

Modeling Periodic Time and Calendars

Atakan Kurt & Meral Ozsoyoglu

Computer Eng. and Sci. Department
Case Western Reserve University
Crawford Hall, 5th floor
Cleveland OH 44116
{kurt, ozsoy}@ces.cwru.edu

Abstract

Periodic temporal events occur frequently in scheduling, planning, medical records, calendars, scientific, multimedia, active databases etc. A framework for describing and reasoning about periodic events in general and calendars and calendric events in particular is presented. Existing temporal models support only aperiodic events with a few exceptions. We present an approach based on a new periodic temporal type called *periodic element* to model periodic time and calendars that can easily be incorporated in an object oriented language as a type hierarchy. Our paradigm is compatible with existing aperiodic temporal data models, i.e., it can be used to extend them seamlessly to incorporate periodic time into the model. Periodic elements (i) are capable of representing not only *aperiodic* and *strictly-periodic* temporal events but also *partially-periodic* events as a single temporal value. (ii) They can represent *absolute* and *relative* events uniformly (with conversions between the two), and (iii) are closed under the set theoretic operators with a set of optimization rules. Our paradigm is user friendly in the sense that we don't introduce new operators or constructs but rather extend the existing operators of aperiodic types for periodic time. We also point out how periodic elements can be used for modeling multimedia, active and temporal databases involving periodic events.

Keywords: Temporal databases, calendars, calendric events, multimedia databases, active-databases, absolute events, relative events, aperiodic events, periodic events, partially periodic events.

1. INTRODUCTION

Applications such as scheduling, planning, forecasting, time-management, time-series, scientific databases, multimedia databases, active databases, banking, law, medical records, accounting, process

control, inventory control, GIS deal with periodic events. For example, the planning of future activities in time-series databases depends on the accuracy of estimated market variables. These variables such as stock prices, inflation may be considered in periodic intervals of time. In scientific databases, experiments and computations are sometimes performed periodically. In real-time control systems and real-time databases, the output and/or input of a real-time system may be analyzed/stored periodically. The set of tasks a robot performs is an example of a periodic event in real-time systems. A multimedia database stores video and audio sequences which must be synchronously played out. Such play-out of audio/video sequences may involve periodic events. Active databases are characterized by the *on-event if-condition do-action* rules. These rules should be evaluated periodically to check if any of the conditions are satisfied. The *if-condition* clause may include a periodic expression such as *every 10 minutes, every Monday or the first day of every month.*

Calendars are a means for specifying and reasoning about time used extensively in business applications. Events in temporal databases such as those in [CICr 87, NaAh 87, Ga 88] are usually defined with respect to one or more calendar granularity. In general, calendars involve multiple granularity such as days, weeks, years [Da 88]. It is difficult to capture the semantics of calendars using aperiodic temporal constructs like instants, intervals [All 83], temporal elements [Ga 85]. Natural language expressions involving calendars (called *calendric expressions*) like *the first Monday of every month, the fourth Thursday of Novembers, the work days in year 1995, every year between Thanksgiving and Christmas* are even more difficult to represent and manipulate, because of the richness of the semantics.

Calendars and calendric events are modeled in the literature using the collection of intervals [Le 86], linear repeating intervals [NiSt 92], temporal

mediators [WaJa 93], or object oriented types [Te 93]. A collection of intervals [Le 86] is a hierarchically structured lists of intervals interpreted cyclically to define calendars and possibly infinite periodic events. Chandra et al [Ch 94] improved and implemented the calendar algebra in [Le 86]. This implementation supports periodic events within a finite interval of time. Michael Soo [So 93] proposes an extension to SQL2 [Me 90] that supports multiple calendars. The main idea of this proposal is to separate the user dependent features of calendars from the universal ones. The SQL2 extension provides the set of universal features for calendars, and leaves the user dependent aspects to the database managers. Temporal mediators [WaJa 93] are a means for converting between different calendar granularity. The main purpose of this paradigm is to display the data in a temporal relational database in terms of a different time granularity. Generalized databases [Ka 90] uses linear repeating points or *lrp* (an infinite set of equally-spaced integers denoted by a linear formula) and a set of constraints to model periodic events in relational databases. The use of *lrps* gives rise to the duplication of data over multiple tuples. In [NiSt 92], Niezette and Stevenne propose linear repeating intervals (*lri*) instead of *lrps* to overcome the difficulties of the *lrps* in expressing calendars within the same theoretical framework. A *lpi* is a possibly infinite set of fixed length equally-spaced set of intervals. However some difficulty with generalized databases still exists. For example, a calendric event or just one calendar may need to be stored as a set of generalized tuples resulting in storage redundancy and compromised data integrity, because a calendar may correspond to more than one *lri* [NiSt 92]. An object-oriented approach is taken for modeling calendars in Eiffel language in [Te 93]. The implementation of calendars are based on instants (time points), intervals, and durations types in a similar fashion to [So 93]. Periodic events in are also studied in the context of deductive databases [ChIm 88, B 91, B 93]

The approaches mentioned above have a number of deficiencies: (i) The existing approaches don't distinguish between the *aperiodic* and *strictly-periodic* part of a *partially-periodic* event. Consider an employee who is employed part-time after being a temporary employee for 3 weeks. The work-schedule of this employee is *partially-periodic*, i.e., the work-schedule as a temporary employee is *aperiodic*, while the work-schedule as a part-time employee is *strictly-periodic* (Temporal events are

described formally in Section 3.) With existing approaches such a temporal history/event has to be stored as at least two separate object resulting in storage redundancy, inefficient query processing. (ii) In most cases [WaJa 93, Ch94, So 93], the reasoning about periodic time is not fully automated, the user has to write a significant amount of code to describe how to do the conversions between calendars. (iii) Calendric events defined periodically with respect to other events like *every year between Thanksgiving and Christmas* can't be expressed by the existing approaches at symbolic level. (iv) Furthermore a good calendar algebra should also support *and*, *or*, and *not* of natural languages. (v) Lastly the existing approaches do not attempt to model relative calendric events (events with no absolute time references). There is need for a comprehensive approach on modeling periodic events in general, and modeling calendars and calendric expressions in particular. In this paper, we attempt to model calendars using a new temporal type called periodic elements.

The rest of the paper is organized as follows: The basic concepts on temporal events and calendars are presented informally in Section 2. We briefly introduce periodic elements in Section 3. Modeling calendars and calendric events through periodic elements are studied in Section 4. Specification and derivation of calendars from existing calendars are also presented. New calendar operators *periodic intervals*, *periodic interval selection* are discussed in this section. In Section 5, the temporal relationships between periodic events are discussed. The modeling of relative events similar to absolute events is studied in Section 6. Optimization of calendric expressions are summarized in Section 7. Section 8 is reserved for comparison of our paradigm with the related research. Finally we summarize and conclude in Section 9.

2. TEMPORAL EVENTS AND CALENDARS

A temporal event is a happening whose temporal aspect is of importance to the user. Temporal events can be categorized into 3 groups in the order of increasing complexity: (i) *aperiodic*, (ii) *strictly-periodic* and (iii) *partially-periodic* events. *Aperiodic* temporal events are those represented by instants, intervals [Al 83], or temporal elements [Ga 85, Ga 88] (These concepts are explained in the next section.) *Strictly-periodic* events are events

happening periodically with no specific end time, i.e., occurring indefinitely. The calendar of Tuesdays is an example for a strictly-periodic event. Events happening aperiodically before a specific instant and happening periodically after that instant are called *partially-periodic* events. (Unless otherwise specified, the term *periodic event* refers to *partially-periodic* events in the remainder of this paper.) Partially-periodic events occur frequently in real world. For example, the work-schedule history of the temporary employee mentioned in the previous section is partially-periodic. Note that as will be clear from the definitions in the next section, a strictly periodic event is a special kind of partially periodic event. The introduction of partially periodic events does not make the distinction between aperiodic and periodic events useless, but rather it extends the types of events we deal into a more general set of events.

A periodic event is called a *basic calendar*, *time unit* or *granularity* (depicted in Figure 4) if it satisfies the following properties:

1. A basic calendar always start from 0th clock tick.
2. Consecutive intervals in a basic calendar are abut in time. i.e., they neither overlap, nor there is a gap between two consecutive intervals.
3. A basic calendar consists of an infinite number of intervals, i.e., it extends into infinity.
4. The intervals are defined cyclically or periodically. The length of a duration cycle is called the period of calendar.

The period of the calendar of years is 4 years because of the leap years in which February is 29 days long. The most common examples for basic calendars are the calendar of seconds, the calendar of minutes, etc. Basic calendars are building blocks of more complex calendars and calendric events.

An *arbitrary calendar* (or calendar for short) is defined similar to basic calendars but differs from them in two aspects. (i) The intervals in a calendar need not be abut in time i.e. there can be gaps between 2 consecutive intervals. (ii) An arbitrary calendar may start from an instant other than the 0th clock tick. Examples of arbitrary calendars include the calendar of time-cards, calendar of fiscal year, the academic calendar, the calendar of paydays.

A calendric event is a temporal event expressed with respect to a calendar. Hence a

calendric event can be aperiodic, strictly-periodic or partially-periodic event. Calendric expressions are formulas corresponding to the natural language expressions on calendars in temporal applications. Calendric expressions are built from calendars and other calendric events through the temporal constructs extended for periodic time, the relational operators and the set theoretic operators.

3. PERIODIC ELEMENTS

In this section we introduce *periodic elements*, a temporal type for modeling periodic temporal events. The existing temporal models and temporal query languages based on instants, intervals or temporal elements can be extended seamlessly with periodic elements to incorporate periodic events into the model, i.e., the syntax and semantics of aperiodic types and operators in temporal databases are preserved under periodic elements. In this paper, we will use periodic elements to model calendars and calendric events, illustrating that periodic elements can be used uniformly in temporal databases to model both the database itself [KuMe 95] and the calendars, eliminating the need for a separate model for calendars.

Since periodic elements are formally based on aperiodic temporal types, we will first review these types briefly. A detailed presentation of aperiodic types and periodic elements can be found in [K 95]. Sometimes the operators for temporal elements are underlined while the ones for periodic elements are double-underlined just for clarity (all operators are overloaded in reality).

We assume the existence of a smallest time interval called *chronon* that can be measured by hardware. A chronon is the time between 2 consecutive clock ticks of the system clock. For simplicity, we assume that the length of a chronon is a second. There are a set of temporal constants defined as follows: The constant 0 is the beginning of the physical time line. The clock ticks are numbered with the natural numbers. *now* is a temporal constant representing the current clock tick (the current time) which is maintained by the underlying operating system. The constant ∞ is used to represent the positive infinity. \emptyset represents an empty temporal value.

An *instant t* of time is a specific point on the physical time line, represented either by integers

(discrete time), rationals (dense time), or reals (continuous time). An interval $[b, e]$, $(b, e]$, $[b, e)$, or (b, e) is the time duration beginning at instant b and ending at instant e with open ends excluded. Every instant t corresponds to the interval $[t, t]$. The functions *begin* and *end* return the first and the last instant of an interval respectively: For $I = [b, e]$, $begin(I) = b$ and $end(I) = e$. Intervals are not closed under set theoretic operators which causes the duplication of data. The temporal relationships between intervals are captured by the relational operators *before*, *after*, *meets*, *overlaps*, etc. defined in [Al 83]. For instance, I before J evaluates to true if interval I ends before interval J begins, i.e., $end(I) < begin(J)$. Similarly, I meets $J = true$ if $end(I) + 1 = begin(J)$ (for discrete time.) A temporal element N is a finite set of intervals representing events happening over more than one interval of time [Ga 88]. A temporal element N is written as

$$N = ([b_1, e_1], [b_2, e_2], \dots, [b_k, e_k]) \text{ for } k \geq 1.$$

Parenthesis can be omitted, if there is no confusion. By definition every interval $[b, e]$ is a temporal element. The set of temporal elements are closed under the set theoretic operations of union \cup , intersection \cap and complementation \sim , with T (the set of all time instants) as its maximum element and \emptyset as its minimum element [Ga 88]. For example, $([3, 7], [10, 14]) \cup ([5, 8], [14, 16]) = ([3, 8], [10, 16])$. A temporal element can be translated and scaled as follows to express the changes about the beginning and the duration of an event respectively. These temporal transformations are formulated as follows:

- *translate* $(([t_1, t_2], \dots, [t_{n-1}, t_n]), d) = ([t_1+d, t_2+d], \dots, [t_{n-1}+d, t_n+d])$
- *scale* $(([t_1, t_2], \dots, [t_{n-1}, t_n]), s) = ([t_1, t_1+(t_2-t_1)*s], \dots, [t_1+(t_{n-1}-t_1)*s, t_1+(t_n-t_1)*s])$

The *translate* function moves an event by d time units on the time line and the *scale* function shrinks or expand an event by a factor of s . The temporal relationship operators *before*, *after*, *contains*, *overlaps*, etc. can easily be extended to temporal elements. Therefore temporal elements are more

appropriate than intervals for modeling temporal data and query languages [Ga 88]. However temporal elements represent only aperiodic events. We now introduce periodic elements that overcomes this drawback. Below I denotes the set of natural numbers. I_m, J_k denote intervals.

Definition 1 (Periodic element): A triplet of two temporal elements $N = (I_1, I_2, \dots, I_n)$ and $P = (J_1, J_2, \dots, J_m)$, and a duration $p \in I$ is called a periodic element, denoted by $N:P\{p\}$, where $p \geq 0$, I_n before $J_1 = true$, and $p \geq end(J_m) - begin(J_1)$.

A periodic element consists of three components depicted in Figure 1:

1. *Aperiodic part* N is a temporal element denoting the initial irregular temporal part of the temporal event,
2. *Periodic part* P is also a temporal element denoting the regular indefinitely-many-repeating part of the temporal event, and
3. *Period* p is a duration of time between two consecutive occurrence of *periodic part* P .

The condition $p \geq 0$ ensures that the periodic event being represented extends into positive infinity (the future). The second condition states that the aperiodic part of the event must happen before the periodic part to distinguish between the two. The third condition asserts that the period must be at least as long as the length of the periodic part of the event. These three conditions may be relaxed to represent more complex events. A periodic element is the union of its periodic $P\{p\}$ and aperiodic N parts:

$$N:P\{p\} = N \cup P\{p\}.$$

where the periodic part P happens in every p time units: $P\{p\} = P \cup \text{translate}(P, p) \cup \text{translate}(P, 2p) \cup \dots$. A periodic element is essentially an infinite union of intervals with the periodic part repeating indefinitely.

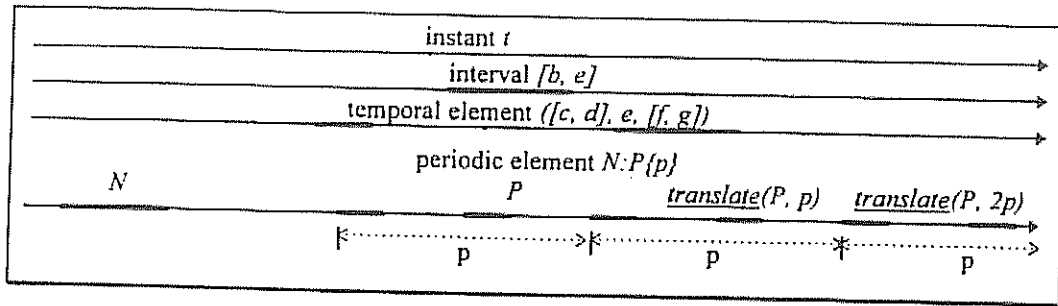


Figure 1: Aperiodic types and Periodic Element.

If the periodic part of a periodic element is an empty element ($P = \emptyset$) or the period is zero ($p = 0$), then this periodic element reduces to a temporal element, in which case the periodic part is omitted in the representation and written N for short. Thus every temporal element (therefore every interval and instant) is a periodic element by definition (This allows us to extend the existing aperiodic temporal database models seamlessly for periodic data.) If the aperiodic part of a periodic element is an empty element ($N = \emptyset$), then the aperiodic part is omitted in the representation and written $P\{p\}$ for short. Functions *aperiodic_part*, *periodic_part* and *period* returns corresponding parts of a given periodic element $N:P\{p\}$ respectively: $period(N:P\{p\}) = p$, $aperiodic_part(N:P\{p\}) = N$, and $periodic_part(N:P\{p\}) = P$. The followings are valid periodic elements:

- 5.
- [3, 7].
- ([2, 5], [8, 10]).
- [2, 5]{6} = ([2, 5], [8, 11], [14, 17], ...)
- ([0, 4], [6, 7]):[9,10]{5} = ([0, 4], [6, 7], [9,10], [14, 15], [19, 20], ...)

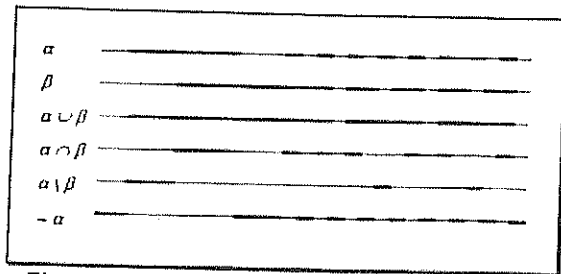


Figure 2: The set theoretic operators on periodic elements.

The Boolean connectives *and*, *or*, *not* are widely used in applications of periodic time. Periodic events are closed under set theoretic operations of union \cup intersection \cap difference \setminus , and

complementation \simeq even if the periods of the operands are different. For example, $[0, 1]\{2\} \cup [0, 2]\{3\} = [0, 5]\{6\}$. These operations are depicted in Figure 2 and formal semantics are omitted because space limitations. When defining a periodic events such as calendric or multimedia events in terms of other periodic events, temporal transformations are utilized quite often. The transformation functions *translate*, and *scale* on periodic elements are defined as follows (Figure 3):

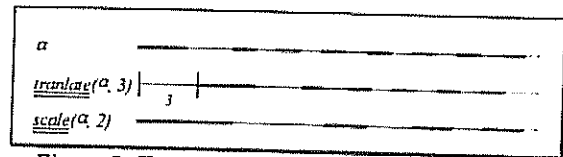


Figure 3: Temporal transformations of periodic elements

- $translate(N:P\{p\}, d) = translate(N, d) : translate(P, d)\{p\}$
- $scale(N:P\{p\}, s) = scale(N,s) : translate(scale(P,s), begin(P)*(s1))\{p*s\}$

4. MODELING CALENDARS WITH PERIODIC ELEMENTS

In this section, we formally introduce aperiodic, strictly-periodic, partially-periodic temporal events first. Basic/arbitrary calendars, and calendric events are then defined as special types of periodic events.

Definition 2: A temporal event represented by a periodic element $N:P\{p\}$ is called a *partially-periodic event* (periodic for short). A *partially-periodic event* is called

- an *aperiodic event* if the periodic part is empty ($P = \emptyset$),
- a *strictly-periodic event* if the aperiodic part is empty ($N = \emptyset$).

5, [3, 7], ([2, 5], [8, 10]) are examples of aperiodic events, while [2, 5]{6} and ([0, 4], [6, 7]):[9,10]{5} are examples for strictly-periodic and partially-periodic events respectively.

Definition 3: A basic calendar α is a strictly-periodic event $(I_1, I_2, \dots, I_n)\{p\}$ satisfying the following conditions for $n, p \in I$:

- $\text{begin}(I_i) = 0$.
- $I_i \text{ meets } I_{i-1} = \text{true}$ for $1 \leq i < n$.
- $p = \sum_{j=1}^n \text{length}(I_j)$

The formula $\text{begin}(\alpha) = 0$ enforces that the beginning of a basic calendar is the first clock tick. The formula $I_i \text{ meets } I_{i-1} = \text{true}$ assures that the intervals do not overlap and there is no gaps between consecutive intervals. The intervals I_1, I_2, \dots, I_n corresponds to the time units of the calendars. p is the period of the calendar. Basic and arbitrary calendars are depicted in Figure 4.

The condition $I_i \text{ before } I_{i-1} = \text{true}$ states that there may be gaps between intervals in an arbitrary calendar. The intervals I_1, I_2, \dots, I_n need not be of equal length.

Example 2: The calendar of Mondays (*Mondays*), the calendar of 7-day work-hours 8:00am-12:00pm and 1:00pm-5:00pm (*Workhours*), the calendar of midnight's (*Midnights*) can be specified as follows:

$$\begin{aligned} \text{Mondays} &= [0, 1d)\{7d\} \\ &= [0, 1d) \cup [7d, 8d) \cup [14d, 15d) \cup \dots \end{aligned}$$

$$\begin{aligned} \text{Workhours} &= ([8h, 12h], [13h, 17h])\{1d\} \\ &= [8h, 12h] \cup [13h, 17h] \cup [32h, 36h] \cup \dots \end{aligned}$$

$$\begin{aligned} \text{Midnights} &= 0 \{1d\} \\ &= 0 \cup 1d \cup 2d \cup \dots \end{aligned}$$

where the constants h and d stands for 3600 and 86400 clock ticks respectively and used for brevity.

As seen from the examples above, specifying

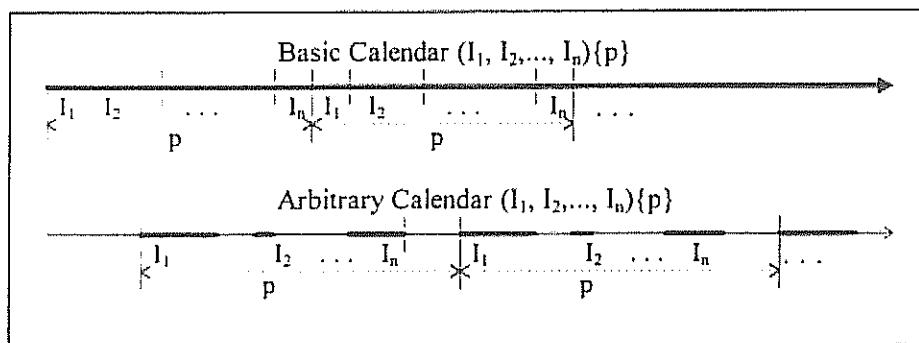


Figure 4: Basic and arbitrary calendars.

Example 1: The basic calendar of seconds (*Seconds*), minutes (*Minutes*), etc. can be given as a single periodic element in terms of clock ticks assuming the clock ticks at every second:

$$\begin{aligned} \text{Seconds} &= [0, 1)\{1\} \\ &= [0, 1) \cup [1, 2) \cup [2, 3) \cup \dots \end{aligned}$$

$$\begin{aligned} \text{Minutes} &= [0, 60)\{60\} \\ &= [0, 60) \cup [60, 120) \cup [120, 180) \cup \dots \end{aligned}$$

$$\begin{aligned} \text{Hours} &= [0, 3600)\{3600\} \\ &= [0, 3600) \cup [3600, 7200) \cup [7200, 10800) \cup \dots \end{aligned}$$

Definition 4: An arbitrary calendar is a strictly-periodic event $(I_1, I_2, \dots, I_n)\{p\}$ where $I_i \text{ before } I_{i-1} = \text{true}$ for $1 \leq i < n$.

calendars in terms of the clock ticks (the physical time) is very cumbersome and difficult. This method may even be impractical for complex calendric events and for calendars like years and centuries. To overcome this difficulty we show how to derive calendars from existing calendars through temporal transformations in the next subsection.

4.1 Derivation of Calendars

Instead of explicitly specifying a calendar in terms of clock ticks, the mathematical relationships between calendars can be utilized for defining a calendar from existing ones in a straightforward manner. This can be done in two ways.

1. Calendar C_1 can be a *scaled* version of calendar C_2 .
2. Calendar C_1 can be a *translation* of calendar C_2 .

We think that using transformations is more natural and easier than using special constructs and operators just for the purpose of defining calendars as in [Le 86, NiSt 92, Ch 94, So 92]. For example, the temporal transformations capture the semantics of *synchronization*, *duration* and *period* operators in [NiSt 92]. Since periodic elements are closed under scaling and translation, every calendar evaluates to a single periodic element.

Example 3: Consider the calendar of minutes (*Minutes*) which is exactly the same as the calendars of seconds (*Seconds*) except the intervals are 60 times longer than the intervals in the calendar of seconds. Similar relationship exists between other calendars.

$$\begin{aligned} \text{Minutes} &= \text{scale}(\text{Seconds}, 60) \\ &= \text{scale}(\{0, 1\}\{1\}, 60) \\ &= \{0, 60\}\{60\} \\ &= \{0, 60\} \cup \{60, 120\} \cup \dots \end{aligned}$$

$$\begin{aligned} \text{Hours} &= \text{scale}(\text{Minutes}, 60) \\ &= \{0, 3600\}\{3600\} \\ &= \{0, 3600\} \cup \{3600, 7200\} \cup \dots \end{aligned}$$

$$\begin{aligned} \text{Days} &= \text{scale}(\text{Hours}, 24) \\ &= \{0, 86400\}\{86400\} \\ &= \{0, 86400\} \cup \{86400, 172800\} \cup \dots \end{aligned}$$

$$\text{Weeks} = \text{scale}(\text{Days}, 7) = \dots$$

Consider the calendar of Mondays, Tuesdays, etc. Mathematically the calendar of Tuesdays is exactly the same as the calendar of Mondays except the intervals in it are shifted to the left by one day. Let *Mondays* = $\{0, 1d\}\{7d\}$ be the calendar of Mondays. Then

$$\begin{aligned} \text{Tuesdays} &= \text{translate}(\text{Mondays}, 1d) \\ &= \{2d, 3d\}\{7d\} \\ &= \{2d, 3d\} \cup \{9d, 10d\} \cup \{16d, 17d\} \cup \dots \end{aligned}$$

$$\text{Wednesdays} = \text{translate}(\text{Mondays}, 2d) = \dots$$

In a similar fashion, the calendar of months of equal-length can be derived from one another. For example, the calendar of months of length 31 days (January, March, May, July, etc.) can be derived from the calendar of March (*Marches*) as follows:

$$\begin{aligned} \text{May} &= \text{translate}(\text{Marches}, 61d) \\ \text{July} &= \text{translate}(\text{Marches}, 122d) \end{aligned}$$

More complex relations between calendars can be expressed by a combination of temporal transformations. For example, the calendar of Aprils (30 days long) can be derived from the calendar of Marches (31 days long) as follows:

$$\text{Aprils} = \text{translate}(\text{scale}(\text{Marches}, 30d/31d), 31d).$$

We have studied the derivation of calendars above. Below we investigate calendric events and how they are specified in terms of a calendar.

4.2 Calendric Events

A periodic temporal event expressed with respect to a calendar is called a *calendric event*.

Definition 5 (Calendric event): A calendric event α_C is a temporal event $\alpha = (\{c_1, c_2\}, \{c_3, c_4\}, \dots, \{c_n, c_{n+1}\}) : (\{c_1, c_2\}, \{c_3, c_4\}, \dots, \{c_n, c_{n+1}\})\{p\}$ expressed with respect to the time units (intervals) of basic calendar $C = \{b, e\}\{e-b\}$.

Here α_C serves as a mapping from C to *Chronons*, i.e., $\alpha_C = \text{scale}(\alpha, e-b)_{\text{Chronon}}$. Each instant t in α corresponds to the beginning of the n th interval in C . The calendar of clock ticks is the default calendar, i.e., $\alpha = \alpha_{\text{Chronon}}$.

Example 4: The 5 work days (*Workdays*) can be specified in terms of the calendar of days (*Days*) as follows:

$$\text{Workdays} = \{0, 5\}\{7\}_{\text{Days}}$$

where the numbers 0, and 5 refers to the beginning points of the first and the 6th intervals in the calendar of days. The number 7 represents the length of 7 days. The 7 day work-hours (*Workhours*) can be given in the calendar of hours (*Hours*) as follows:

$$\text{Workhours} = (\{8, 12\}, \{13, 17\})\{24\}_{\text{Hours}}$$

The work days of an employee who works Mondays through Wednesdays after going through 2 full weeks of training may be represented by the following calendric event with respect to the calendar of days, assuming Monday is the first day in the calendar:

$$\text{EmployeeHour} = [0, 15) : [15, 18) \{7\}_{\text{Days}}$$

Since calendric events are represented by periodic elements, the set theoretic operations for periodic elements (discussed in Section 3) can be used on calendric events as illustrated by the example below.

Example 5: Weekends (Weekends) consists of Saturdays and Sundays:

$$\text{Weekends} = \text{Saturdays} \cup \text{Sundays}$$

Workdays (*Workdays*) Monday through Friday, can be given by the difference or the union operator:

$$\begin{aligned} \text{Workdays} &= \text{Days} \setminus \text{Weekends} \\ &= \text{Mondays} \cup \dots \cup \text{Fridays} \end{aligned}$$

Workdays in the month of Januarys (*WorkdaysinJanuarys*) and in Jan. 1995 (*WorkdaysinJan95*) can be computed by the intersection operator where the results are strictly-periodic and aperiodic calendric events respectively.

$$\begin{aligned} \text{WorkdaysinJanuarys} &= \text{Workdays} \cap \text{Januarys} \\ \text{WorkdaysinJan95} &= \text{Workdays} \cap [\text{Jan-1-95, Jan-31-95}] \end{aligned}$$

4.3 Periodical Interval Selection

Expressions like *every Christmas* (which may occur in an active database application), i.e., the set of all Christmases, can not be expressed easily, because Christmas is the 25th day of December which has a period of 4 years (because of leap years). In this section we introduce the periodic interval selection function on periodic elements to express such statements. Periodical interval selection is a natural extension of the interval selection function on temporal elements. The function *interval* selects an interval from a temporal element indicated by the parameter j , i.e., $\text{interval}((I_1, I_2, \dots, I_n), j) = I_j$ for $1 \leq j \leq n$. Functions *firstinterval*, *lastinterval* are special cases of function *interval* selecting the first and the last intervals respectively. For $\alpha = ([3, 5], [7, 11])$, $\text{interval}(\alpha, 2) = [7, 11]$ and $\text{firstinterval}(\alpha) = [3, 5]$. The periodic *interval* selection function selects intervals from calendric events periodically according to another calendric event or calendar. Periodical *firstinterval* and *lastinterval* functions are defined similarly.

Definition 6 (Periodic Interval Selection): Let α and β be two calendric events. The periodical interval selection function $\text{interval}^\beta(\alpha, i)$ is a periodic application of function *interval* to event α within each interval in event β .

$$\text{interval}^\beta(\alpha, i) = \cup_{k=1}^{\infty} \text{interval}(\text{interval}(\beta, k) \cap \alpha, i)$$

The formula $\text{interval}(\text{interval}(\beta, k) \cap \alpha, i)$ selects the i th interval from α within the duration of k th interval in β . Note that both events can be partially-periodic. The formula on the left hand side eventually evaluates to a single periodic element because of the closure property of periodic elements under set theoretic operators. For aperiodic events, *interval* reduces to aperiodic interval selection function *interval*. The following example illustrates the periodic interval selection.

Example 6: Let *Months* and *Years* be the calendar of months and years¹. Then the calendar of Decembers (*Decembers*) can be specified by periodically selecting the 12th interval within every year:

$$\text{Decembers} = \text{interval}^{\text{years}}(\text{Months}, 12)$$

The expression *every Christmas (Christmases)* can be specified by periodically selecting the 25 day of Decembers:

$$\text{Christmases} = \text{interval}^{\text{years}}(\text{Decembers} \cap \text{Days}, 25)$$

Note that the expression $\text{interval}^{\text{Decembers}}(\text{Decembers} \cap \text{Days}, 25)$ also yields the same result. The last Friday of every January can be expressed by periodically choosing the last interval from the Fridays in Januarys (expressed by intersecting the calendar of Januarys with the calendar of Fridays).

$$\begin{aligned} \text{LastFridayofJanuarys} &= \\ &\text{lastinterval}^{\text{years}}(\text{Januarys} \cap \text{Fridays}) \end{aligned}$$

4.4 Periodic Intervals

Aperiodic (ordinary) intervals [A1 83] denote events with specific beginning and end times. It would be very desirable to express the statements like *every year between Thanksgiving and Christmas*, which

¹ $\text{Months} = ([0, 31), \dots, [1430, 1461)) \{1461\}_{\text{Days}}$

might specify the time with the highest business activity in a business application, as a single interval at symbolic level as in [Thanksgiving, Christmas] with Thanksgiving, and Christmas representing the two periodic events involved. We introduce a new construct called *periodical interval* which is the natural extension of aperiodic interval construct to periodic elements:

Definition 7 (Periodic Interval): Let $\alpha = N_\alpha \cdot P_\alpha(p)$ and $\beta = N_\beta \cdot P_\beta(p)$ be two calendric events with the same number of intervals in their aperiodic part N_α , N_β and periodic parts P_α , P_β . Then the periodical interval $[\alpha, \beta]$ is a calendric event whose k th interval is constructed by picking the beginning time from the k th interval in event α and the end time from the k th interval in event β :

$$[\alpha, \beta] = \bigcup_{i=1}^{\infty} [\text{begin}(\text{interval}(\alpha, i)), \text{end}(\text{interval}(\beta, i))]$$

where $\text{begin}(\text{interval}(\alpha, i)) \leq \text{end}(\text{interval}(\beta, i))$ for $i > 0$.

The formula above evaluates to a single periodic element because of the fact that periodic elements are closed under set union. Hence periodic elements are closed under periodic intervals. Note that the period of α and β need not be a multiple of each other. Open ended periodic intervals are defined in similar fashion where open ends (and) of periodical intervals don't cover the corresponding intervals from events α and β . For instantaneous events $\alpha = t_1$ and $\beta = t_2$, the periodic interval $[\alpha, \beta]$ reduces to regular (aperiodic) interval $[t_1, t_2]$. For aperiodic events α and β where each event is a temporal element, the periodic interval $[\alpha, \beta]$ evaluates to another temporal element. For example, for aperiodic events $\alpha = [5, 9], [12, 16], [20, 25], [30, 35]$ and $\beta = [7, 12], [13, 16], [23, 32], [31, 40]$, the periodic interval evaluates to $[\alpha, \beta] = [5, 12], [12, 16], [20, 32], [30, 40]$ which can be simplified to $[5, 16], [20, 40]$. Consider the following example for the application of periodic intervals.

Example 7: Let Thanksgivings = interval^{Years} (Novembers \square Thursdays, 4) and Christmases (Example 6) be two calendric events denoting the set of all Thanksgivings (the fourth Thursday of Novembers) and Christmases. Then [Thanksgiving, Christmas] is a calendric event representing the time between Thanksgivings and Christmases every year including the day of Christmas and Thanksgiving.

We believe that using the familiar interval construct $[b, e]$ to express events such as the above is a novel approach which can produce user-friendly query languages in which queries are expressed more naturally. To our knowledge only our approach has such a construct.

5. TEMPORAL RELATIONSHIPS BETWEEN CALENDRIC EVENTS

Temporal relationships between intervals are studied in detail [Al 83] and are adopted by temporal model for aperiodic data in [Sn 93] and others. Similar relationships exist between calendric events. We briefly present these temporal relationships operators. The following are relationships are defined for given calendric events α_c and β_D (we omit the calendars C and D from representation since conversions are performed implicitly. See Section 7.1.1):

- α before $\beta = \text{true}$ if $\text{end}(\alpha)$ before $\text{begin}(\beta) = \text{true}$
- α starts $\beta = \text{true}$ if $\text{begin}(\alpha) = \text{begin}(\beta)$.
- α finishes $\beta = \text{true}$ if $\text{end}(\alpha) = \text{end}(\beta)$.
- α contains $\beta = \text{true}$ if $\alpha \setminus \beta \neq \emptyset$.
- α meets $\beta = \text{true}$ if $\text{end}(\alpha) + 1 = \text{begin}(\beta)$ (for discrete time.)
- α overlap $\beta = \text{true}$ if $\alpha \cap \beta \neq \emptyset$.
- α equals $\beta = \text{true}$ if $\alpha \setminus \beta = \beta \setminus \alpha = \emptyset$.

The functions *begin* and *end* return the first and the last instant of a periodic element (for strictly and partially periodic events function *end* returns ∞). The inverse relationship operators (*after*, *startedby*, *finishedby*, *containedby*, *metby*, *overlappedby*) are defined likewise. These operators reduce to their aperiodic counterparts for aperiodic operands.

6. MODELING RELATIVE EVENTS

Relative events are planned actions and designs for future such as the ordering of tasks in a production line, the presentation order of a set of multimedia streams, or a set of synchronized transactions in a real-time system. Relative events (*temporal templates*) can be expressed similarly to absolute events using a special kind of periodic element called *relative periodic element*. Relative events are defined with respect to a *virtual reference point* (vrp), i.e., the unknown beginning of the event

denoted by the \circ symbol. The rest of the event is specified with respect to *vrp*.

Example 8: A week is a duration of 7 days:

$$[\circ, 7]\{7\}_{D_{day}}$$

The duration of a task performed in 3 parts can be given as a *relative temporal element*:

$$([\circ, 5], [10, 15], [20, 30]).$$

A multimedia presentation consists of synchronized multimedia streams such as audio, video, graphics etc. The playout of video sequences with 24 and 12 frames per second with a duration of 1/48 seconds can be stated (assuming the clock ticks 48 times a second for simplicity) as

$$\alpha = [\circ, 1]\{2\} \text{ and } \beta = [\circ, 1]\{4\}.$$

Temporal transformation becomes necessary when planned events are to be aligned with respect to one another. We present these operators informally here since they are defined similar to absolute transformations.

Example 9. Consider the video streams above. If the user wants to start the second video stream (β) 10 seconds (480 ticks) after the start of the first stream, this could be done by translating the beginning of β by 10 second with respect to *vrp* of the presentation.

$$\text{translate}(\beta, 10) = (\circ, \circ) : [480, 481]\{4\}.$$

Note that (\circ, \circ) , or simply \circ , serves as a reference point only and is not part of the event. If sequences are to be played faster or slower, the events should be scaled. To slow down the playout of the first video stream, the user can scale up the event by a factor of two:

$$\text{scale}(\alpha, 2) = [\circ, 2]\{4\}.$$

At design stage events are defined with no absolute time references providing flexibility to user. However at the actual happening time, the plans should be converted to absolute time so that they can be recorded in the database. Functions *anchor* and *unanchor* convert between relative (templates,

plans, etc.) and absolute (the planned event actually taking place) events:

$$\text{unanchor}([\![t_0, t_1], \dots, [t_{k-1}, t_k] : [l_0, l_1], \dots, [l_{n-1}, l_n]\{p\}\]) = [\![t_1 - t_0, \dots, [t_{k-1} - t_0, t_k - t_0] : [l_0 - t_0, l_1 - t_0], \dots, [l_{n-1} - t_0, l_n - t_0]\{p\}\].$$

$$\text{anchor}([\![a_1], \dots, [a_{k-1}, a_k] : [p_0, p_1], \dots, [p_{n-1}, p_n]\{p\}, t\]) = [t, a_1 + t], \dots, [a_{k-1} + t, a_k + t] : [p_0 + t, p_1 + t], \dots, [p_{n-1} + t, p_n + t]\{p\}.$$

We illustrate the relative-to-absolute conversion using the multimedia sequences above (The event in a multimedia presentation can be stated with respect to different calendars such as the calendar of 24 frames per second for video, the calendar of 12 frames for graphics, etc.)

Example 10: The actual presentation of the video sequences α and β at instant 100 results in the following absolute multimedia events α' and β'

$$\alpha' = \text{anchor}([\circ, 1]\{2\}, 100) = [100, 101]\{2\}.$$

$$\beta' = \text{anchor}([\circ, 2]\{4\}, 100) = [100, 102]\{4\}.$$

The set theoretic operators (union, intersection, and difference) and the relational operators (before, after, overlaps, contains, equals, ...) naturally extend to relative events, not included here for space limitations [K 95]. The basic idea is that the events are aligned at the *vrp*.

7. CALENDRIC EXPRESSIONS AND OPTIMIZATION

Calendric expressions are built from calendric events and calendric operators including the *set theoretic operators* \cup , \cap , \setminus , \supseteq , the *periodical interval operators* $[]$, $[[$, $[)$, $[)$, and the *periodical interval selection operator* *interval*. This algebra is closed under these operators [K 95]. Calendric expressions can be defined recursively as follows:

1. *Choronon* = $[0, 1]\{1\}$ (the physical time, i.e., clock ticks) is a calendric expression.
2. Every basic or arbitrary calendar C is a calendric expression.
3. Every calendric event α , or α_C is a calendric expression (If C is omitted in α_C it is assumed to be *Choronon*).

4. Let α and β be calendric expressions, C and D basic calendars. Then $\alpha_C \sqcap \beta_D$, $\alpha_C \cup \beta_D$, $\alpha_C \setminus \beta_D$, $\simeq \alpha_C$, $\text{interval}^a(\beta_D, i)$, $[\alpha_C, \beta_D]$, $[\alpha_C, \beta_D]$, $[\alpha_C, \beta_D]$, $[\alpha_C, \beta_D]$ are calendric expressions.

Optimization is an important part of query evaluation in temporal query languages. Calendric expressions can be optimized for efficient query evaluation. Optimization can be performed at two different places. (i) Choosing a set of simplification rules for a given expression (ii) choosing a calendar to perform an operation between two events stated with respect to different calendars. These issues are discussed below.

7.1.1 Calendar Conversion

When events α_C and β_D in basic calendars C and D are involved in a calendric expression like $\alpha_C \sqcap \beta_D$, a conversion of α_C and β_D to another calendar E must be performed first, otherwise the intersection will produce an incorrect result. The easiest way to do the intersection is to convert the events α_C and β_D to *Chronon* calendar first, then to perform the intersection. The result can then be converted to a desired calendar. However a more efficient evaluation can be obtained by choosing a calendar E whose period is the greatest common divisor (*gcd*) of the periods of C and D , that is,

$$\text{period}(E) = \text{gcd}(\text{period}(C), \text{period}(D)).$$

The *period* function was defined in Section 3. This rule is called the *calendar optimization rule*. For instance, for calendars *Weeks* and *Days*, the rule chooses *Days* as the calendar to perform operations, because $\text{period}(\text{Days}) = \text{gcd}(\text{period}(\text{Days}), \text{period}(\text{Weeks}))$.

7.1.2 Optimization of Periodic Expressions

We use a set of simplification rules for efficient query processing. Some of these rules are given below. These rules along with the calendar

optimization rule above can be incorporated into a parser for calendric expressions. Let \mathcal{P} be the set of all periodic elements. Let d, e, s , and t be integers. Let α, β , and γ be calendric expressions. Then for every α, β , and γ in \mathcal{P} ,

1. (Distributive Law) $\text{translate}(\alpha \cup \beta, d) = \text{translate}(\alpha, d) \cup \text{translate}(\beta, d)$.
2. (Distributive Law) $\text{translate}(\alpha \sqcap \beta, d) = \text{translate}(\alpha, d) \sqcap \text{translate}(\beta, d)$.
3. (Distributive Law) $\text{scale}(\alpha \cup \beta, d) = \text{scale}(\alpha, d) \cup \text{scale}(\beta, d)$.
4. (Distributive Law) $\text{scale}(\alpha \sqcap \beta, d) = \text{scale}(\alpha, d) \sqcap \text{scale}(\beta, d)$.
5. (Idempotence) For every α in \mathcal{P} :
 $\alpha \cup \alpha = \alpha$ and $\alpha \sqcap \alpha = \alpha$
6. T and \emptyset are distinct and also
 $\simeq T = \emptyset$ and $\simeq \emptyset = T$.
7. (Distributive Law)
 $\alpha \sqcap (\beta \cup \gamma) = (\alpha \sqcap \beta) \cup (\alpha \sqcap \gamma)$ and
 $\alpha \cup (\beta \sqcap \gamma) = (\alpha \cup \beta) \sqcap (\alpha \cup \gamma)$.
8. (De Morgan's Law)
 $\simeq (\alpha \sqcap \beta) = (\simeq \alpha) \cup (\simeq \beta)$ and
 $\simeq (\alpha \cup \beta) = (\simeq \alpha) \sqcap (\simeq \beta)$.
9. (Associativity)
 $\alpha \sqcap (\beta \sqcap \gamma) = (\alpha \sqcap \beta) \sqcap \gamma$ and
 $\alpha \cup (\beta \cup \gamma) = (\alpha \cup \beta) \cup \gamma$.

The correctness of above rules follows from the definitions and can be found in [K 95].

8. RELATED WORK

[So 93, Te 93, WaLa 93] mainly deal with aperiodic calendric events. We briefly compare our work with collection of intervals [Le 86, Ch 94], generalized databases [Ka 90, NiSt 92] which exclusively deal with periodic time. Note that only our approach has the ability (i) to represent partially-periodic events as a single temporal value, (ii) to model relative events as well as absolute events uniformly, (iii) to express events involving periodic intervals at symbolic level using the periodic interval construct.

Table 1: Collection of intervals versus periodic elements.

Event	Collection of intervals	Periodic Elements
Mondays	2/Days:during:Weeks	interval ^{Weeks} (Days, 2)
Januaries	1/Months:during:Years	firstinterval ^{Years} (Months)
First day of every month	1/Days:during:Months	firstinterval ^{Months} (Months ∩ Days)
First Monday in January 86	1/Mondays:during:Januaries:during:1986/Years	firstinterval (Mondays ∩ Januaries ∩ interval(Years, 1986))
Every year between Thanksgiving and Christmas	?	[Thanksgiving, Christmas]

(11, 17), (18, 24), (25, 31))

8.1 Collection of Intervals

A collection of intervals [Le 86] is a hierarchical list structure to represent events. For example, the collection of months where each month is represented by a collection of the days in that month in order is an order 2 collection. Suppose *Weeks* is a calendar of the weeks in 1993.

Weeks = {(-4, 3), (4, 10), (11, 17), (18, 24), (25, 31), (32, 38), (39, 45), ... }

Two types second order operators are defined on collections: *slicing* and *dicing*. The *strict dicing* takes a collection *C*, an interval *t* and a relational interval operator *R* such as overlaps, during, etc. and breaks up *t* into pieces according to *C*.

$$C \cdot R \cdot t = \{c \cap t \mid c \in C \text{ and } c R t = \text{true}\}$$

For example, *Weeks.overlap:<January-93>* breaks the *<January-93>* interval on the week boundaries, i.e., it will give the whole and partial weeks during *<January-93>* interval. Assume that *<January-93>* is represented by interval (1, 31). Then

$$\{Weeks.overlap:<January-93>\} = \{(1, 3), (4, 10),$$

The *selection operators*, *f/C* and $\{f_1, f_2, \dots, f_n\}/C$, applies the selection functions *f* (which could be an integer or *n* or $-n$) to the collection *C* and returns the corresponding intervals or collections from *C*. *n/C* and $-n/C$ select the *n*th interval from the beginning and end of *C* respectively. Collection of intervals are implemented in [Ch 94]. A set of examples on calendar expressions in collection of calendars and periodic elements are given in Table 1. A question mark indicates that we could not express the given statement in the formalism.

8.2 Generalized Databases

A framework for handling possibly periodic time based on constraints is proposed in [Ka 90]. Temporal data is represented by a generalized tuple consisting of a set of temporal attributes, a set of time stamps, and a set of constraints. Each time stamp is represented by a linear repeating point (*lrp*). A *lrp* is the set of time instants computed by the formula $a+kn$ where *a* and *k* are integer constants and *n* takes integer values between $-\infty$ and ∞ . For example, $2+3n = \{\dots, -4, -1, 2, 5, 8, \dots\}$. Then two linear repeating points, one for the beginning time and the other for end time, represent an infinite union of the intervals obtained from the

Table 2: Slices versus periodic elements.

Event	Slices	Periodic Elements
Sundays	weeks+1.days	interval ^{Weeks} (Days, 1)
The last day of every month	?	lastinterval ^{Months} (Days)
The 3rd day of every month	months+3.days	$x = \text{interval}^{\text{Months}}(\text{Months} \cap \text{Days}, 3)$
The 4th hour of the 3rd day of each month	months+3.days+4.hours	interval ^{Days} ($x \cap \text{Hours}, 4$)
Every year between Thanksgiving and Christmas	?	[Thanksgiving, Christmas]

two *lrps* satisfying a set of constraints. Consider the following for a generalized tuple from [Ka 90]:

$$[1, 1+2n], X_2 \geq 0$$

where $X_1=1$ and $X_2=1+2n$. This tuple represents the following infinite set of intervals: $\{[1, 1], [1, 3], [1, 5], \dots\} = [1, \infty)$. The generalized tuples are defined for relational model using tuple timestamping. Use of *lrps* and constraints makes it difficult to use. Niezette and Steven [NiSt 92] try to overcome this shortcoming by using *linear repeating intervals (lri)* in place of *lrps*. A *lri* is given by the following formula: $kn+(b, e)$ where k, b , and e are integers. b and e denotes the beginning and end time of fixed length intervals.

Every *lrp* and *lri* can be given as single periodic element as follows: $a+kn \equiv a\{k\}$ and $kn+(b, e) \equiv (b, e)\{k\}$. (The \equiv symbol states that the formulas on both sides of the symbol are equivalent. The set of constraints can be translated to set operators on periodic elements.) However the opposite is not true. For example, periodic element $[t_1, t_2], [t_3, t_4], \dots [t_{k-1}, t_k] : [t_{k-1}, t_{k-2}], \dots [t_{n-1}, t_n]\{p\}$ with each interval is of distinct length can be represented only by a set of *lpis* (because an *lpi* are a single repeating fixed-length interval):

$$\begin{aligned} n^+ [t_1, t_2] \\ \dots \\ n^+ [t_{k-1}, t_k] \\ np^+ [t_{k-1}, t_{k-2}] \\ \dots \\ np^+ [t_{n-1}, t_n] \end{aligned}$$

A calendar may corresponds to more than one generalized tuples each with one *lri* [NiSt 92] causing data duplication. Authors propose the *slice* $P \triangleright D$ operator to choose intervals from calendars where $P = O_1C_1 + \dots + O_nC_n$ is the starting points of intervals from a set of calendars C_1, \dots, C_m and $D = N.C$ is a duration of N intervals in calendar C . We

informally compare slices and periodic elements in Table 2.

9. CONCLUSION

We introduced a new temporal type called periodic elements as an extension of temporal elements for modeling periodic time in general, and calendars and calendric events in particular where calendars and calendric events are treated as special types of periodic events. Periodic elements are capable of representing aperiodic, strictly-periodic and partially-periodic events as a single value without using formulas. Calendric events are periodic events stated with respect to a calendar. We extended the temporal transformations functions, set theoretic operators, and temporal relational operators of aperiodic types to calendric events to keep the model simple and user-friendly. Absolute and relative events are treated uniformly, i.e., temporal transformations, set theoretic operators, and temporal relational operators apply to relative events similarly allowing us to model multimedia streams. Our calendar algebra on periodic elements are closed for absolute and relative periodic events with conversions between the two. We extended the interval construct $[b, e]$ for calendric events to express complex calendric statements. The periodical interval selection function selects intervals from periodic calendric events according to another event. To our knowledge these two constructs are supported only in our model (the selection operators in [Le 86, Ch 94, NiSt 92] operates only on strictly-periodic or aperiodic events).

Periodic elements can be used to model temporal databases involving periodic elements in a simple manner [KuMe 95]. Consider the relation in Table 3 defined in a similar fashion to relations in [Ga 88]. In the top row, $[91/1, now] \rightarrow \1000 states that Mary's salary is \$1000 between Jan. 91 and *now*. The first two tuples in this relation represent aperiodic events. The third tuple represents a

Table 3: The Employee class/relation with periodic elements.

NAME	SALARY	TASK
$[91/1, now] \rightarrow$ Mary	$[91/1, now] \rightarrow \1000	$[91/1, now] \rightarrow$ Toy
$[91/1, 97/1] \rightarrow$ Jack	$[91/1, 95/1] \rightarrow \2000 $[95/2, 97/1] \rightarrow \2300	$[91/1, 94/3] \rightarrow$ Shoe $[94/4, 97/1] \rightarrow$ Toy
$[92/1, \infty) \rightarrow$ John	$[92/1, 92/6]\{1year\} \rightarrow \1100 $[92/7, 2/12]\{1year\} \rightarrow \1200	$[92/1, 92/3]\{6months\} \rightarrow$ Toy $[92/4, 92/6]\{6months\} \rightarrow$ Shoe

periodic event. John is paid \$1100 and \$1200 a month alternatively for six months starting on January 1992. He works in the toy and shoe departments for 3 months alternatively starting from January 1992.

The user can specify a calendric event with respect to any calendar. Calendric events defined with respect to different calendars can be mixed in a calendric expression. We identified some key optimization rules for efficient query evaluation. We also pointed out through examples that periodic elements could be used for temporal databases, multimedia databases, active databases involving periodic temporal events as well. The operators defined on periodic elements have clear semantics. We are planning to implement the periodic elements as a class hierarchy in c++. Based on these classes, the calendar specific operations can be implemented.

References

- [Al 83] J. F. Allen. *Maintaining Knowledge about Temporal Intervals*. *Comm. of ACM*, 26(11):832-843, Nov. 1983.
- [B 91] M. Baudinet, M. Niezette, and P. Wolper, *On the Representation of Infinite Temporal Data and Queries*, *ACM PODS*, 1991.
- [B 93] M. Baudinet, J. Chomicky, and P. Wolper. *Temporal Databases Beyond: Beyond Finite Extensions*. *Proceedings of the ARPA/NFS Int. Workshop on an Infrastructure for Temporal Databases*. Arlington, Texas, June 14-16, 1993.
- [Ch 94] R. Chandra, A. Segev, and M. Stonebraker, *Implementing Calendars and Temporal Rules in Next Generation Databases*, *Proceedings of the Third Int. Conf. on Data Eng*, IEEE 1994.
- [ClCr 87] J. Clifford and A. Crocker, *The Historical Relational Data Model and Algebra Based on Lifespans*, *Proceedings of the 3rd Int. Conf. on Data Eng*, pp. 528-537, Los Angeles, Feb 87.
- [ChIm 88] J. Chomicky, and T. Imielinsky, *Temporal Deductive Databases and Infinite Objects*, *ACM PODS* 1988.
- [Da 88] C. J. Date, *A Proposal for Adding Date and Time Support to SQL*, *SIGMOD Record*, 17, no. 2, June 1988, pp 53-76.
- [Ga 88] S. K. Gadia, *A Homogenous Relational Model and Query Language for Temporal Databases*. *ACM Trans. on Databases Systems*, 13, No 4, Dec 1988, pp 418-448.
- [Ka 90] F. Kabanza, J. M. Stevenne, P. Wolper, *Handling Infinite Temporal Data*, *ACM PODS*, 1990.
- [K 95] A. Kurt, *Modeling Periodic Temporal Databases with Periodic Elements*, Technical Report (in preparation), 1995.
- [KuMe 95] Kurt, A. And Ozsoyoglu, M. *Modeling and Querying Periodic Temporal Databases*, *DEXA Workshop* 1995.
- [Le 86] B. Leban, D. D. McDonald, and D. R. Forster. *A Representation for Collections of Temporal Intervals*. *AAAI* 1986 pp. 367-371.
- [Me 90] J. Melton, (ed) *Solicitation of Comments: Database Language SQL2*, ANSI, Washington, DC. 1990.
- [NaAh 87] S. B. Navathe, R. Ahmed. *TSQL-A language interface for history database*. *Proceeding of The Conf. on The Temporal Aspects in Info. Systems*, pp. 113-128 France May 87. AFCET North-Holland.
- [NiSt 82] M. Niezette, J. M. Stevenne, *An Efficient Symbolic Representation of Periodic Time*, *CIKM* 1992, pp 161-168.
- [So 93] M. Soo. *Multiple Calender Support for Conventional Database Management Systems*. *Proceedings of the ARPA/NFS Int. Workshop on an Infrastructure for Temporal Databases*, Arlington TX, Jun. 14-16, 1993.
- [Sn 93] R. Snodgras, *An overview of Tquel*, in A. Tansel et al editors, *Temporal Databases, Theory, Design and Implementation*, Benjamin/Cummings 1993
- [Waja 93] X. S. Wang, S. Jajodia, and V.S. Subrahmanian, *Temporal Mediators as a way to Support Multiple Temporal Representations*, *Proceedings of the ARPA/NFS Int. Workshop on an Infrastructure for Temporal Databases*. Arlington Texas June 14-16, 1993.
- [Te 93] T. Lawson, C. Bathhazaar, A. Gray, *An Object-Oriented Approach to Temporal Modeling*, *Proceedings of the ARPA/NFS Int. Workshop on an Infrastructure for Temporal Databases*. Arlington Texas June 14-16, 1993.