Model and requirements for a Multiresolution Time Series Database Management System

A. Llusà Serra, T. Escobet Canal and S. Vila-Marta
{aleix, sebas}@dipse.upc.edu, teresa.escobet@upc.edu
Department of Electronic System Design and Programming (DiPSE)
Universitat Politècnica de Catalunya, Manresa, ES–CT

December 17, 2012

In this paper we define a model for multiresolution time series database management systems. The main objective is to store compactly a time series and manage consistently its temporal dimension. It is achieved by extracting different resolutions and attributes summaries from the time series.

Our work is concerned in putting together two areas of study: time series analysis and database management systems (DBMS). Time series analysis offers a great deal of methodologies and algorithms to process time series data and database field provides software expertise in managing data. Therefore it is of primary relevance that DBMS support time series.

Keywords: time series, data model, database systems, monitoring systems.

1 Introduction

Modern society depends on the expert operation of many complex engineering systems which provides products and services. Examples include electric power systems, water distributions networks, transportation systems, manufacturing processes, intelligent buildings, communication systems, etc. The emergence of embedded systems and sensor networks has made possible the collection of large amounts of data for monitoring and control of such complex systems. In most application, these data need to be processed and synthesised efficiently to provide relevant information to engineers, researchers, accident investigators, operators, and many other users.

Nevertheless, before using the collected signals, it is of primary importance to promptly detecting eventual sensor failures or malfunctions and possibly reconstructing the incorrect signals in order to avoid processing misleading information which may lead to

unsafe or inefficient actions. In all these instances, 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 interval, 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 backend suffers some drawbacks [6, 14, 15, 17]. Time series come from a continuous nature in which they are recorded at regular intervals, such as hourly or daily, or at irregular intervals, such as recording when a pump is open or closed.

One problem when dealing with time series data results from the fact that these data are often voluminous [7]. As a result, storing and accessing them can be complicated. Moreover, it is specially critical when developing small embedded systems, whose resources (capacity, energy, processing, and communications) suffer a genuine restriction [16]. Another problem is that the procedure of processing and synthesising information becomes complicated if 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 for a multiresolution TSMS (MTSMS). This model allows to store time series using different time resolutions and organised in an aggregated way. The model is specifically designed to cope well with bounded storage computers like those found in sensor systems.

This paper is organised as follows. Following this introduction, we summarise the TSMS and the MTSMS features and we show some related work concerning these systems. Next, we formalise the MTSMS model: in section 2 we introduce nomenclature preliminaries, mainly about measures and time series; in section 3 we define the data structure, the main part of the model; and in section 4 we extend for the summarising operators, which we call attribute aggregate functions. In section 5 we show a multiresolution database example for real data. Finally in section 6 we summarise our MTSMS model proposal and we consider some future working directions.

1.1 TSMS features

A TSMS is a special purpose DBMS aimed at storing and managing 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 relation model [4] is the referent.

In time series analysis there are some common operations that can be generalised when managing time series.

The main attribute of time series is the time, therefore dealing with time is a common operation, such as querying time intervals, finding time correlations, or calculating dis-

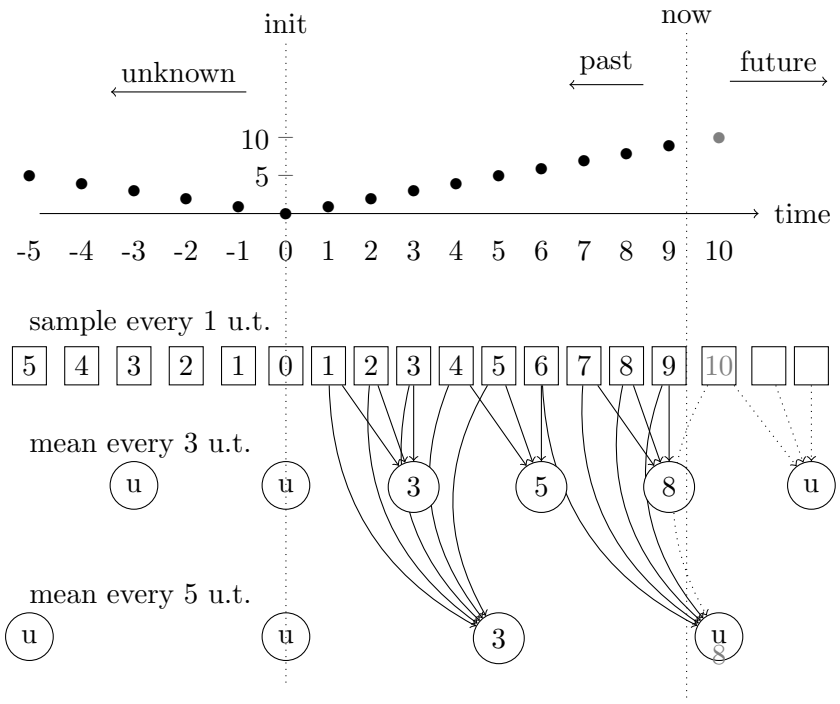

Figure 1: Multiresolution snapshot diagram with regular sampling

tances between two time series. TSMS must respect the temporal coherence of the time series. In the context of statistics, aggregation of time series is also common operation. Aggregation consists in summarising a time series subset with an attribute such as the mean, the maximum, or the mode.

A particular feature of time series is representation. A time series is discrete in the set sense, that is a set of value and time pairs. Representation is the function model approximating the time series to its continuous nature. TSMS operate on time series respecting the representation coherence. Furthermore, the values of a time series can be of any type.

1.2 Multiresolution motivation example

A MTSMS is a TSMS with multiresolution capabilities. In a MTSMS a schema has to be configured, which mainly consist in defining different pairs of a time resolution and an attribute summarising function. We show it with an example.

Figures 1 and 2 show a multiresolution summary for a regular and irregular time series, respectively. Both figures show a snapshot in time, suppose between time 9 and 10.

At the top of the figures there is a plot of a time series with time axis in general units of time (u.t.) and with value axis in undetermined units. The 'now' point shows when the snapshot has been taken, so the time before is the past and the time after is the future, which is grey coloured. The 'init' point shows when the database system has

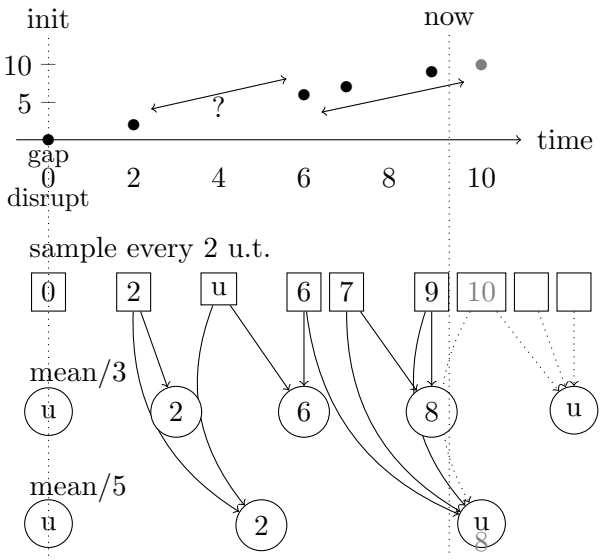

Figure 2: Multiresolution snapshot diagram with irregular sampling

started sampling, so data in time before is unknown; we indicate the starting point as being zero u.t. and unknown time points with negative units.

At the bottom of the figures there is a digram showing the multiresolution action. The first row shows the numerical time series' values corresponding to the above plot; in fig. 1 the time series is sampled every one unit of time and in fig. 2 every two. The second and the third row show a particular schema of a multiresolution database consisting in two time resolutions for the time series: one computes the mean of the sampled values every three u.t. and the other computes the mean every five u.t. In this example, computing the mean acts as the attribute summarising function of which we have spoken earlier. All data stored before zero time, this included, is unknown as sampling had not started. For the future values we also simulate as having unknown stored values which will change as time advances.

If we look at fig. 1 we see, drawn by arrows, that every three sampled values a mean is stored and independently every five values another mean is stored. For the future values we show in gray that if we advance the time one u.t. then value 10 is sampled an the mean for time 10 can be computed resulting 8 but not yet the mean for time 12.

Fig. 2 is essentially the same but showing two possible monitoring irregularities: a gap and a time disruption. In other words, we want to sample the time series every 2 u.t. but first for some reason it can not be done in time 4 and second the sampling clock is disrupted and samples are done in time 7 and 9 instead of 8. The resulting stored time schema is the same: on time resolution every 3 u.t. and the other every 5 u.t.; that is, without time disruptions. The resulting stored values are computed from the known sampled values, some coincide with fig. 1 whereas some differ specially in the gap. A better function than mean would solve this, we extend this further in section 4.

MTSMS improve TSMS features in various aspects:

- Voluminous data. Monitoring systems capture a huge amount of data from sensors. In order to be able to process this information, data volume must be reduced. With the multiresolution approach only the most interesting segments of data are stored. This segments are seen as different resolutions for the same time series and the user configures how they are extracted and summarised by defining different consolidation steps and functions. Multiresolution can also be useful at visualisation time as the user is able to select the best time range and time step that fits into the screen; there is no need to process with more quantity of data than the one that can be shown. In figure 1 there is an example of extracting two resolutions: one every three units of time and another every five.

- Data validation. Monitoring systems capture data but can occur some drawbacks that will affect later the process of time series analysis. Main problems are found when monitors can not capture data, known as gaps, or capture data erroneously, such as outlayers. The multiresolution attribute functions cope well with validating, filtering and calculating with this unknown data in order to keep a consistent historic. In figure 2 an example of a gap can be seen.

- Data time regularising. Another monitoring side effect happens when the sampling rate is not constant, that is when the resulting data is not equi-time spaced. This no regularities can come from sampling jitters in periodic sampling or from no periodic event-based sampling. The multiresolution consolidation regularises the time interval when processes a time series, therefore each resulting time series segment has a regular time resolution. This regularising approach could also be used when the user wants to consult another resolution for a time series, such as changing periodic data from a month to a year step. In figure 2 an example of time regularising can be seen.

- Information summaries. Time series analysis typically focuses on reconstructing the original signal. However, the user objective in a database system is to consult some information. The multiresolution approach is a lossy compression storage solution for data. Therefore it can be regarded as not only approximating to the original time series function but also extracting the interesting information. The selected information must be determined a priori assuming the context where the future queries will be done. In figure 1 there is an example of summarising by mean attribute.

1.3 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 in 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 structures: one for sensor properties stored into 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 sensor data is stored distributed in sensors. Queries are resolved combining sensor data in a data stream orientation, which improves processing performance. However, data streams imposes restriction on operators so this TSMS can not be generalised to other time series types.

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. Although SciDB is based on arrays, in which it includes time series, it does not consider time series special needs, that is not considering how continuously voluminous data or temporal coherence are achieved. In contrast, SciQL shows how some time series properties are achieved by the DBMS such as time series regularities, interpolation or correlation queries. However, difference between tables and arrays seems too physical approached and consequently can collide in representing ambiguously time series.

Bitemporal DBMS is another database field related with time. Bitemporal data is mainly targeted at keeping 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 between time series and bitemporal data that can be considered. First, extending a relational database model to manage bitemporal data shows the way to extend relational DBMS 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.

2 Preliminaries

In this section we clarify the time series' definitions and nomenclature which we use next when defining the MTSMS model. The main objects of a MTSMS are *measures* and *time series*. A *measure* is a value measured in an instant in time and a *time series* is a collection of *measures*.

A *measure* is a tuple (v, t) where v is the value of the measure and $t \in \mathbb{R}$ is the instant in time of measurement. The values of a time series can be of any type; for simplicity examples are presented generally with integers or real numbers but can also be strings or structures such as arrays. 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 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 that are ordered in time.

Definition 1 (Time series) *A* time series S *is a a set of measures* $S = \{m_0, \ldots, m_k\}$ *without repeated time values* $\forall i, j : i \leq k, j \leq k, i \neq j : T(m_i) \neq T(m_j)$*, where* $k+1 = |S|$ *is the cardinality of the set.*

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 similarly as it is done in [10, 8]. Let $S = \{m_0, \ldots, m_k\}$ be a time series. We define the subset $S(r, t] \subseteq S$ as the time series $S(r, t] = \{m \in S | r < T(m) \leq t\}$, 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 $S = \{m_0, \ldots, m_k\}$ be a time series, t an time instant and δ a time duration, the time series' measures can be located in the time interval $i_0 = [t, t + \delta]$ and its multiples $i_j = [t + j\delta\,,\, t + (j + 1)\delta] : j = 0, 1, 2, \ldots$. In signal processing, these time intervals are called sampling intervals, δ is called sampling period and t is called initial time. When the measures are equally spaced the time series is called regular.

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

3 The multiresolution TSMS data model

In this section we design a mathematical model for the multiresolution time series database management systems (MTSMS). Some concepts come from an abstraction of RRDtool operations [13].

A MTSMS manages time series. A time series is regarded as a chronological data collection, so it needs an appropriate management by the DBMS. The MTSMS model is an storage solution for a time series where, in short, the time series information is spread in different time resolutions. The objects of a MTSMS are *measures* and *time series* as defined in section 2 and each database from a MTSMS contains only one time series.

The general schema of the MTSMS model can be seen in figure 3. A *multiresolution database* is a collection of *resolution discs*, which temporarily accumulate the *measures*

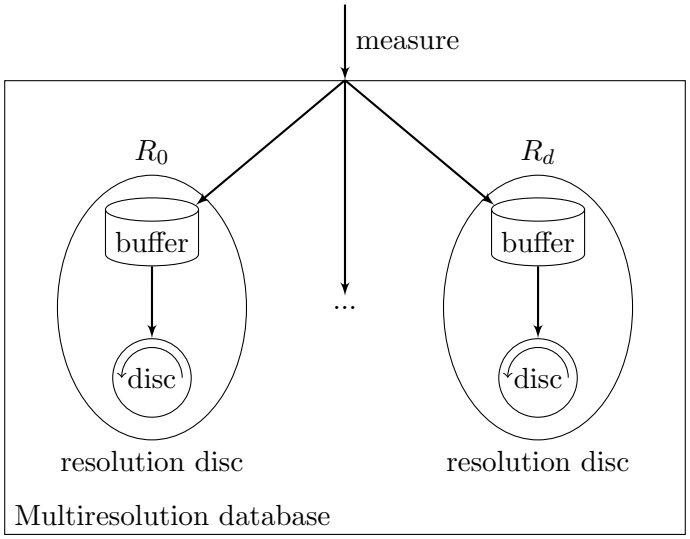

Figure 3: Architecture of MTSMS model

in a *buffer* where they are processed and finally stored in a *disc*. Mainly, the data process is 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, which are spread in the *discs*.

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

Regarding operations, MTSMS model needs operators to change the time intervals between measures. Mainly, these operators are *attribute aggregate* functions and *consolidation* actions. In this section we design the basic MTSMS model, that is four basic data model definitions – *buffer*, *disc*, *resolution disc*, and *multiresolution database* – and 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,so we define them apart in section 4.

3.1 Buffer

A buffer is a container for a regular or a no-regular time series. The buffer objective is to regularise the time series with predetermined δ and attribute function. We call consolidation to this action. In the context of buffers, the sampling period is called consolidation step and the function is called attribute aggregate function.

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

We define the operator *addBuffer* as the addition of a new measure to buffer's time series, addBuffer : $B = (S, \tau, \delta, f) \times m = (v, t) \mapsto B'$ where $B' = (S', \tau, \delta, f)$ and $S' = S \cup \{m\}$.

$$\text{addBuffer} : \text{Buffer} \times \text{Measure} \longrightarrow \text{Buffer}$$

We say the buffer is ready to consolidate when a time of a 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 $T(m) \geq \tau + \delta$.

Let $B = (S, \tau, \delta, f)$ be a buffer ready to consolidate, the consolidation from B in the time interval $i = [\tau, \tau + \delta]$ results in a measure $m' = (v, \tau + \delta)$ where $m' = f(S, i)$. Attribute aggregate function f is described next in section 4. We define the operator *consolidateBuffer* that calculates the measure of consolidation and reduces the consolidated part of the time series from the buffer. In a simplified, the *consolidateBuffer* is only applied to the present consolidation interval, consolidateBuffer : $B = (S, \tau, \delta, f) \mapsto B' \times m'$ where $B' = (S', \tau + \delta, \delta, f)$, $S' = S$ and $m' = f(S, [\tau, \tau + \delta])$. Note that the resulting buffer's time series can be reduced, such as $S' = S(\tau + \delta, \infty)$, when the historic is no more needed.

$$\text{consolidateBuffer} : \text{Buffer} \longrightarrow \text{Buffer} \times \text{Measure}$$

3.2 Disc

A disc is a container with a finite capacity. Each 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.

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 $D_\emptyset = (\emptyset, k)$ is a disc with an empty time series and the k maximum cardinal that S is allowed to take.

The cardinal of the times series is kept under control by the add operator. The operator *addDisc* is defined as the procedure to add a measure to the time series stored on the disc. If the predefined capacity is exceeded, the minimum measure of the time series is discarded, addDisc : $D = (S, k) \times m \mapsto D'$ where $D' = (S', k)$,

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

$$\text{addDisc} : \text{Disc} \times \text{Measure} \longrightarrow \text{Disc}$$

3.3 Resolution disc

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

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 are related to the buffer and disc ones.

Operator *addRD* is 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 $B' = B$ addBuffer m.

$$\text{addRD} : \text{Resolution Disc} \times \text{Measure} \longrightarrow \text{Resolution Disc}$$

Let $R = (B, D)$ be a resolution disc, R is ready to consolidate if and only if B is ready to consolidate.

If the resolution disc is ready to consolidate, it can be consolidated. Operator *consolidateRD* calculates a consolidation measure from the buffer and adds it to the disc, consolidateRD $: R = (B, D) \mapsto R'$ where $R' = (B', D')$, $B' \times m' =$ consolidateBuffer B and $D' = D$ addDisc m'.

$$\text{consolidateRD} : \text{Resolution Disc} \longrightarrow \text{Resolution Disc}$$

3.4 Multiresolution Database

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 has been seen in figure 3.

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 $M_\emptyset = \{R_{0_\emptyset}, \ldots, R_{d_\emptyset}\}$.

Generally, in a multiresolution database there are not two resolution discs with the same information. That is, let $R_a = (B_a, D_a)$ and $R_b = (B_b, D_b)$ be two resolution discs, its buffers $B_a = (S_a, \tau_a, \delta_a, f_a)$ and $B_b = (S_b, \tau_b, \delta_b, f_b)$ have different consolidation interval and attribute aggregate function $\delta_a \neq \delta_b \wedge f_a \neq f_b$.

With reference to the operators, the add and consolidate in a multiresolution database are applied to every resolution disc it contains.

Operator *addMD* is defined as the addition of a measure to every resolution disc, addMD $: M = \{R_0, \ldots, R_d\} \times m \mapsto M'$ where $M' = \{\forall R_i \in M : R_i \text{ addRD } m\}$.

$$\text{addMD} : \text{Multiresolution Database} \times \text{Measure} \longrightarrow \text{Multiresolution database}$$

Operator *consolidateMD* consolidates the resolution discs that are ready to consolidate, consolidateMD $: M = \{R_0, \ldots, R_d\} \mapsto M'$ where $M' = \big\{\forall R_i \in M :$
$\begin{cases} \text{consolidateRD } R_i & \text{if } R_i \text{ ready to consolidate} \\ R_i & \text{else} \end{cases}\big\}$.

$$\text{consolidateMD} : \text{Multiresolution Database} \longrightarrow \text{Multiresolution Database}$$

4 Attribute aggregate function

An attribute aggregate function is used when a buffer is consolidated in order to summarise information from the time series. 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 an attribute of S in the time interval $i = [T_0, T_f]$:

$$f : \text{Time series} \times \text{time interval} \longrightarrow \text{Measure}$$

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

Different types of attribute aggregate functions may be utilised in order to summarise a time series. As instance, we may calculate attribute statistics from a time series such as the maximum value, the average or apply digital signal processing operations as is done in [17]. Furthermore, attribute aggregate functions copes with data validation, which we will discuss latter in this section.

Next, we exemplify some discrete operators. For simplicity, we present examples with real numbers but they could also be integers, strings or more elaborated structures such as arrays. Let $S' = S(T_0, T_f]$:

- 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.

With reference to data validation, attribute aggregate functions can cope with this process. When data has not been captured or has been captured erroneously, it must be treated as unknown data.

- When data has not been captured it is unknown by nature. For example, we try to capture data from a sensor and there is no response.
- When data is erroneously it must be marked as unknown. For example, we capture data from a sensor but it responses in a not reasonable time or we capture data that is clearly outside a reasonable limits.

As a consequence, attribute aggregate functions deals with these two subprocesses: treating unknown data and marking data as unknown. Following with real numbers example, we extend the domain with a value that means 'unknown', let this unknown value be represented by the improper element infinity (∞).

An attribute aggregate functions treating unknown data is a one that can calculate a result when there are unknown values in the original time series, $f^u : S \times i \mapsto m'$ where $\exists m \in S : V(m) = \infty$. Although from a strict point of view operating with unknown

data makes unknown result, aggregate functions are free to calculate whatever is needed such as time series analysis does with data reconstruction.

For example, arithmetic mean^d aggregate function returns $V(m') = \infty$ if $\exists m \in S : V(m) = \infty$. We can define a new mean function, based on the original arithmetic mean^d aggregate, that naively treats unknown values by keeping the known mean; in other words, it ignores unknown values found in the time interval: arithmetic mean^{du}: $S \times i \mapsto m'$ where $m' =$ arithmetic mean$^d(S'', i)$ and $S'' = \{m'' \in S' : V(m'') \neq \infty\}$.

An attribute aggregate functions marking data as unknown is a one that can give unknown value as the resulting measure's value, $f^{mu} : S \times i \mapsto m'$ where $V(m') \in \mathbb{R} \cup \{\infty\}$.

For example, we can define a maximum aggregate, based on the maximum^d aggregate, that returns unknown if there is a measure's value bigger than 2: maximum^{dmu2}: $S \times i \mapsto m'$ where $V(m') = \begin{cases} \infty & \text{if } m'' > 2 \\ m'' & \text{else} \end{cases}$ and $m'' =$ maximum$^d(S, i)$.

Summarising, 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* (PLR). This last is remarked as the most used owing to the most common representation is with linear functions [9].

The variety of time series representations results in a variety of the same attribute aggregate functions. As instance, a maximum attribute aggregate function may give different values if we consider a linear or a constant piecewise representation.

In conclusion, a huge amount of attribute aggregate functions can be defined and no global assumptions can be made. Therefore, the time series representation results in attribute aggregate functions families. MTSMS must give freedom to the users to define their own functions.

5 Example

Next we show an example database for a real time series data. It is situated in a distributed sensor monitoring system where temperature data has been captured from various sensors [1]. In this example we focus on data from one sensor.

Original data is shown in figure 4 for a period of one year and a half, from 29th April 2010 to 18th October 2011. The time series is plotted interpolating linearly its measures. In this plot we can see that there is missing data and some outlying observations, and there are 146709 stored values.

5.1 Schema

In a multiresolution time series database (MTSDB) a time series is stored in different resolution pieces, that is in different 'time subseries'. In this example we store the time series with high resolution at recent times and with low resolution at older times. The

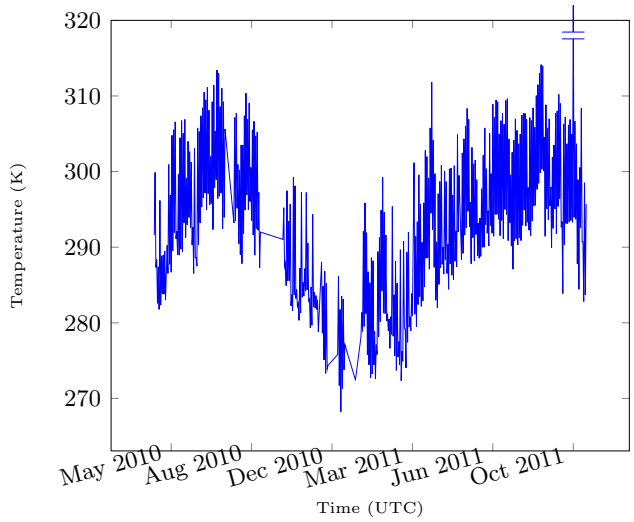

Figure 4: Example of a temperature time series data

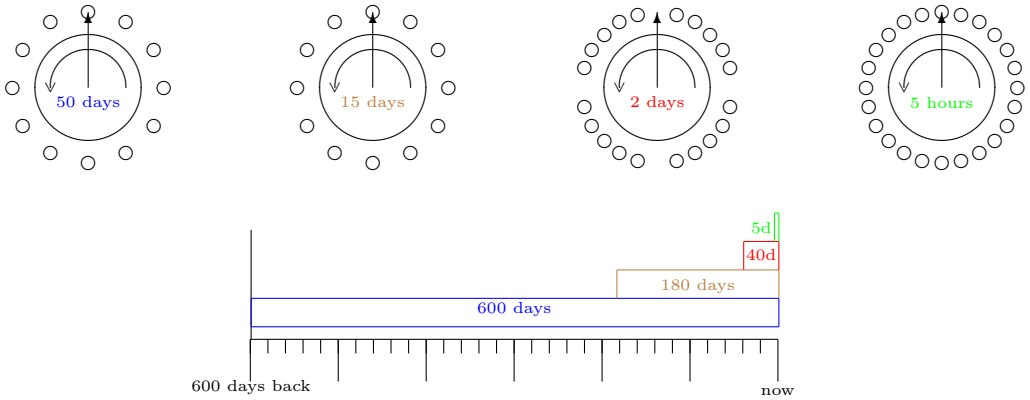

Figure 5: Schema of different resolutions in a MTSDB

schema of this example is illustrated in figure 5. 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. For recent times, every 5 hours a *measure* is stored in the fourth *disc* which has a capacity of 24 *measures* so that results in a 5 day piece. For low mid times, every 2 days a *measure* is stored in the third *disc* which has a capacity of 20 *measures* so that results in a 40 days piece. For high mid times, every 15 days a *measure* is stored in the second *disc* which has a capacity of 12 *measures* so that results in a 180 days piece. For old times, every 50 days a *measure* is stored in the first *disc* which has a capacity of 12 *measures* so that results in a 600 days piece.

5.2 Attribute aggregate functions

In order to illustrate this example we consolidate all the resolution discs by zohe arithmetic mean aggregate function and the highest resolution disc by zohe maximum aggregate function, owing to their simplicity. Next, we show the process in designing both aggregate functions.

In this example 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 called *zero-order hold backwards* (zohe) consisting in holding each value until the preceding value, which a similar representation is used by RRDtool [12].

Let $S = \{m_0, \ldots, m_k\}$ be a time series, we define $S(t)^{\text{zohe}}$ as its *zero-order hold backwards* continuous representation along time t:

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

We now define the *zero-order hold backwards* attribute aggregate function family as the one interpreting the consolidation time interval left-continuous $i = (T_0, T_f]$ and the resulting aggregated measure's time always being T_f, in accordance to the *zero-order hold backwards* representation being defined using left-continuous step functions. Let $S = \{m_0, \ldots, m_k\}$ be a time series and $i = [T_0, T_f]$ be a time interval, the attribute aggregate function $f^{\text{zohe}} \in f$ summarises S with a measure, $f^{zohes} : S = \{m_0, \ldots, m_k\} \times i = [T_0, T_f] \mapsto m'$ where $m' = (v', T_f)$ and the resulting value v' depends on the attribute aggregate function calculated from the subset $S^{zohe} = S(T_0, T_f] \cup \{\min(S - S(-\infty, Tf))\}$.

Let $S' = S^{zohe}$:

- 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 $T(m') = T_f$.

5.3 Results

The resulting time series in the MTSDB are shown in figure 6, where each graphic corresponds to a resolution disc's time series. Each graphic is titled by its resolutions

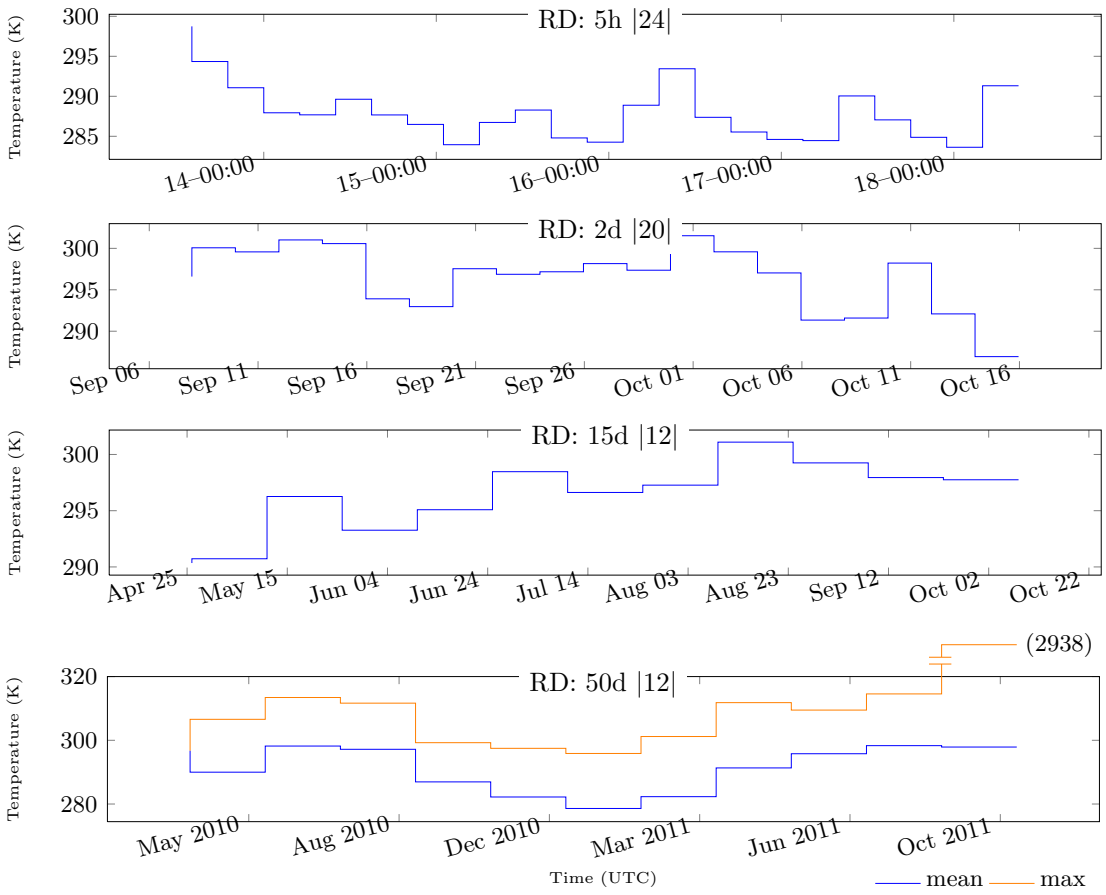

Figure 6: Resolution discs' time series in a MTSDB

disc's delta and cardinal values, and each attribute aggregate function is plotted with different colour. Each time series is plotted with zohe continuous representation. The time axis is shown in UTC units rounded to nearest time points in order to reduce printing space, for example the timestamp December 2010 is a rounding for a day between 1 and 31. The temperature axis is shown in Kelvin units and outlayers are marked as discontinuities, for instance see the 2938 K maximum in the fourth plot.

In all the four plots, we can see that arithmetic mean aggregate function has filled missing data and filtered outlayers observations. This is due to the aggregate function coming from a zohe interpretation, that is known values are time left hold. Otherwise, we could be stricter by using a non filling aggregate such as an unknown marking aggregate function as stated in section 4.

Moreover, in the MTSDB the number of stored values has been reduced to 92, which is the sum of disc's cardinalities. However, now there is only high resolution for recent data. Note that 18th October 2011 corresponds to the 'now ' point showed in figure 5 and May 2010 corresponds to the oldest data known.

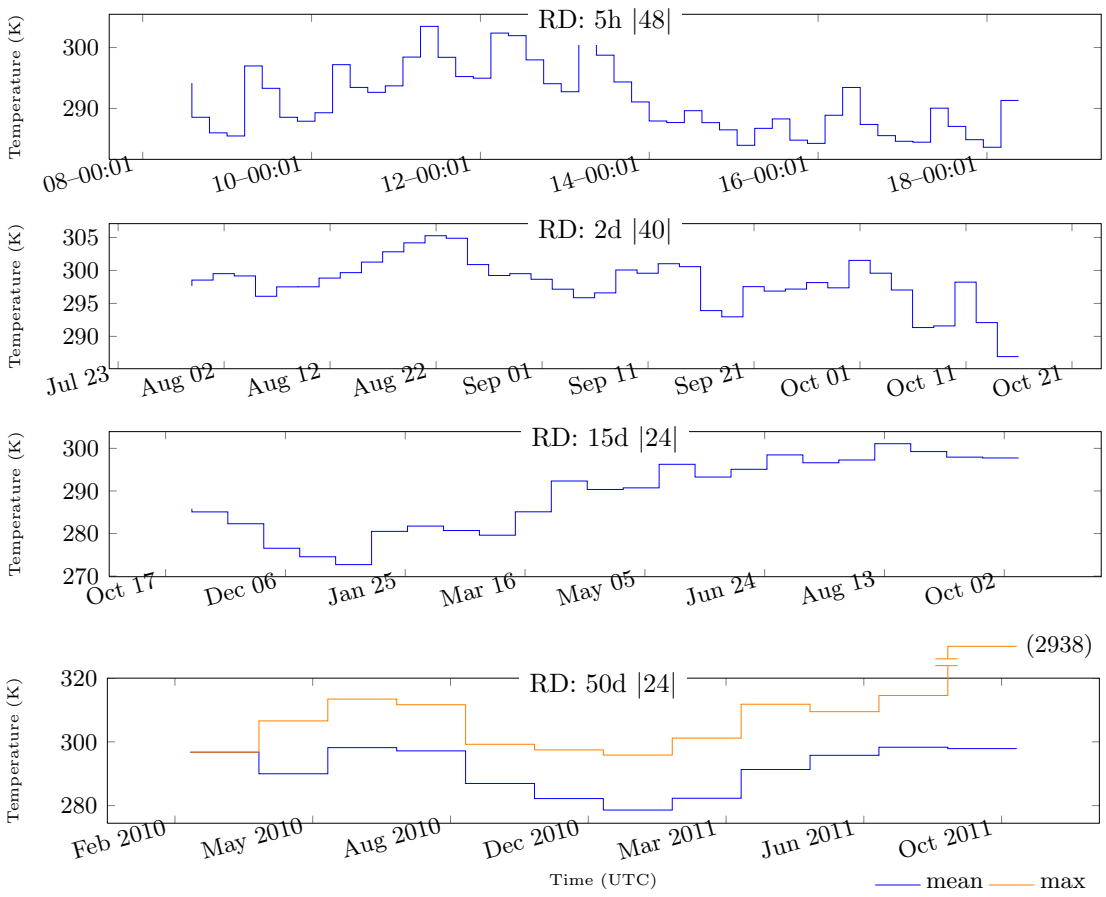

Figure 7: Resolution discs' time series in a bigger MTSDB

We can easily have a bigger MTSDB by doubling the capacity of resolution discs. Then we have 184 stored values and the resulting time series are shown in figure 7. Now we can preserve more historic but no more resolution. Although plots seem smoother, this is only a visualisation issue owing to the fact we are showing more points in the same width. Fourth plot is the same in both figures as there is no more known historic.

With the multiresolution database filled now we may want to ask queries, in which we are currently working in order to develop a model for multiresolution operators. For example, one query could be to unite the resolution discs in order to see the complete graphic. The union of the four time subseries is shown in figure 8. In the intervals where time subseries overlap, the one with the highest resolution is chosen. Therefore we obtain a piecewise function where each piece has a different resolution. 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 aggregation. Comparing this figure with the original figure 4, as we have applied mean aggregation it resembles an incremental low-pass filter and the maximum aggregation resembles an envelope

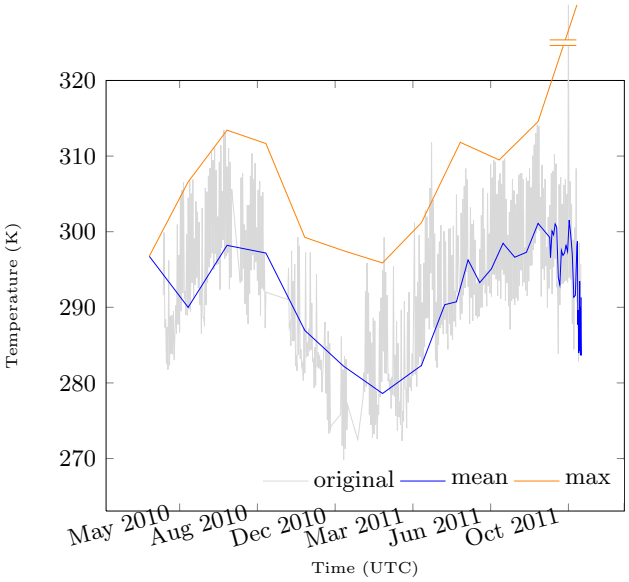

Figure 8: All time series united from the MTSDB

function.

Concluding, we have shown that with these attribute aggregate functions we want not an approximation to the original function but an extraction of some interesting information.

6 Conclusions and future work

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 the three parameters associated with each: 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 but in 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.

In an example we have shown a possible application of a MTSMS. The resulting database has the information we have extracted with the attribute aggregate function. We show that in this example we want not an approximation to the original function but an extraction of some interesting information. Then the database is ready to answer

time series questions keeping in mind that it holds this information summary.

It may not only be interesting to keep more resolution at more recent times. Multiresolution could be selectable, that is high resolution is kept for periods of time considered interesting. For this purpose, the model of multiresolution needs to be changed accordingly. Furthermore, an interesting feature for MTSMS is to be able to configure resolution discs on top of the others, that is setting the one's disc as the buffer for another. The multiresolution model presented shows the simple case where each resolution disc is independent from the others.

Time series have meta information associated [6]: last value measured, value units, classification tags, location, etc. RDBMS are quite good at managing this kind of information. Therefore it would be great if RDBMS and TSMS could interrelate.

Sensor networks is a promising field for TSMS. Sensor networks focus on monitoring great amount of sensors with limited resources which must be used efficiently [16]. One approach is to store data distributed in sensors and resolve each query accordingly [3]. From this point of view, it is useful to treat time series as data streams, which means continuously arriving data in time order, and operate with them incrementally as data arrives. With reference to MTSMS, most recent data could be stored distributed in sensors and least resolution summarised data stored in central nodes. Then queries should be resolved distributed: if adequate resolution is available then it is calculated, else query is sent to sensor. With reference to data streams in MTSMS, it seems that buffers should be reviewed to incorporate streaming capabilities and aggregate functions should be restricted only to those being able to calculate incrementally.

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 time series management in DBMS.

Acknowledgements

The research presented in this paper has been supported by Spanish research project AVIC (TEC2009-09924), SHERECS (DPI2011-26243), the UE project i-Sense (FP7-ICT-270428), and Universitat Politècnica de Catalunya (UPC).

References

[1] Cesare Alippi et al. "An hybrid wireless-wired monitoring system for real-time rock collapse forecasting". In: *7th International Conference on Mobile Adhoc and Sensor Systems*. MASS '10. IEEE, Nov. 2010, pp. 224 –231. DOI: 10.1109/MASS.2010.5663999.

[2] Sabyasachi Basu and Martin Meckesheimer. "Automatic outlier detection for time series: an application to sensor data". In: *Knowledge and Information Systems* 11.2 (2007), pp. 137–154. ISSN: 0219-1377. DOI: 10.1007/s10115-006-0026-6.

[3] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. "Towards Sensor Database Systems". In: *Proceedings of the Second International Conference on Mobile Data Management*. Vol. 1987. MDM '01. Hong Kong: Springer-Verlag, Jan. 2001, pp. 3–14. DOI: 10.1007/3-540-44498-X_1. URL: http://www.cs.cornell.edu/johannes/papers/2001/MDM2001-sensor.pdf (visited on 04/16/2012).

[4] Christopher J. Date. *An Introduction to Database Systems*. 7th ed. Boston, MA, USA: Addisson-Wesley, 2000. ISBN: 0-201-38590-2.

[5] Christopher J. Date, Hugh Darwen, and Nikos A. Lorentzos. *Temporal Data and the Relational Model. A detailed investigation into the application of interval and relation theory to the problem of temporal database management*. San Francisco, CA, USA: Morgan Kaufmann, 2002. ISBN: 1-55860-855-9. DOI: 10.1016/B978-155860855-9/50047-7.

[6] Werner Dreyer, Angelika Kotz Dittrich, and Duri Schmidt. "Research perspectives for time series management systems". In: *SIGMOD Record* 23.1 (Mar. 1994), pp. 10–15. DOI: 10.1145/181550.181553. URL: http://www.ubilab.org/publications/index.html (visited on 11/30/2011).

[7] Tak chung Fu. "A review on time series data mining". In: *Engineering Applications of Artificial Intelligence* 24.1 (Feb. 2011), pp. 164–181. DOI: 10.1016/j.engappai.2010.09.007.

[8] Magnus Lie Hetland. "A survey of recent methods for efficient retrieval of similar time sequences". In: *Data mining in time series databases*. Ed. by Mark Last, Abraham Kandel, and Horst Bunke. Series in Machine Perception and Artificial Intelligence 57. Singapore: World Scientific, 2004. Chap. 2, pp. 23–41.

[9] Eamonn Keogh et al. "Locally adaptive dimensionality reduction for indexing large time series databases". In: *Proceedings of ACM SIGMOD International Conference on Management of Data*. SIGMOD '01. Santa Barbara, California, USA: ACM, May 2001, pp. 151–162. URL: http://www.cs.ucr.edu/~eamonn/sigmod_apca_2001.pdf (visited on 03/15/2011).

[10] Eamonn Keogh et al. "Segmenting time series: a survey and novel approach". In: *Data mining in time series databases*. Ed. by Mark Last, Abraham Kandel, and Horst Bunke. Series in Machine Perception and Artificial Intelligence 57. Singapore: World Scientific, 2004. Chap. 1, pp. 1–21.

[11] Mark Last, Abraham Kandel, and Horst Bunke, eds. *Data mining in time series databases*. Series in Machine Perception and Artificial Intelligence 57. Singapore: World Scientific, 2004.

[12] Tobias Oetiker. "MRTG The Multi Router Traffic Grapher". In: *Proceedings of the 12th Systems Administration Conference*. LISA '98. Boston, Massachusetts: USENIX Association, Dec. 1998, pp. 141–148. URL: http://www.usenix.org/publications/library/proceedings/lisa98/oetiker.html.

[13] Tobias Oetiker. *RRDtool, Round Robin database*. 1998–2011. URL: http://oss.oetiker.ch/rrdtool/ (visited on 10/10/2011).

[14] Duri Schmidt et al. “Time Series, A Neglected Issue in Temporal Database Research?” In: *Proceedings of the International Workshop on Temporal Databases: Recent Advances in Temporal Databases*. Workshops in Computing. Zürich, Switzerland: Springer, Sept. 1995, pp. 214–232. URL: http://www.ecofin.ch/aktuelles/presseartikel/zNeglected_Issue.pdf (visited on 12/01/2011).

[15] Michael Stonebraker et al. “Requirements for Science Data Bases and SciDB”. In: *Fourth Biennial Conference on Innovative Data Systems Research*. CIDR ’09. Asilomar, CA, USA: www.cidrdb.org, Jan. 2009. URL: http://www.scidb.org/Documents/SciDB-CIDR2009.pdf (visited on 04/20/2012).

[16] Yong Yao and Johannes Gehrke. “The Cougar Approach to In-Network Query Processing in Sensor Networks”. In: *SIGMOD Record* 31.3 (Sept. 2002), pp. 9–18. DOI: 10.1145/601858.601861. URL: http://www.cs.cornell.edu/johannes/papers/2002/sigmod-record2002.pdf (visited on 04/16/2012).

[17] Ying Zhang et al. “SciQL: bridging the gap between science and relational DBMS”. In: *Proceedings of the 15th Symposium on International Database Engineering & Applications*. IDEAS ’11. Lisboa, Portugal: ACM, 2011, pp. 124–133. DOI: 10.1145/2076623.2076639. URL: http://www.cwi.nl/~zhang/papers/ideas11.pdf (visited on 04/20/2012).