Temporal Granularity for Unanchored Temporal Data[*]

Iqbal A. Goralwalla[†], Yuri Leontiev, M. Tamer Özsu, Duane Szafron
Laboratory for Database Systems Research
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
{iqbal,yuri,ozsu,duane}@cs.ualberta.ca

and

Carlo Combi (corresponding author)
Laboratory of Artificial Intelligence
Department of Mathematics and Computer Science
University of Udine
via delle Scienze 206, 33100 Udine, Italy
combi@dimi.uniud.it

Abstract

Granularity is an integral feature of both *anchored* (e.g., 25 *October* 1995, *July* 1996) and *unanchored* (e.g., 3 *minutes*, 6 *hours* 20 *minutes*, 5 *days*, 1 *week*) temporal data. In supporting temporal data that is specified in different granularities, numerous approaches have been proposed to deal with the issues of converting temporal data from one granularity to another. The emphasis, however, has only been on granularity conversions with respect to anchored temporal data. This is because a granularity in these approaches is modeled as an *anchored* partitioning of the time axis, thereby making it difficult to deal with granularity conversions in unanchored temporal data. In this paper we provide a novel approach to the treatment of granularity in temporal data. A granularity is modeled as a special kind of unanchored temporal primitive that can be used as a unit of time. That is, a granularity is modeled as a *unit unanchored temporal primitive*. Granularities are accommodated within the context of *calendars* and granularity conversions are presented and discussed in terms of unanchored durations of time. This allows us to consistently model and operate on unanchored temporal data that is comprised of different and mixed granularities. Specifically, we show how unanchored temporal data is represented, give procedures for converting the data to a given granularity, provide canonical forms for the data, and describe how operations between the data are performed.

Keywords: temporal databases, granularity, calendars

[*]This research has been supported by the Natural Sciences and Engineering Research Council of Canada under research grants OGP0951 and OGP8191.

[†]Current address of author is IBM Toronto Lab, IBM Software Solutions Division, 1150 Eglinton Avenue E., North York, ON M3C 1H7, CANADA

1 Introduction

We address the problem of modeling and managing, within a database management system (DBMS), anchored and unanchored temporal data of multiple granularities. Anchored data has a specific location on the time axis, while unanchored data has no specific location. For example, 31 *July* 1995 is an anchored temporal primitive in that we know exactly where it is located on the time axis, whereas 31 *days* is unanchored since we do not know where it is located on the time axis; it can stand for any block of 31 consecutive days. Anchored and unanchored temporal information is usually available in multiple granularities. Such information is prevalent in various fields: *clinical data* [CPP96, CCP97]; *real-time systems* [CMR91]; *geographic information systems* [Flo91]; *office information systems* [BP85, CR88, MPB92]. Clearly, many applications require support for (a) unanchored temporal primitives that are specified in different (for example, 3 *months*, 150 *seconds*) and mixed granularities (for example, 2 *hours and* 20 *minutes*), and (b) anchored temporal primitives that are specified in different granularities (for example, 1990, *October* 1993, 15 *June* 1996).

Supporting anchored and unanchored temporal primitives with different granularities necessitates the proper handling of granularity mismatches in operations between temporal primitives with different granularities. This usually requires converting a temporal primitive from one granularity to another. Although there have been various recent proposals that handle multiple granularities [CC87, WJL91, WJS93, WBBJ97, BP85, MPB92, MMCR92, Sno95], the focus has been on representing anchored temporal primitives that are specified in different granularities. Granularity conversions are given for anchored temporal primitives only. However, we contend that supporting unanchored temporal primitives with different granularities is equally important, and the issues that arise therein must be addressed. In the next section we give a real-world example from clinical medicine that motivates our claim.

1.1 Motivation

We focus on a clinical example related to a patient with cardiological problems, particularly related to the widely known problem of diagnosing and following up unstable angina [BMJ94]. Unstable angina[1] is a transitory clinical syndrome usually associated with an increased duration and/or

[1]Formally known as angina pectoris. A clinical syndrome typically characterized by a deep, poorly localized chest or arm discomfort that is reproducibly associated with physical exertion or emotional stress and relieved promptly by rest or sublingual nitroglycerine. The discomfort of angina is often hard for patients to describe, and many patients do not consider it to be "pain." [BMJ94].

intensity of symptoms related to coronary artery disease; risk of cardiac death and myocardial infarction increase. In this situation it is important to consider both the time when the symptoms (like chest pain) began and the time duration of these symptoms.

Let us consider the following sentences which are related to information contained in the cardiological medical record of a patient:

$S1$. The patient suffered from chest pain at rest for 2 hours and 55 minutes on 13 December 1995.
$S2$. The patient has been admitted to an Intensive Care Unit from 21:00 29 January 1996, and he has undergone intensive medical management for 36 hours.
$S3$. At 3 pm 12 April 1996 the patient presented a new episode of chest pain of 7 minutes and 35 seconds during a soft exertion.
$S4$. In 1997 the patient had to take a thrombolytic therapy for 7 months.
$S5$. In January 1998 the patient had to take another thrombolytic therapy for 15 days.

We can observe from the above sentences that there are many different granularities for time instants (days in $S1$; minutes in $S2$; hours in $S3$; months in $S5$; years in $S4$) and different and mixed granularities for time spans (hours and minutes in $S1$; hours in $S2$; minutes and seconds in 3; months in $S4$; days in $S5$). Moreover, in a single sentence there may be time instants and time spans having heterogeneous granularities. For example, in $S2$ the time instant at which the patient is admitted is specified at the granularity of minutes, while the time duration of the medical management that the patient undertook is specified at the granularity of hours.

In addition to the different and mixed granularities in the patient-related information, the definition of unstable angina itself involves time spans given at different granularities. Unstable angina is, in fact, defined as: (1) symptoms of angina at rest, for more than 20 minutes, or (2) new onset, within two months, of exertional angina, involving marked limitations of ordinary physical activity, or (3) increasing angina within two months from the initial presentation, or (4) post-myocardial infarction angina, i.e., angina occurring from 1 to 60 days after an acute myocardial infarction [BMJ94].

In a clinical setting, we need to be able to derive some extra information from the stored sentences about the patient. For example:

1. What is the global span of the symptoms of angina?

   To answer this question, the time spans 7 *minutes and* 35 *seconds*, and 2 *hours and* 55 *minutes* (see sentences $S1$ and $S3$) have to be added.

2. What is the global duration of the thrombolytic therapy?

   In this case we have to add the time span 7 *months* to the time span 15 *days* (see sentences $S4$ and $S5$).

These questions substantiate the need for a temporal DBMS to provide the means for (a) representing and storing time instants with different granularities, and time spans with different and mixed granularities, (b) handling granularity mismatches in operations between temporal primitives with different granularities, and (c) converting a temporal primitive from one granularity to another. In the rest of the paper, we show how these issues can be supported in a temporal DBMS, focusing on unanchored temporal primitives. Details on anchored primitives can be found in [Gor98].

1.2 Background

In this work, we model a granularity as a *unit unanchored temporal primitive* (*unit time span*). More specifically, a granularity is modeled as a special kind of time span that can be used as a unit of time. Granularity conversions are presented and discussed in terms of unanchored durations of time. To the best of our knowledge, this feature is novel to our work. It allows us to consistently model and operate on unanchored temporal data that is comprised of different and mixed granularities. Our work should be seen as complementing other works on temporal granularity. It fills in the missing piece by allowing unanchored temporal primitives to be specified in different and mixed granularities, and facilitates the conversion of unanchored temporal primitives from one granularity to another.

The inherent problem in adequately supporting unanchored temporal primitives with different granularities in [CC87, WJL91, WJS93, WBBJ97, BP85, MPB92, MMCR92, Sno95], is that a granularity is treated as an *anchored* partitioning of the time axis. Since unanchored temporal primitives are *independent* of anchored temporal primitives (i.e., their location on the time axis is unknown since they are not anchored at any particular point) problems arise in the conversion of unanchored temporal primitives from one granularity to another when a granularity is modeled as an anchored partitioning of the time axis. In converting (scaling) an unanchored temporal primitive from one granularity to another in TSQL2 [Sno95], it is noted:

> "... the problem is that a granularity is an anchored partitioning, whereas an interval[2] is unanchored ... the consequence of the unanchored nature of intervals is that whenever

[2]An *interval* is the basic unanchored temporal primitive in TSQL2. It is similar to a *time span* in our work.

an interval is scaled, an indeterminate interval will result, even when an interval is scaled from a finer to a coarser granularity" (page 370 in [Sno95]).

1.3 Paper Organization

The rest of the paper is organized as follows: in Section 2 we describe how multiple granularities are accommodated within the context of calendars and the derivation procedures of converting one granularity to another. In Section 3 we present our model for unanchored temporal primitives. Specifically, we show how these primitives are represented, give procedures for converting a primitive to a given granularity, provide canonical forms for the primitives, and provide a description of how operations between the primitives are performed. Section 4 sheds some light on implementation issues. Finally, Section 5 presents conclusions and outlines future avenues of research.

2 Calendars

2.1 Modeling Issues

A calendar is a means by which physical time can be represented so as to be human readable. It is comprised of time units of varying granularities that enable the representation of different temporal primitives. Common calendars include *Gregorian* and *Lunar*. Educational institutions also use *Academic* calendars. In many applications, it is desirable to have multiple calendars that have different calendric granularities. In this paper, we base our work on a single calendar and refer the reader to [GLÖS96, Gor98] for details on multiple calendar support. There are a number of important issues that we address in our model:

1. How are calendars modeled? What are their components? Does a calendar provide relationships between granularities?

2. How is anchored and unanchored temporal information modeled within the context of calendars? Can anchored temporal primitives be of different granularities? How about unanchored temporal primitives?

3. Can anchored time be converted from one granularity to another? How about unanchored time?

4. What are the semantics of operations between anchored times, unanchored times, and mixed anchored and unanchored times which have operands of different granularities?

We start with a definition of a calendar in our model, followed by details on granularities and the conversion process that we propose.

Definition 2.1 *Calendar ($\mathcal{C}$):* A calendar $\mathcal{C}$ is a triplet $\langle O, \mathcal{G}, \mathcal{F} \rangle$, where O is the origin of $\mathcal{C}$, $\mathcal{G}$ is the set of calendric granularities belonging to $\mathcal{C}$, and $\mathcal{F}$ is a list of conversion functions associated with $\mathcal{C}$. ■

The origin marks the start of a calendar. Calendric granularities define the reasonable time units (e.g., *minute*, *day*, *month*) that can be used in conjunction with it. Calendric granularities within a calendar are counted from its origin. The functions establish the conversion rules between the different granularities of a calendar.

2.2 Calendric Granularities

A calendar is comprised of a finite number of time units. For example, the Gregorian calendar includes days and months as time units; the Academic calendar adds semesters. We call these time units *calendric granularities.* In the Gregorian calendar the set of calendric granularities consists of *year*, *month*, *day*, *hour*, *minute*, and *second.* Generally speaking, a calendric granularity is a unit of measurement for time durations. For example, the calendric granularity of days (G_{day}) in the Gregorian calendar behaves similar to the unanchored duration 1 *day* (unanchored durations are discussed in more detail in Section 3.1).

Definition 2.2 *Calendric granularity (G):* A calendric granularity is a special kind of unanchored duration that can be used as a unit of time. ■

Since a calendric granularity is a special kind of a time span, it is meaningful to compare two calendric granularities with each other.

Definition 2.3 *Comparison between calendric granularities:* G_A is *coarser* than G_B if $G_A > G_B$ as a time span. Similarly, G_A is *finer* than G_B if $G_A < G_B$ as a time span. ■

Example 2.1 The span of 1 *day* is shorter ($<$) than the span of 1 *month* and therefore the calendric granularity of days (G_{day}) is finer than the calendric granularity of months (G_{month}) in the Gregorian calendar. Similarly, G_{month} is coarser than G_{day}. □

We must always be able to compare two calendric granularities with each other. Thus,

Condition 2.1 The set of all possible calendric granularities is totally ordered with respect to the comparison operators given in Definition 2.3. □

Condition 2.1 provides us with a sharp contrast from other work dealing with temporal granularity. While granularities are totally ordered in our framework, in many others they are typically only partially ordered [Sno95, WBBJ97]. This allows us to carry out granularity conversions in terms of time spans. In this paper we do not consider granularities with gaps (as *business week* or *business month*), as in [WBBJ97, BWJ96].

Each calendric granularity has a list of *calendric elements.* For example in the Gregorian calendar, G_{day} has the calendric elements $1, 2, \ldots, 7$. Similarly in the Academic calendar, $G_{semester}$ has the calendric elements *Fall*, *Winter*, *Spring*, and *Summer*.

2.3 Functions

Associated with a calendar is a list of functions ($\mathcal{F}$) which determine the number of finer calendric elements in coarser calendric elements. For example, assume we have a calendar C with the calendric granularities *year*, *month* and *day.* Three functions are defined: The first returns the number of months in a given year; the second returns the number of days in a given month of a given year; and the third maps a given year, month, and day to a real number on a global timeline. Notice that these functions depend on the particular value of a granularity and not just the granularity itself. For example, the number of days in a month depend on the month itself. More generally, let C be a calendar with calendric granularities $G_1, G_2, \ldots, G_n$, where G_1 is the coarsest calendric granularity and G_n is the finest calendric granularity. The following functions are then defined:

Definition 2.4 *Conversion functions:*

$$\begin{array}{rcl} f_C^1(i_1) & \rightarrow & N_{G_2},\ 1 \leq i_1 \leq p_1 \\ f_C^2(i_1, i_2) & \rightarrow & N_{G_3},\ 1 \leq i_1 \leq p_1, 1 \leq i_2 \leq p_2 \\ & \vdots & \\ f_C^n(i_1, i_2, \ldots, i_n) & \rightarrow & R,\ 1 \leq i_1 \leq p_1, 1 \leq i_2 \leq p_2, \ldots, 1 \leq i_n \leq p_n \end{array}$$

where i_j $(1 \leq j \leq n)$ are natural numbers which correspond to the ordinal number of a calendric element of the j^{th} calendric granularity in calendar C. For example, the ordinal values of the year 1995 and the month September in the Gregorian calendar would be 1995 and 9, respectively. N_{G_x} $(1 \leq x \leq n)$ is a natural number which stands for the number of G_x's. p_i is the range of calendric elements for each considered granularity. R is a real number[3]. ■

[3]We assume the underlying global timeline is real.

The first function ($f_C^1(i_1)$) gives the number of G_2's in a given calendric element of G_1. The second function ($f_C^2(i_1, i_2)$) gives the number of G_3's defined by a given pair of calendric elements of types G_1 and G_2. The last function ($f_C^n(i_1, i_2, \ldots, i_n)$) maps a calendric element of the finest calendric granularity (G_n) to a real number on an underlying global real timeline, hereafter referred to as G_{tl}. The scale of G_{tl} is dependent on the precision of the respective machine architecture. For simplicity and explanatory purposes in this paper, we assume the scale of G_{tl} to be *seconds*.

Example 2.2 To illustrate the workings of the above functions, let us suppose we are interested in the number of months in 1995, the number of days in September 1995 and the number of seconds in 12 September 1995 in calendar C. The ordinal values corresponding to the year 1995, the month September, and the day 12, are 1995, 9, and 12, respectively. Then:

$$\begin{aligned} f_C^1(1995) &\rightarrow 12 \\ f_C^2(1995, 9) &\rightarrow 30 \\ f_C^3(1995, 9, 12) &\rightarrow 86400.0 \end{aligned}$$

□

Although the above example is trivial, it illustrates how the conversion functions work. It sets the stage for the more complicated cases that are discussed in the next section.

2.4 Conversions between Calendric Granularities

In a temporal model where times with different calendric granularities are supported, we need to be able to convert a finer calendric granularity to a coarser calendric granularity, and vice-versa. We discuss these conversions below by first defining two functions. These functions are necessary since the number of units of one granularity contained in a unit of another granularity is not fixed.

Definition 2.5 *Lower bound factor* [$lbf(G_A, G_B)$]: The lower bound factor of G_A and G_B is the minimum number of G_B units that can form 1 G_A unit. ■

Definition 2.6 *Upper bound factor* [$ubf(G_A, G_B)$]: The upper bound factor of G_A and G_B is the maximum number of G_B units that can form 1 G_A unit. ■

Example 2.3 $lbf(G_{month}, G_{day}) = 28$ and $ubf(G_{month}, G_{day}) = 31$. Both factors coincide in the case of those granularities that have exact conversions. For instance, $lbf(G_{hour}, G_{minute}) = ubf(G_{hour}, G_{minute}) = 60$. □

The user can define new calendric granularities in terms of existing ones. For example, the new calendric granularity *decade* could be defined in terms of the existing calendric granularity *year* using $lbf(G_{decade}, G_{year}) = ubf(G_{decade}, G_{year}) = 10$.

We now show how $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ are derived from the conversion functions defined in Section 2.3 when G_A is coarser than G_B, and when G_A is finer than G_B.

Derivation 2.1 *G_A is coarser than G_B:* Let $G_1, \ldots, G_A, \ldots, G_B, \ldots, G_n$ be the totally ordered calendric granularities of calendar C with G_1 being the coarsest calendric granularity and G_n the finest. The following conversion functions are defined in C:

$$\begin{array}{rcl} & \vdots & \\ f_C^A(i_1, \ldots, i_A) & \rightarrow & N_{G_A+1} \\ & \vdots & \\ f_C^B(i_1, \ldots, i_B) & \rightarrow & N_{G_B+1} \\ & \vdots & \end{array}$$

Now, the number of G_B units in any given calendric element i_A is given by the following summation:

$$f_C^{A\rightarrow B}(i_1, \ldots, i_A) = \sum_{j_1=1}^{f_C^A(i_1,\ldots,i_A)} \sum_{j_2=1}^{f_C^{A+1}(i_1,\ldots,i_A,j_1)} \cdots \sum_{j_{B-A-1}=1}^{f_C^{B-2}(i_1,\ldots,i_A,j_1,\ldots,j_{B-A-2})} f_C^{B-1}(i_1, \ldots, i_A, j_1, \ldots, j_{B-A-1})$$

The minimum (maximum) number of G_B units in calendric element i_A is then the minimum (maximum) of the above formula over all $i_1, \ldots, i_A$. More specifically,

$$lbf(G_A, G_B) = \min_{(i_1,\ldots,i_A)\in C} \{f_C^{A\rightarrow B}(i_1, \ldots, i_A)\} \qquad (1)$$

$$ubf(G_A, G_B) = \max_{(i_1,\ldots,i_A)\in C} \{f_C^{A\rightarrow B}(i_1, \ldots, i_A)\} \qquad (2)$$

∎

Example 2.4 Let C be a calendar with the calendric granularities *year*, *month* and *day*. The following functions are defined in C:

$$\begin{array}{rcl} f_C^1(y) & \rightarrow & N_{months} \\ f_C^2(y, m) & \rightarrow & N_{days} \\ f_C^3(y, m, d) & \rightarrow & R \end{array}$$

where y, m, and d are ordinal values of calendric elements in the calendric granularities year, month, and day, respectively. Suppose we want to find $lbf(G_{year}, G_{day})$ and $ubf(G_{year}, G_{day})$. The number of days in any year y is given by the summation: $\sum_{m=1}^{f_C^1(y)} f_C^2(y, m)$. The minimum (maximum) number of days in a year is then the minimum (maximum) of this summation over all y. More specifically,

$$lbf(G_{year}, G_{day}) = \min_{y}\{\sum_{m=1}^{f_C^1(y)} f_C^2(y,m)\} \qquad ubf(G_{year}, G_{day}) = \max_{y}\{\sum_{m=1}^{f_C^1(y)} f_C^2(y,m)\}$$

□

Derivation 2.2 *Minimum and maximum number of G_B in K units of G_A:* Formulas (1) and (2) calculate the minimum and maximum number of G_B in *one* unit of G_A, respectively. We now generalize formulas (1) and (2) to calculate the minimum and maximum number of G_B in K units of G_A, e.g., the minimum and maximum number of days in $2 \cdot G_{month}$ where $K = 2$.

$$lbf(K, G_A, G_B) = \min_{i_1,\ldots,i_A}\{\sum_{0 \le dist_{k_A}((i_1^{'},\ldots,i_A^{'}),(i_1,\ldots,i_A)) \le K-1} f_C^{A\to B}(i_1^{'},\ldots,i_A^{'})\} \quad (3)$$

$$ubf(K, G_A, G_B) = \max_{i_1,\ldots,i_A}\{\sum_{0 \le dist_{k_A}((i_1^{'},\ldots,i_A^{'}),(i_1,\ldots,i_A)) \le K-1} f_C^{A\to B}(i_1^{'},\ldots,i_A^{'})\} \quad (4)$$

The summation in formulas (3) and (4) is the number of B units in K consecutive A units starting with $(i_1,\ldots,i_A)$. The function $dist_{k_A}((i_1^{'},\ldots,i_A^{'}),(i_1,\ldots,i_A))$ finds the number of k_A units elapsed between $(i_1^{'},\ldots,i_A^{'})$ and $(i_1,\ldots,i_A)$. For example, the number of months elapsed between (1996, February) and (1995, January) is 13. The lower and upper bound factors are then obtained by taking the minimum and maximum of the summation over all $(i_1,\ldots,i_A)$. ■

Embedding the coefficient K within formulas (3) and (4) reduces the information lost in the process of calculating the number of G_B units in K units of G_A as compared to first finding the number of G_B units in one unit of G_A and then multiplying it by K to find the number of G_B in K units of G_A. For example, using formulas (1) and (2) to calculate the minimum and maximum number of days in $2 \cdot G_{month}$ gives us 56 and 62, respectively, while formulas (3) and (4) give us 59 and 62, respectively – thereby reducing the information lost by 3 days. Note that for exact conversions, $lbf(K, G_A, G_B) = ubf(K, G_A, G_B) = K \cdot lbf(G_A, G_B) = K \cdot ubf(G_A, G_B)$. For example, $lbf(K, G_{days}, G_{hours}) = ubf(K, G_{days}, G_{hours}) = K \cdot 24$.

Derivation 2.3 *G_A is finer than G_B:* If G_A is finer than G_B, then the lower and upper bound factors can be calculated using the formulas:

$$lbf(N, G_A, G_B) = \max_{K \in \mathbf{Z}}\{K \mid N \ge ubf(K, G_B, G_A)\} \quad (5)$$

$$ubf(N, G_A, G_B) = \min_{K \in \mathbf{Z}}\{K \mid N \le lbf(K, G_B, G_A)\} \quad (6)$$

■

Example 2.5 To illustrate the formulas in Derivation 2.3, suppose we want to find the number of months in 45 *days*. Then:

$$\begin{aligned} lbf(45, G_{day}, G_{month}) &= \max_{K \in \mathbf{Z}}\{K \mid 45 \geq ubf(K, G_{month}, G_{day})\} = 1 \\ ubf(45, G_{day}, G_{month}) &= \min_{K \in \mathbf{Z}}\{K \mid 45 \leq lbf(K, G_{month}, G_{day})\} = 2 \end{aligned}$$

Hence, the number of months in 45 *days* is 1 $\sim$ 2. □

Note that it is not necessary that K be an integer. It can be a real number as well, in which case we reduce the amount of indeterminacy in finding the number of months in 45 *days*.

In TSQL2 [Sno95], a calendar has a specification file which provides regular and irregular mappings between granularities. It is not clear however, how these mappings are derived. In this section, we gave detailed derivation procedures for the $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ functions which represent regular and irregular mappings between any two granularities in a calendar. In Section 3.2, we will see how the $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ functions are used in the conversion of unanchored temporal primitives to a given calendric granularity.

3 Unanchored Temporal Primitives

We identify a *time span* as being an unanchored, relative duration of time. Examples of time spans include 5 hours, 10 days, 2 to 3 months, etc. A time span is basically an atomic, cardinal quantity, independent of any time instant or time interval, with a number of operations defined on it. These operations include comparison with another time span with the transitive comparison operators $<$ and $>$ (which forms a partial order between time spans) and subtraction or addition of another time span to return a third time span.

Time spans can be further characterized as being determinate or indeterminate. A *determinate span* represents complete information about a duration of time. For example, the maximum time allowed for students to complete an examination is a determinate span. An *indeterminate span* represents incomplete information about a duration of time. It has lower and upper bounds that are determinate spans. 1 *day* $\sim$ 2 *days*, for example, is an indeterminate span that can be interpreted as "a time period between one and two days." Any determinate span can be represented as a special kind of indeterminate span with identical lower and upper bounds.

3.1 Representation of Time Spans

Since a calendric granularity is a unit time span, we can use calendric granularities to construct time spans. For example, the time span of 36 *hours* which represents the duration of intensive medical management the patient underwent (see sentence $S3$ in Section 1.1), is obtained as $36 \cdot G_{hour}$. A time span of 2 *hours and* 55 *minutes*, which represents the duration of chest pain the patient suffered (see sentence $S1$ in Section 1.1), can be obtained as $2 \cdot G_{hour} + 55 \cdot G_{minute}$. In general, a time span is made up of mixed calendric granularities and is defined as a finite sum:

Definition 3.1 *Discrete Determinate span:*

$$S_{discr} = \sum_{i=1}^{N} (K_i \cdot G_i) \tag{7}$$

where K_i is an integer coefficient of G_i, which is a distinct calendric granularity in the calendar. ■

In a temporal model where time spans with different calendric granularities are supported, we need to be able to convert a time span to a given calendric granularity. This conversion process, together with the semantics of operations on time spans with mixed granularities, are discussed in the following sections.

3.2 Conversion of Time Spans

The first question is whether it is always possible to convert a time span from a coarser to a finer calendric granularity without loss of information. The answer, perhaps surprisingly, is negative. To illustrate this point, consider the following: the conversion of the time span 1 *hour* to the calendric granularity of minutes is exact and will result in the time span of 60 *minutes*. However, the conversion of the time span 1 *month* to the finer calendric granularity of days cannot possibly be an exact one. Should the resulting time span be 31, 30, 29 or 28 days? We cannot tell unless we know which month is involved. Since a time span is *unanchored* this information is not available. We could convert 1 *month* to the indeterminate span 28 *days* $\sim$ 31 *days* but in this case the conversion is not exact and some information is lost. Therefore, the following observation is made:

Observation 3.1 The set of all calendric granularities is not totally ordered with respect to the binary relation "exactly convertible to." □

We now define the conversion of a determinate time span to any given calendric granularity G_A.

Definition 3.2 *Discrete time span conversion:* The conversion of a time span of the form depicted in Definition 3.1 to a calendric granularity G_A results in a time span[4] with lower bound

$$\lfloor \sum_{i=1}^{N} L_i \rfloor \cdot G_A \tag{8}$$

and upper bound

$$\lceil \sum_{i=1}^{N} U_i \rceil \cdot G_A \tag{9}$$

where

$$L_i = lbf(K_i, G_i, G_A) \text{ and } U_i = ubf(K_i, G_i, G_A) \tag{10}$$

■

In the following example we focus on the more complex and interesting case of an inexact time span conversion.

Example 3.1 Let us convert the discrete time span 2 *months and* 45 *hours* to a time span in the calendric granularity of days (G_{day}). First we represent the given span in the form given in Definition 3.1: $2 \cdot G_{month} + 45 \cdot G_{hour}$. In this span, $K_1 = 2, K_2 = 45, G_1 = G_{month}, G_2 = G_{hour}$. We now use formula (10) to compute L_1, L_2, U_1, U_2:

$$\begin{aligned} L_1 &= lbf(K_1, G_1, G_{day}) \\ &= lbf(2, G_{month}, G_{day}) \\ &= 59 \end{aligned} \qquad \begin{aligned} U_1 &= ubf(K_1, G_1, G_{day}) \\ &= ubf(2, G_{month}, G_{day}) \\ &= 62 \end{aligned}$$

$$\begin{aligned} L_2 &= lbf(K_2, G_2, G_{day}) \\ &= lbf(45, G_{hour}, G_{day}) \\ &= max\{K \mid 45 \geq ubf(K, G_{day}, G_{hour})\} \\ &= max\{K \mid 45 \geq K \cdot 24\} \\ &= 1.875 \end{aligned} \qquad \begin{aligned} U_2 &= ubf(K_2, G_2, G_{day}) \\ &= ubf(45, G_{hour}, G_{day}) \\ &= min\{K \mid 45 \leq lbf(K, G_{day}, G_{hour})\} \\ &= min\{K \mid 45 \leq K \cdot 24\} \\ &= 1.875 \end{aligned}$$

$lbf(K, G_{month}, G_{day}), lbf(K, G_{day}, G_{hour})$, $ubf(K, G_{month}, G_{day})$, and $ubf(K, G_{day}, G_{hour})$ are calculated from the conversion functions in the Gregorian calendar. Lastly, we compute the lower and upper boundary of the resulting time span according to formulas (8) and (9), respectively:

[4]Note that conversion of a time span to any calendric granularity may be exact or inexact. In the former case, the lower and upper bounds of the resulting time span are identical, which signifies that the time span is determinate. In case of an inexact conversion, the resulting time span will be indeterminate. Details on temporal indeterminacy are given in [Gor98].

$$\begin{aligned} \textit{lower bound} &= \lfloor L_1 + L_2 \rfloor \cdot G_{day} & \textit{upper bound} &= \lceil U_1 + U_2 \rceil \cdot G_{day} \\ &= \lfloor 59 + 1.875 \rfloor \cdot G_{day} & &= \lceil 62 + 1.875 \rceil \cdot G_{day} \\ &= 60 \cdot G_{day} & &= 64 \cdot G_{day} \end{aligned}$$

Hence, the result of our conversion is the indeterminate discrete time span 60 *days* $\sim$ 64 *days.*

□

3.3 Canonical Forms for Time Spans

In addition to the set of granularities $G_1, \ldots, G_N$ and conversion functions discussed earlier, each calendar also implicitly defines the relation *exactly convertible to* between its granularities. We say that G_i is *exactly convertible to* G_j iff $ubf(k, G_i, G_j) = lbf(k, G_i, G_j) = k \cdot C$, where C is a natural number. Note that exact convertibility is a partial order on granularities which is a suborder of magnitude ordering. If G_i is exactly convertible to G_j, then $G_i = C \cdot G_j$, where C is a natural number. Since discrete determinate time spans have the form $S = \sum_{i=1}^{N} K_i G_i$, where K_i are integer numbers, the presence of the exact conversion rules implies the existence of different forms of a time span. For example, 2 *hour* 55 *minutes* and 175 *minutes* are different forms of the same time span S'. To adhere as much as possible to human readability and user intuition, it is usually desirable to represent time spans in some canonical form. For example, when the time span 1 *hour* 30 *minutes* is added to the time span 35 *minutes*, the user would expect the time span 2 *hours* 05 *minutes* rather than the time span 1 *hour* 65 *minutes*. In this section, we define canonical forms for time spans. We begin by defining *representations* for time spans.

Definition 3.3 *Span Representation:* The N-tuple $r = \langle a_i \rangle_{i=1}^{N}$ (where a_i are integer numbers and N is the number of calendric granularities in a calendar) is called a *representation* of a span S (denoted $r \in Rep(S)$) iff $S = \sum_{i=1}^{N} a_i G_i$. ■

Example 3.2 Let's assume that the Gregorian calendar has the calendric granularities *year*, *month*, *day*, *hour*, *minute* and *second.* Then 2 *hour* 55 *minutes* and 175 *minutes* which are two forms of S' have the representations $r_1 = \langle 0, 0, 0, 2, 55, 0 \rangle$ and $r_2 = \langle 0, 0, 0, 0, 175, 0 \rangle$, respectively.

□

We will use span representations to define a *canonical form* for a time span. In order to do that, we introduce the notion of a *strictly non-negative span.*

Definition 3.4 *Strictly Non-Negative Span:* A span S is a *strictly non-negative span* (denoted $S >^{+} 0$) iff $\exists r = \langle a_i \rangle_{i=1}^{N} \in Rep(S) : a_i \geq 0$ for $i = 1, \ldots, N$. ■

Example 3.3 The time span 2 *hour* 55 *minutes* is strictly non-negative while the time spans 1 *week* $-$ 10 *days* and 1 *month* $-$ 30 *days* are not strictly non-negative since no positive representations of either of them exist. In the first time span, although we can convert 1 *week* to *days* exactly, the resulting span -3 *days* does not have a positive representation. In the second time span, no positive representations are possible since 1 *month* does not have an exact conversion to *days*. $\square$

Another definition that we need to define for a canonical form is a dominancy relation between span representations. The dominancy relation is in fact a lexicographical order on span representations, which is used in determining the canonical representation of a span..

Definition 3.5 *Dominancy:* A representation $r = \langle a_i \rangle_{i=1}^{N}$ *dominates* another representation $r' = \langle b_i \rangle_{i=1}^{N}$ (denoted $r \succ r'$), $r, r' \in Rep(S)$, iff $\exists k : a_k > b_k \wedge a_i = b_i$ for $i = 1, \ldots, (k-1)$. $\blacksquare$

Example 3.4 $r_1 = \langle 0,0,0,2,55,0 \rangle \succ r_2 = \langle 0,0,0,0,175,0 \rangle$. $\square$

Having defined strictly non-negative spans and dominancy, we can now proceed to define the *canonical representation* and the *canonical form* for strictly non-negative spans[5].

Definition 3.6 *Canonical Representation:* A representation $r = \langle a_i \rangle_{i=1}^{N} \in Rep(S)$ is *the canonical representation* of span $S >^{+} 0$ iff $a_i \geq 0$ for $i = 1, \ldots, N \wedge \forall r' \in Rep(S) : r \succ r' \vee r = r'$. $\blacksquare$

Example 3.5 r_1, i.e., $\langle 0,0,0,2,55,0 \rangle$ is the canonical representation of the time span S'. $\square$

Observation 3.2 Every strictly non-negative span has one and only one canonical representation. $\square$

The canonical representation is the best representation of a given strictly non-negative span.

Definition 3.7 *Canonical Form:* A strictly non-negative span $S = \sum_{i=1}^{N} a_i \cdot G_i$ is in *canonical form* iff $r = \langle a_i \rangle_{i=1}^{N}$ is the canonical representation of S. $\blacksquare$

Example 3.6 The canonical form for the time span S' is 2 *hour* 55 *minutes*. $\square$

3.4 Operations between Time Spans

In this section we give the semantics of arithmetic and comparison operations between time spans and show how some of the questions posed in Section 1.1 are answered.

[5]Strictly non-positive spans can be defined similarly and the canonical form can also be defined for them.

3.4.1 Arithmetic Operations between Time Spans

As described in Section 3.1, a time span is represented as a summation of different calendric granularities. In this section we elaborate on the arithmetic operations between time spans using various examples. The semantics of adding (subtracting) two time spans is to add (subtract) the components which have the same calendric granularity, concatenate the remaining components to the resulting time span, and reduce the resulting time span to canonical form as described in Section 3.3.

Example 3.7

1. $(5\ years + 4\ months) + 2\ years \rightarrow (7\ years + 4\ months)$
2. $(5\ years + 4\ months) + 15\ days \rightarrow (5\ years + 4\ months + 15\ days)$

□

Similar semantics hold true for addition (subtraction) of determinate time spans and indeterminate time spans. The following example shows the global duration of the symptoms of angina, described in sentences $S1$ and $S5$ for the patient considered in the motivating example presented in Section 1.1.

Example 3.8

$$\begin{aligned}(2\ hours + 55\ minutes) + (7\ minutes + 35\ seconds) &\rightarrow (2\ hours + 62\ minutes + 35\ seconds)\\ &\rightarrow (3\ hours + 2\ minutes + 35\ minutes)\end{aligned}$$

□

We note from the above example that the global duration of the two angina episodes is converted to its canonical form by the addition operation.

Subtraction leads to the notion of negative spans. In our model, both positive and negative spans are allowed. Positive spans have the semantics of forward duration in time, while negative spans have the semantics of backward duration in time. Allowing positive and negative spans enables us to carry out the subtraction operation between spans of different calendric granularities which could result in either a positive or negative span, for example, $1\ month - 30\ days$.

3.4.2 Comparison Operations between Time Spans

The semantics of comparing two time spans is to first convert each time span to the finest granularity that exists between the two time spans, and then carry out the comparison. The following example illustrates the various combinations that could occur:

Example 3.9

1. $(1\ \mathit{hour} + 30\ \mathit{minutes}) = 90\ \mathit{minutes}$?

   $\Leftrightarrow 90\ \mathit{minutes} = 90\ \mathit{minutes}$

   $\Leftrightarrow$ `True`

2. $1\ \mathit{month} > 30\ \mathit{days}$?

   $\Leftrightarrow (28\ \mathit{days} \sim 31\ \mathit{days}) > 30\ \mathit{days}$

   $\Leftrightarrow$ `Unknown`

□

We note from the above example that time spans which *overlap* (or even *meet* each other) cannot be compared. This follows from Observation 3.1 (see Section 3.2) which states that calendric granularities are partially ordered with respect to the binary relation "exactly convertible to."

3.5 Related Work

In this section we compare our approach of representing and operating on unanchored time durations (time spans) to that of Lorentzos [Lor94] and TSQL2 [Sno95]. Since a time span is independent of any time instant or time interval due to its relative nature, granularity conversions in the context of anchored temporal primitives cannot be used for unanchored temporal primitives. Hence, none of the temporal models [CC87, WJL91, WJS93, MPB92, MMCR92, Sno95, WBBJ97] can completely support the unanchored temporal information needs of an application like the clinical example given in Section 1.1.

Although the work of Lorentzos [Lor94] does not explicitly deal with temporal granularity, it proposes a scheme for representing and operating on non-metric types. Mixed granularity time durations, with separate fields for their composite parts (e.g., hours, minutes, seconds) are one example of a non-metric data type. These can be represented as elements of sets of composite numbers which provide conversion relationships (mappings) between the composite fields. However, only exact (regular) mappings are discussed. The representation does not provide inexact (irregular) mappings. Therefore time durations with composite parts having granularities of months and days cannot be exactly modeled. In our approach, a time span is simply a summation of calendric granularities. Both exact and inexact mappings between granularities are provided (using the $lbf(G_A, G_B)$ and $ubf(G_A, G_B)$ functions). This allows time durations to be converted to any given calendric granularity.

The conversion of a time duration to a particular granularity is possible in [Lor94]. However, the target granularity is restricted to be one of the granularities of the composite parts of the time duration. For example, if the time duration is 2 hours, 50 minutes, 30 seconds, then the time duration can be converted to hours, minutes, or seconds. We do not enforce such a restriction in our work. A time duration can be converted to any desired granularity in the calendar. The conversion process of the time duration 2 *months and* 45 *hours* to a time duration in the granularity of days is shown in Example 3.1.

In [Lor94], addition between time durations is also possible. However, the operands have to be *addition compatible*. If S_1 and S_2 are time durations, then they are addition compatible if S_2 consists of at most as many composite parts as S_1, and for these composite parts, the granularities should be the same. For example, the time durations with composite granularities (days,hours,minutes,seconds), (hours,minutes,seconds), (minutes,seconds), and (seconds) are addition compatible, and thus can be added to each other. Our approach is more general in that time durations do not have to be addition compatible. The components of the time durations which have the same calendric granularity are simply added to each other, and the remaining components are concatenated to the resulting time span, as shown in Section 3.4.1.

In TSQL2 [Sno95], time spans (durations) which have mixed granularities cannot be represented [Sno96]. For example, the duration of the chest pain in sentence $S1$ (see Section 1.1) would have to be represented in hours or in minutes. Since a time span is a summation of distinct granularities in our approach, representing symptom durations with mixed granularities is straightforward. Our approach of representing mixed granularity time spans is also more general than that used in SQL-92 in that we do not restrict time spans to only *year-month* or *day-time* combinations.

A time span in TSQL2 is necessarily indeterminate at both coarser and finer granularities. This is because a granularity is modeled as an anchored partitioning of the timeline, whereas a time span in unanchored. Therefore, all time span conversions in TSQL2 are treated as inexact, resulting in indeterminate time spans. In our approach, a time span conversion can be exact or inexact. Consider the simple conversion of the time span 1 *hour* to the granularity of minutes. In

TSQL2, this conversion results in the indeterminate span $1 \sim 119$ *minutes* – an indeterminacy of 120 minutes. In our approach however, the conversion is exact and results in the determinate span 60 *minutes*, which is what is expected in reality.

Operations involving time spans in TSQL2 could give rise to ambiguities and even incorrect results. Consider the addition of the time span 1 *hour* to the time span 40 *minutes* in TSQL2. There are two semantics defined: left-operand (coarser granularity) semantics and finer granularity semantics. In left-operand (coarser granularity) semantics, this addition can result in two different time spans:

$$\begin{aligned}
&1.\ 1\ hour + 40\ minutes \rightarrow 1\ hour + scale(40\ minutes)\\
&\qquad\qquad\qquad\qquad\quad \rightarrow 1\ hour + (0 \sim 1\ hour)\\
&\qquad\qquad\qquad\qquad\quad \rightarrow 1 \sim 2\ hours\\
&2.\ 1\ hour + 40\ minutes \rightarrow 1\ hour + cast(40\ minutes)\\
&\qquad\qquad\qquad\qquad\quad \rightarrow 1\ hour + cast(0 \sim 1\ hour)\\
&\qquad\qquad\qquad\qquad\quad \rightarrow 1\ hour + 0\ hour\\
&\qquad\qquad\qquad\qquad\quad \rightarrow 1\ hour
\end{aligned}$$

The first operation *scales* the time span of 40 *minutes* to the granularity of hours (granularity of the left operand), which results in the indeterminate time span $0 \sim 1$ *hour*. In the second operation, the time span of 40 *minutes* is *cast* to the granularity of hours. The cast operation first scales the time span 40 *minutes* to the granularity of hours which result in the indeterminate time span $0 \sim 1$ *hour* from which the first component, 0 *hour*, is arbitrarily chosen.

In finer granularity semantics, this addition can result in two different time spans:

$$\begin{aligned}
&1.\ 1\ hour + 40\ minutes \rightarrow scale(1\ hour) + 40\ minutes\\
&\qquad\qquad\qquad\qquad\quad \rightarrow (1 \sim 119\ minutes) + 40\ minutes\\
&\qquad\qquad\qquad\qquad\quad \rightarrow 41 \sim 159\ minutes\\
&2.\ 1\ hour + 40\ minutes \rightarrow cast(1\ hour) + 40\ minutes\\
&\qquad\qquad\qquad\qquad\quad \rightarrow cast(1 \sim 119\ minutes) + 40\ minutes\\
&\qquad\qquad\qquad\qquad\quad \rightarrow 1\ minute + 40\ minutes\\
&\qquad\qquad\qquad\qquad\quad \rightarrow 41\ minutes
\end{aligned}$$

The first operation *scales* the time span of 1 *hour* to the granularity of minutes (the finer granularity of the operands), which results in the indeterminate time span $1 \sim 119$ *minutes*. In the second operation, the time span of 1 *hour* is *cast* to the granularity of minutes resulting in the time span 1 *minute*.

In both cases, the addition operation yields results which are counter-intuitive to what a user actually expects, since some information is lost in the conversion process. Indeed neither semantics gives the desired result of 100 *minutes* or 1 *hour and* 40 *minutes*. In our approach, the resulting time span for the addition of 1 *hour* to 40 *minutes* is 1 *hour and* 40 *minutes*. Since a time span is represented as a summation of different calendric granularities, our semantics of arithmetic operations between time spans of different calendric granularities exactly model what is intuitively expected in the real-world.

Comparison of time spans of different granularities in TSQL2 can also lead to incorrect results. Consider the comparison of the time span 30 *minutes* with the time span 1 *hour* in TSQL2 [Dyr96], using left-operand semantics or finer granularity semantics:

$30\ minutes > 1\ hour$?
$\Leftrightarrow 30\ minutes > cast(1\ \sim\ 119\ minutes)$
$\Leftrightarrow 30\ minutes > 1\ minute$
$\Leftrightarrow$ `True`!

The time span 1 *hour* is first converted to the granularity of the leftmost operand. Since a time span is indeterminate at any finer or coarser granularity in TSQL2, the conversion of 1 *hour* to the granularity of minutes yields the indeterminate time span 1 $\sim$ 119 *minutes*. The cast operation then converts this to a determinate time span by arbitrarily choosing the lower bound. This leads to comparing the time span 30 *minutes* to the time span 1 *minute*, and subsequently returning `True` which is the opposite of what is expected. In our approach, the time span 1 *hour* would be converted exactly to the time span 60 *minutes*, and the comparison would then return `False`.

Some of the counter-intuitive results in TSQL2 may be avoided by defining duration literals for time spans with mixed granularities in the adopted calendar. The calendar would also manage different interpretations in mapping a time span from one granularity to another. Even tough calendar definition in TSQL2 allows users to overcome some of the underlined limitations, this solution seems to be less general than our approach. Further problems in TSQL2 could arise in according different calendar-dependent interpretations with the adopted scaling/left operand semantics.

4 Implementation Issues

The formulae (3) and (4) for $lbf(K, G_A, G_B)$ and $ubf(K, G_A, G_B)$ (see Derivation 2.2) are computationally expensive. However, they are not designed for direct computation. These formulae are just mathematical definitions. Any approximation of these formulae will suffice, and such approximations for the most common conversions can be chosen at the time when the calendar is defined. Another technique that can be used to make computations less expensive would be to simplify these formulae since they allow for many simplifications once a set of particular calendric functions is chosen. As an example, let us consider the Gregorian calendar. In this calendar,

$$\begin{aligned}
f^{\text{year}}(y) &= 12 \text{ (months)} \\
f^{\text{month}}(y,m) &= \begin{cases} 30 & \text{if } m \text{ is 3, 5, 8, or 10} \\ 31 & \text{if } m \text{ is 0, 2, 4, 6, 7, 9, or 11} \\ 28 & \text{if } m = 1 \text{ and } y \text{ is not leap} \\ 29 & \text{if } m = 1 \text{ and } y \text{ is leap} \end{cases} \quad \text{(days)} \\
f^{\text{day}}(y,m,d) &= 24 \text{ (hours)}
\end{aligned}$$

where y is leap when $y \bmod 400 = 0 \vee y \bmod 4 = 0 \wedge y \bmod 100 \neq 0$.

We will consider the conversions from years to months, from years to days, and from months to days. Then,

$$\begin{aligned}
f^{\text{year}\to\text{month}}(y) &= f^{\text{year}}(y) = 12 \\
f^{\text{year}\to\text{day}}(y) &= \begin{cases} 366 & \text{if } y \text{ is leap} \\ 365 & \text{otherwise} \end{cases} \\
f^{\text{month}\to\text{day}}(y,m) &= f^{\text{month}}(y,m)
\end{aligned}$$

Then, using formulae (3) and (4) we find that

$$\begin{aligned}
lbf(K, G_{\text{year}}, G_{\text{month}}) &= \min_y \{ \sum_{0 \leq dist_{\text{year}}(y',y) \leq K-1} f^{\text{year}\to\text{month}}(y') \} \\
&= \min_y \{12K\} \\
&= 12K \\
ubf(K, G_{\text{year}}, G_{\text{month}}) &= 12K
\end{aligned}$$

$$\begin{aligned}
lbf(K, G_{\text{year}}, G_{\text{day}}) &= \min_y \{365(y+K) + \lfloor (y+K)/4 \rfloor - \lfloor (y+K)/100 \rfloor + \lfloor (y+K)/400 \rfloor \\
&\quad - 365y - \lfloor y/4 \rfloor + \lfloor y/100 \rfloor - \lfloor y/400 \rfloor \} \\
&\geq 365K + \lfloor K/4 \rfloor - \lfloor (K+96)/100 \rfloor \\
ubf(K, G_{\text{year}}, G_{\text{day}}) &\leq 365K + \lfloor (K+3)/4 \rfloor
\end{aligned}$$

The above lbf and ubf bounds can be used instead of exact formulae. These bounds are easily computable and introduce an error that is less than a day per century. Analogous methods can be used to find computationally cheap approximations for conversion of months to days; however, to obtain reasonable approximations, values for small K ($K \leq 48$) have to be tabulated. Let $g_{min}(K)$ and $g_{max}(K)$ be such tabulations. Then we have:

$$lbf(K, G_{\text{month}}, G_{\text{day}}) = g_{min}(K \bmod 48) + lbf(\lfloor K/48 \rfloor \cdot 4, G_{\text{year}}, G_{\text{day}})$$
$$ubf(K, G_{\text{month}}, G_{\text{day}}) = g_{max}(K \bmod 48) + ubf(\lfloor K/48 \rfloor \cdot 4, G_{\text{year}}, G_{\text{day}})$$

Using these formulae we can now make fast and quite precise conversions. For example, the number of days (d) in 100 months according to the above formulae is $120 + 2921 = 3041 \leq d \leq 3044 = 122 + 2922$, which is the correct estimate. The simplistic approach where the number of months is not taken into account when the coefficients are computed would give $28 * 100 = 2800 \leq d \leq 3100 = 31 * 100$, which is an error of more than 8%, or 200 days.

5 Conclusion

This paper is a first step in completing the puzzle on temporal granularity by providing support for unanchored temporal primitives with different and mixed granularities and addressing the issues that arise therein. This will help temporal DBMSs to better support the various applications in which such primitives are inherent.

A model for supporting calendars is given and it is shown how multiple granularities and unanchored temporal entities are integrated within the context of calendars. A calendric granularity is described as being part of a calendar and represented as a special kind of span - one with a unit duration. The process of conversion between time spans of mixed granularities is then given and canonical forms for time spans defined. These forms are used by arithmetic operations on time spans to return time spans which are what a user intuitively expects. The arithmetic and comparison operations involving time spans were described and compared to similar ones in [Lor94] and in the TSQL2 query language [Sno95]. We show how our semantics of operations are more general than that of [Lor94]. Furthermore, our operations exactly model intuition, while the semantics of operations in TSQL2 yield results which were either counter-intuitive (in the case of arithmetic operations) or incorrect (in the case of comparison operations).

In Section 4 we show that it is possible to establish computationally inexpensive yet quite precise formulae for lower and upper bound coefficients. Currently, the derivation of these formulae has to be done by a database administrator; their automatic derivation is a topic for future research. In

[GLÖS96, Gor98] we describe how the calendar model, and the anchored and unanchored temporal primitives introduced in this paper are incorporated into the TIGUKAT object model [ÖPS$^+$95].

In conclusion, we emphasize that modeling of unanchored temporal data with different and mixed granularities is an essential ingredient of temporal granularity, and should therefore be considered in the design of temporal models and temporal query languages. It is our position that assuming a simplistic view of unanchored temporal data and thereby avoiding the inherent issues which arise will only make the resulting temporal model and temporal query language very restricted for real-world temporal data usage.

References

[BMJ94] E. Braunwald, D.B. Mark, and R.H. Jones. Diagnosing and Managing Unstable Angina - Quick Reference Guide for Clinicians. (10. AHCPR Publication No. 94-0603), May 1994.

[BP85] F. Barbic and B. Pernici. Time Modeling in Office Information Systems. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pages 51–62, Austin, Texas, May 1985.

[BWJ96] C. Bettini, X. Wang, and S. Jajodia. Testing complex temporal relationships involving multiple granularities and its application to data mining. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium in Principles of Database Systems (PODS)*, Montreal, Canada, 1996.

[CC87] J. Clifford and A. Crocker. The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans. In *Proc. 3rd Int'l. Conf. on Data Engineering*, February 1987.

[CMR91] E. Corsetti, A. Montanari, and E. Ratto. Dealing with Different Time Granularities in Formal Specifications of Real-Time Systems. *The Journal of Real-Time Systems*, 3(2):191–215, 1991.

[CPP96] C. Combi, F. Pinciroli, and G. Pozzi. Managing Time Granularity of Narrative Clinical Information: The Temporal Data Model TIME-NESIS. In L. Chittaro, S. Goodwin, H. Hamilton, and A. Montanari, editors, *Third International Workshop on Temporal Representation and Reasoning (TIME'96)*, pages 88–93, Los Alamitos, California, 1996. IEEE Computer Society Press.

[CCP97] C. Combi, G. Cucchi, and F. Pinciroli. Applying Object-Oriented Technologies in Modeling and Querying Temporally-Oriented Clinical Databases Dealing with Temporal Granularity and Indeterminacy. *IEEE Transactions on Information Technology in Biomedicine*, 1(2):100–127, 1997.

[CR88] J. Clifford and A. Rao. A Simple, General Structure for Temporal Domains. In C. Rolland, F. Bodart, and M. Leonard, editors, *Temporal Aspects in Information Systems*, pages 17–30. North-Holland, 1988.

[DS93] C.E. Dyreson and R.T. Snodgrass. Valid-time Indeterminacy. In *Proc. 9th Int'l. Conf. on Data Engineering*, pages 335–343, April 1993.

[Dyr96] C. Dyreson. June 1996. Private correspondence.

[Flo91] R. Flowerdew. *Geographical Information Systems.* John Wiley and Sons, 1991. Volume 1.

[GLÖS] I.A. Goralwalla, Yuri Leontiev, M.T. Özsu, and Duane Szafron. Modeling Temporal Primitives: Back to Basics. Sixth International Conference on Information and Knowledge Management (CIKM'97), pages 24 - 31, November, 1997.

[GLÖS95] I.A. Goralwalla, Y. Leontiev, M.T. Özsu, and D. Szafron. A Uniform Behavioral Temporal Object Model. Technical Report TR-95-13, University of Alberta, May 1995.

[GLÖS96] I.A. Goralwalla, Y. Leontiev, M.T. Özsu, and D. Szafron. Modeling Time: Back to Basics. Technical Report TR-96-03, University of Alberta, February 1996.

[Gor98] I. Goralwalla. Temporality in Object Database Management Systems. PhD Thesis, University of Alberta, Canada, March 1998.

[Lor94] N. Lorentzos. DBMS Support for Non-Metric Measuring Systems. *IEEE Transactions on Knowledge and Data Engineering*, 6(6):945–953, December 1994.

[MMCR92] A. Montanari, E. Maim, E. Ciapessoni, and E. Ratto. Dealing with Time Granularity in Event Calculus. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 702–712, June 1992.

[MPB92] R. Maiocchi, B. Pernici, and F. Barbic. Automatic Deduction of Temporal Information. *ACM Transactions on Database Systems*, 17(4):647–688, 1992.

[ÖPS$^+$95] M.T. Özsu, R.J. Peters, D. Szafron, B. Irani, A. Lipka, and A. Munoz. TIGUKAT: A Uniform Behavioral Objectbase Management System. *The VLDB Journal*, 4:100–147, August 1995.

[Sno95] R. Snodgrass. *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, 1995.

[Sno96] R. Snodgrass. May 1996. Private correspondence.

[WBBJ97] X.S. Wang, C. Bettini, A. Brodsky, and S. Jajodia. Logical Design for Temporal Databases with Multiple Granularities. *ACM Transactions on Database Systems*, 22(2):115-170, 1997.

[WJL91] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with Granularity of Time in Temporal Databases. In R. Andersen, J.A. Bubenko Jr., and A. Solvberg, editors, *Advanced Information Systems Engineering, 3rd Int'l Conference CAiSE '91*, pages 124–140. Springer-Verlag, 1991.

[WJS93] X.S. Wang, S. Jajodia, and V. Subrahmanian. Temporal Modules: An Approach Toward Temporal Databases. In *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pages 227–236, 1993.