximation to the original function which contains more information for recent times.

8 Conclusion

In this paper we have shown a MTSMS model, including the requirements for these special systems and how they can be applied to an example time series. The main objective is to store compactly a time series and manage consistently its temporal dimension.

Our MTSMS model proposes to store a time series split into time subseries, which we call resolution discs. Each resolution disc has a different resolution and is compacted with an attribute aggregate function. Therefore, in a multiresolution database the configuration parameters are the quantity of resolution discs and each of their three parameters: the consolidation step, the attribute aggregate function and the capacity.

The data model shown is the first step to develop a complete model for a MTSMS. In the future the operations will be defined. In this context, there is a need for a model collecting generic properties for the TSMS, as it can be the time series union operation or the time interval operations. Then, the multiresolution model would be build upon the generic TSMS model.

Concluding, in this paper we show that using TSMS facilitates substantially time series management. The current field interest makes us optimistic to expect soon an adequate management in DBMS.

Acknowledgements: The research presented in this paper has been supported by Spanish research project NASP (TEC2012-35571), SHERECS (DPI2011-26243), the UE project i-Sense (FP7-ICT-270428), and Universitat Politècnica de Catalunya.

References:

[1] C. Alippi et al., An hybrid wireless-wired monitoring system for real-time rock collapse forecasting, *MASS '10*, IEEE, 2010, pp224 –231.

[2] S. Basu and M. Meckesheimer, Automatic outlier detection for time series: an application to sensor data, *Knowledge and Information Systems* 11.2, 2007, pp137–154.

[3] P. Bonnet et al., Towards Sensor Database Systems, *MDM '01*, Springer-Verlag, 2001, pp3–14.

[4] C. J. Date, An Introduction to Database Systems, 7th ed., Addisson-Wesley, 2000.

[5] C. J. Date et al., Temporal Data and the Relational Model, Morgan Kaufmann, 2002.

[6] W. Dreyer et al., Research perspectives for time series management systems, *SIGMOD Record* 23.1, 1994, pp10–15.

[7] T. chung Fu, A review on time series data mining, *Engineering Applications of Artificial Intelligence* 24.1, 2011, pp164–181.

[8] M. L. Hetland, A survey of recent methods for efficient retrieval of similar time sequences, in [11], chap. 2, pp23–41.

[9] E. Keogh et al., Locally adaptive dimensionality reduction for indexing large time series databases, *SIGMOD '01*, ACM, 2001, pp151–162.

[10] E. Keogh et al., Segmenting time series: a survey and novel approach, in [11], chap. 1, pp1–21.

[11] M. Last et al., eds., Data mining in time series databases, *Series in Machine Perception and Artificial Intelligence* 57, World Scientific, 2004.

[12] T. Oetiker, MRTG The Multi Router Traffic Grapher, *LISA '98*, USENIX Association, 1998, pp141–148.

[13] T. Oetiker, RRDtool, Round Robin database, 1998–2011, URL: `http://oss.oetiker.ch/rrdtool/` 10/10/2011.

[14] D. Schmidt et al., Time Series, A Neglected Issue in Temporal Database Research?, *Workshops in Computing*, Springer, 1995, pp214–232.

[15] M. Stonebraker et al., Requirements for Science Data Bases and SciDB, *CIDR '09*, `www.cidrdb.org`, 2009.

[16] Y. Yao and J. Gehrke, The Cougar Approach to In-Network Query Processing in Sensor Networks, *SIGMOD Record* 31.3, 2002, pp9–18.

[17] Y. Zhang et al., SciQL: bridging the gap between science and relational DBMS, *IDEAS '11*, ACM, 2011, pp124–133.