# Temporal Queries in OQL

Atakan Kurt*      Cemil Öz**

*Department of Computer Eng.Fatih University, Istanbul, Turkey
**Department of Computer Eng. Sakarya University, Sakarya, Turkey

## Abstract

The issue of modeling and querying such temporal events has been the subject of research the database community in recent years. We present an formalism based on a new periodic temporal type called *periodic element* to model periodic events in temporal databases, and illustrate its suitability by extending the object-oriented query language QQL of $O_2$ with a temporal data type hierarchy with an implementation. The temporal extension to the language is achieved through a set of absolute and relative periodic types which have well-defined temporal operators/functions written in the form of methods. Periodic elements are capable of representing *aperiodic*, *strictly-periodic,* and *partially-periodic* absolute and relative events, and can be used for implementing calendars, scheduling and other temporal applications.

**Keywords**: Representation of time, temporal databases, object-oriented databases, QQL, SQL, temporal query languages, periodic events, aperiodic events, absolute events, relative events, scheduling.

Introduction

Applications such as scheduling, planning, forecasting, time-management, time-series, scientific, multimedia, real-time, and active databases, banking, law, medical records, accounting, process control, inventory control, geographical information systems may deal with periodic events in addition to aperiodic events. The following example illustrates the need for modeling periodic events.

*Example 1*: Consider a company employing full-time and part-time employees. The work hours of full-time employees can be maintained by a conventional temporal database system, since they can be represented by intervals [1], or temporal elements [2]. The work hours of part-time employees can't be stored in a conventional temporal database, because employees have (periodical) daily, weekly, or monthly schedules. The work hours of employees rotating periodically among departments can't be modeled in conventional temporal databases either. The company also maintains a set of tentative schedules or *temporal templates* for different types of tasks and employees. These templates are work plans with no absolute time references. Each employee works according to a temporal template assigned to him/her when he/she is employed. The management of the company also wants to see the work history of an employee as a whole especially when a part-time employee becomes a full time employee or vice-versa, or when a temporary employee becomes a part-time or full-time employee. In the last case, an initial part of the work history as a temporary employee is an aperiodic event, whereas the second part as a part-time employee is a strictly-periodic event. We call this type of events partially-periodic events.

The rest of the paper is organized as follows. In Section 2, the related work is discussed briefly. A comparison of our work with the related work can be found in [3]. We present *periodic elements* a data type capable of representing periodic events and introduce a closed algebra on this type as well as other temporal operators and functions in Section 3. The underlying object-oriented model and associated temporal data model is introduced in Section 4. In this context, a set of temporal data types used for extending O2 for periodic time and their usage in OQL the query language of O2 are discussed. In Section 5, we show how an object-oriented periodic temporal database can be queried using OQL and periodic elements through a set of examples. Finally the conclusions are given in Section 6.

## 1. RELATED WORK

Even though the schedule of full time employees in the example above can be modeled by aperiodic temporal databases such as [4,5, 6], the work schedule of part-time employees or rotating full-time employees i.e. employees with a periodic schedule can not be represented in these because these models do not consider periodic events. In this case, periodic events must be directly maintained and interpreted by application programs in an ad hoc manner.

Periodic events are modeled in the literature using repeating intervals [7], collection of intervals [8], linear repeating points with constraints [9], linear repeating intervals with constraints [10], temporal mediators [11], or object oriented types [12]. Chandra et al. [13] have improved and implemented the calendar algebra in [8]. This implementation supports periodic events within a finite interval of time. Michael Soo [14] proposes an extension to SQL2 that supports multiple calendars. An object-oriented approach is taken for modeling calendars in Eiffel language in [15]. Periodic events in are also studied in the context of deductive databases as in [16]. Recently more work has been introduced on periodic time: [17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32]

## 2. REPRESENTING PERIODIC TIME

In this sections we present a data type for representing and manipulating temporal events called periodic elements. This type introduces an implicit representation of periodic time with clear semantics which is more user friendlier for the users of existing aperiodic temporal databases than the use of formulas or constraints. By seamless extension we mean that the existing operators and functions defined on aperiodic types can be applied to the values of the new type periodic element without any modifications to the syntax while keeping the a semantics similar to that of aperiodic operator/function..

Periodic elements are formally based temporal elements which are based on intervals which in turn based on instants. The temporal types of instants, intervals and temporal elements are first given below. Then periodic elements and the algebra defined on periodic elements are given. 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 except in the implementation this was not possible because of a restriction in OQL).

### Instants, Intervals and Temporal Elements

There is a set of temporal constants defined as follows: We choose integers to represent clock ticks for the sake of presentation. The constant $0$ is the beginning of the physical time line. *now* is a temporal constant representing the current time or clock tick. The constant $\infty$ is used to represent the positive infinity. $\emptyset$ is 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. Open ends of an intervals are indicated by paranthesis where closed ends are indicated with bracket [ ] symbols.

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

$$N = ([b_1, e_1], [b_2, e_2], ..., [b_k, b_k]) \ for \ k \geq 1.$$

Parenthesis can be omitted, when there is no confusion. 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]. The temporal relationship operators of *before, after, contains, overlaps,* etc. on intervals are extended to temporal elements as well.

### Periodic Elements

Temporal elements are more appropriate than intervals for modeling temporal data and query languages, but temporal elements represent only aperiodic events. We now introduce periodic elements that overcomes this limitation.

*Definition 1 (Periodic element): A triplet of two temporal elements $N = (I_1, I_2, ..., I_n)$ and $P = (J_1, J_2, ..., J_m)$, 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 non-periodic possibly irregular temporal pattern/part of the temporal event being represented,

2. *Periodic part P* is also a temporal element denoting the regular indefinitely-many-repeating part of the temporal event,

3. *Period* is a duration of time between two consecutive occurrence of *periodic part P*.

The condition $p \geq 0$ of definition ensures that the periodic event being represented extends into positive infinity i.e., 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. For example, a negative p can be used to represent an event that can be thought of happening backwards in an application where it might make sense. A periodic event represented by a periodic element happens over 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: $P\{p\} = P \cup translate(P, p) \cup translate(P, 2p) \cup ...$ A periodic element is essentially an infinite union of intervals with the periodic part repeating indefinitely.

If the periodic part $P$ of a periodic element $N{:}P\{p\}$ is an empty element ($P = \varnothing$) 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 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 temporal element which is an aperiodic temporal type seamlessly). If the aperiodic part $N$ of a periodic element $N{:}P\{p\}$ is an empty element ($N = \varnothing$), then the aperiodic part is omitted from the representation and written $P\{p\}$ for short.
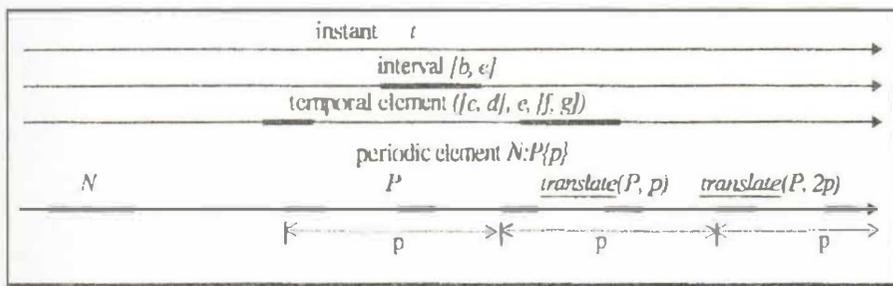


*Figure 1: Aperiodic types and Periodic Element.*

*Example 2:* The followings are valid periodic elements which are written as union of intervals explicitly:

> 5 (an instant),
>
> [3, 7] (an interval),
>
> ([2, 5], [8, 10]) (a temporal element),
>
> [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] ∪ ...)

Functions *aperiodic_part*, *periodic_part* and *period* return the 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* $(N{:}P\{p\}) = P$.

**Set Theoretic operators**

The Boolean connectives *and, or, not* are widely used in applications of periodic time. A good temporal query language should support set theoretic operators *union* ∪, *intersection* ∩, *difference* \, and *complementation* ~.

As with aperiodic events, algebraic set operators are essential in expressing variety of temporal events. Through these operators users can express complex events in terms of simple events. Temporal queries can be formulated with ease using the set theoretic operators because even the most naive users are familiar with the straightforward semantics of the set theoretic operators.

The algebra defined on the intervals is not closed under the set of all intervals which causes data duplication over a set of tuples for a single object. This in turn causes data integrity problems. Temporal elements however are closed under the set theoretic operators. The result of any algebra expression is also a temporal element. On one hand this solves the data duplication and the associated problems in the data model, because an object will be stamped by a single temporal element (itself is a set of intervals) at all times. On the other hand the resulting data model is not in the 1NF anymore. From the ease of expressing queries point of view, the non-1NF model provides user with a more intuitive and natural query language than the 1NF models, because user is given a clearer picture of data.

Periodic elements support all set theoretic operators. To our knowledge, none of the existing periodic types offer a set algebra including all set theoretic operators. Furthermore the set of all periodic elements are closed under set theoretic operator as shown by the following three lemmas. The set theoretic operations on periodic elements are illustrated in Figure 2.2.

In order to develop the set theoretic operators on periodic elements we need to define some function called formatting functions. The formatting functions should not be confused with the temporal transformations even though there is similarity between *translation* and *right/left* functions and between *scaling* and *fold/unfold* functions. The result of a *transformation* of a periodic event is a different event, while the result of *formatting* is the same event expressed differently. We now formally describe these formatting functions.

**Definition 2.2** (Right-shift)

*Given a periodic element $N{:}P\{p\}$ and a duration $d \geq 0$, the function right : $P \times N \rightarrow P$ right-shifts the beginning of the periodic part of $\alpha$ by d time units:*

$$right\ (N{:}P\{p\},\ t) = N \cup (\bigcup_{i=1}^{k} translate\ (P,\ (i{-}1)p)) \cup$$

$$(translate\ (P,\ kp) \cap [begin\ translate\ (P,\ kp)),$$
$$translate\ (begin\ (P),\ d)]) : [translate\ (begin\ (P),$$

d), _translate_ (_begin_ (P), d+p)]$\cap$ (_translate_ (P,
kp) $\cup$ _translate_ (P, (k+1)p)) {p}

where $k = \lfloor d / p \rfloor$.□

### Definition 2.3 (Left-shift)

_Given a periodic element N:P{p} and a duration d (0 $\leq$ d_
_$\leq$ p), the function left: **P** x **N** $\rightarrow$ **P** shifts the beginning of_
_the periodic part P of N:P{p} by d time units to the left._
_Let Q = N$\cap$ [translate (begin (P), –d), begin (P)]. Then_

left (N:P{p}, d) = N $-$ Q : Q $\cup$ (P$\cap$ [_begin_ (P), _translate_
(_begin_ (P), p–n)]){p}

_if Q = translate ((P$\cap$ [translate (begin (P), p-n), end_
_(P)]), p)._□

### Definition 2.4 (Fold)

_Given a periodic element N:P{p} and an integer n > 0,_
_the function fold: **P** x **N** $\rightarrow$ **P** divides the periodic part of_
_$\alpha$ into q equal-length, identical-pattern temporal_
_elements as follows:_

fold (N:P{p}, q) = N : [_begin_ (P), _translate_
(_begin_ (P), p/n)] $\cap$ P {p/n}

_if translate ($Q_i$, (j–i)p/n) = $Q_j$ for 1 $\leq$ i < j $\leq$ n where_

$Q_k$ = [_translate_ (_begin_ (P), (k–1)p/n), _translate_
(_begin_ (P), kp/q)] $\cap$ P. □

### Definition 2.5 (Unfold)

_Given a periodic element N:P{p} and an integer n (> 0),_
_the function unfold replicates the periodic part of $\alpha$ n_
_times as follows:_

unfold (N:P{p}, n) = N : $\bigcup_{i=1}^{n}$ _translate_ (P, (i–1)p){pq}.□

The formatting functions are used to write a periodic
event in different ways. Below we give 3 lemmas used
for proving a theorem stating that periodic elements are
closed under set theoretic operators. The proof of the
following lemma is based on formatting functions.

### Lemma 2.1 (Union)

_The union $\alpha \cup \beta$ of two periodic elements $\alpha$ and $\beta$ is also_
_a periodic element._ □

### Proof

Let $\alpha = N_1:P_1\{p_1\}$ and $\beta = N_2:P_2\{p_2\}$. Then
$$\alpha \cup \beta = N_1 : P_1\{p_1\} \cup N_2 : P_2\{p_2\}.$$

Show that $N_1:P_1\{p_1\} \cup N_2:P_2\{p_2\}$ is a periodic element.
Assume without loss of generality that _begin_ ($P_1$) $\leq$
_begin_ ($P_2$). Let $N_3:P_3\{p_1\}$ = _right_ ($N_1:P_1\{p_1\}$, d) where

$N_3$ = $N_1 \cup \bigcup_{i=1}^{k}$ _translate_ ($P_1$, (i–1)$p_1$) $\cup$(_translate_ ($P_1$,

$kp_1$) $\cap$ [_begin_ _translate_ (P, $kp_1$)), _translate_
(_begin_ ($P_1$), d)]),

$P_3$ = [_translate_ (_begin_ ($P_1$), d), _translate_ (_begin_ ($P_1$),
d+$p_1$)]$\cap$ (_translate_ (P, $kp_1$) $\cup$_translate_ ($P_1$,
(k+1)$p_1$))

where

$k = \lfloor d / p_1 \rfloor$ and d = **distance** (_begin_ ($P_2$), _begin_ ($P_1$))
Replace $N_1:P_1\{p_1\}$ with $N_3:P_3\{p_1\}$, since $N_1:P_1\{p_1\}$ =
$N_3:P_3\{p_1\}$ by the right shift rule. Then
$$\alpha \cup \beta = N_3 : P_3\{p_1\} \cup N_2 : P_2\{p_2\}.$$
Let $N_3 : P_4 \{lcm (p_1, p_2)\}$ = _unfold_ ($N_3 : P_3 \{p_1\}$, lcm
($p_1, p_2$)/$p_1$) where

$P_4$ = $\bigcup_{i=1}^{n}$ _translate_ ($P_3$, (i–1)$p_1$) and n = lcm ($p_1, p_2$)/$p_1$

Since $N_3 : P_3 \{p_1\} = N_3 : P_4 \{lcm (p_1, p_2)\}$
replace $N_3 : P_3 \{p_1\}$ with $N_3 : P_4 \{lcm (p_1, p_2)\}$.
Similarly let $N_2 : P_5 \{lcm (p_1, p_2)\}$ = _unfold_ ($N_2 : P_2$
$\{p_2\}$, lcm ($p_1, p_2$)/$p_2$)

where $P_5$ = $\bigcup_{i=1}^{n}$ _translate_ ($P_2$, (i–1)$p_2$) and n = lcm ($p_1$,

$p_2$)/$p_1$.

Since $N_2 : P_2 \{p_2\} = N_2 : P_5 \{lcm (p_1,p_2)\}$
replace $N_2 : P_2 \{p_2\}$ with $N_2 : P_5 \{lcm (p_1, p_2)\}$ (Note
that any common multiplier of $p_1$ and $p_2$ would suffice
for proving the lemma, however lcm ($p_1, p_2$) is chosen to
minimize redundancy.):
$$\alpha \cup \beta = N_3 : P_4\{lcm (p_1,p_2)\} \cup N_2 : P_5\{lcm (p_1,p_2)\}$$
By definition
$\alpha \cup \beta = N_3 \cup P_4 \cup$_translate_ ($P_4$, lcm ($p_1, p_2$)) $\cup$
_translate_ ($P_4$, 2lcm ($p_1, p_2$)) $\cup$... $\cup N_2 \cup P_5 \cup$_translate_
($P_5$, lcm ($p_1, p_2$)) $\cup$_translate_ ($P_5$, 2lcm ($p_1, p_2$)) $\cup$...
After regrouping the terms,
$\alpha \cup \beta = N_3 \cup N_2 \cup P_4 \cup P_5 \cup$_translate_ ($P_4$, lcm
($p_1,p_2$)) $\cup$_translate_ ($P_5$, lcm ($p_1,p_2$)) $\cup$_translate_ ($P_4$,
2lcm ($p_1,p_2$)) $\cup$_translate_ ($P_5$, 2lcm ($p_1,p_2$)) $\cup$...
Hence,
$\alpha \cup \beta = N_3 \cup N_2 : P_4 \cup P_5 \{lcm (p_1, p_2)\}.$□

Next two lemmas can be proved similarly.

## Lemma 2.2 (Intersection)

*The intersection $\alpha \cap \beta$ of two periodic elements $\alpha$ and $\beta$ is also a periodic element.*□

## Lemma 2.3 *(Complementation)*

*The complement ~ $\alpha$ of a periodic element $\alpha$ is a periodic element.*□

## Theorem 2.1

*The set of all periodic elements $\textbf{P}$ is closed under set theoretic operators union ($\cup$), intersection ($\cap$), and complement (~).* □

## Proof

The proof follows from Lemma 2.1, Next two lemmas can be proved similarly.

**Lemma 2.2**, and Lemma 2.3. □

The closure property also holds true between aperiodic and periodic, and aperiodic and aperiodic events, since aperiodic events are defined as a special case of periodic events.

## Example 2.3

Let $\alpha = [0, 8] : [24, 26] \{12\}$ and $\beta = [5, 12] : [24, 27] \{8\}$

$\alpha \cup \beta = [0, 12] : [24, 27, [32, 35], [36, 38], [40, 43] \{24\}.$

$\sim (\alpha \cup \beta) = (12, 24), (27, 32): (35, 36), (38, 40), (43, 48)\{24\}.$

□

Let *pp* and *app* stand for *periodic_part* and *aperiodic_part* functions. Then the union and intersection operators are algorithmically defined as follows:

## Definition 2.6 (Union Algorithm)

Let $\alpha = M : P \{p\}$ and $\beta = N : Q \{q\}$ be 2 periodic elements. Assuming <u>begin</u> (P) *before* <u>begin</u> (Q) = true, the union $\alpha \cup \beta$ is computed by the following formula:

$\alpha \cup \beta = app$ (right ($\alpha$, **distance** (<u>begin</u> (P), <u>begin</u> (Q)))) $\cup N :$

pp (unfold (right ($\alpha$, **distance** (<u>begin</u> (P), <u>begin</u> (Q))), lcm(p, q)/p)) $\cup$

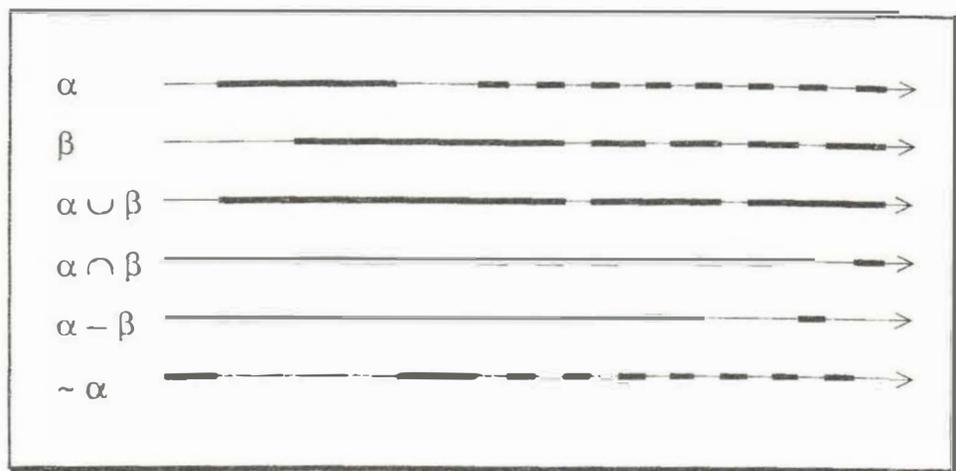pp (unfold ($\beta$, lcm(p, q)/q))) {lcm (p, q)}.□



*Figure 2.2: The set theoretic operators*

The correctness of this definition follows from Lemma 2.1. When $\alpha$ and $\beta$ are temporal elements, the union $\cup$ on periodic elements reduces to the union $\underline{\cup}$ on elements. Similarly if $\alpha$ and $\beta$ are intervals, then the union $\underline{\cup}$ on elements reduces the union $\underline{\cup}$ of intervals. The intersection of periodic elements is defined similarly to union.

## Temporal Transformations

When defining a periodic events such as periodic or multimedia events in terms of other periodic events, temporal transformations are utilized quite often. The usual transformation functions *translate*, and *scale* on periodic elements are defined

## Temporal Relationship Operators

The relationships between temporal event represented by periodic elements can be captured by a set of operators such as before, after, contains, starts etc.. The semantics and the syntax of these operators follows from their counterparts defined on temporal element and intervals and omitted here.

## 3. PERIODIC OBJECT-ORIENTED TEMPORAL DATA MODEL

Many Object-Oriented Temporal Models have been proposed in recent years such as [33 34 35 36 37].Temporal events can be modeled by associating time with either objects or by attributes called *attribute stamping* and *tuple stamping* respectively [2]. Tuple stamping is achieved by introducing one or more attributes in a relation/class to represent the valid time. Attribute stamping associates time-stamps with each value, thus resulting in a non-first normal form relation. The valid time of a complex object is recursively derived from its constituent objects.

The history of an attribute is stored as a function of time from the attribute domain into the time domain

(The meaning of the term *history* here extends into the future in the case of periodic elements). A function may be stored or computed (a procedure) function which is invisible to users. In an implementation, the history can be stored as a set of mappings as in *Salary = ([89/1/1, 95/5/1] → $2000, [95/6/1, 96/7/1] → $2100)* for a salary attribute. *Lifespan* of an attribute or object is the time over which it exists or happens. The polymorphic *Vtime* function computes the lifespan of an object/attribute by unioning the periodic elements in the temporal mapping of the object/attribute. For example, the lifespan is *Vtime(Salary) = [89/1/1, 96/7/1]* for the salary attribute. The lifespan of a complex object (defined through *tuple*, *set*, *list*, *multi-set* constructs) is the union of the lifespans of its constituent objects/attributes defined recursively. Since a class is a set of objects, the lifespan of a class is derived from the union of the lifespans of the objects in the class. The lifespan of a superclass from its subclasses can be derived similarly.

## 4. QUERIES IN OQL

We introduce the periodic temporal queries in the query language OQL of $O_2$. (Temporal databases can also be directly queried within an application through C, C++ and $O_2$C interfaces to $O_2$) The queries are expressed using the *Select* command similar to SQL Select. The Select command in $O_2$ is more powerful than the Select in SQL. For example, nested subqueries can be placed anywhere in the *select*, *from*, or *where* clauses as opposed to only in the *where* clause of SQL Select. The $O_2$ query processor is also capable of evaluating any valid expressions involving object-ids, mathematical formulas, function calls, etc.

Consider the employee database in Section 1. We define an employee class for storing employee data where the birth date is a non-temporal attribute, name and salary are defined as temporal attributes. The name attribute could have been also defined as static attribute in which case the changes to employee names cannot be stored and queried. Below are some queries given in OQL, more queries can be found in [3].

```
class emp inherit t_object
public type
    tuple (birthdate: Date,
            t_empname: list (tuple (time: P_element,
value: string)),
            t_salary: list (tuple (time: P_element, value:
integer)),
            ...)
method
        public empname (t: Instant): string,
```

```
        public    set_empname    (list    (tuple
(time:P_element, value: string))): boolean,
        public display_empname: string,
        public salary (t: Instant): string,
        public    set_salary    (list    (tuple
(time:P_element, value: string))): boolean,
        public display_salary: string,
        public vtime: P_element,
        public t_display (i: Instant): string,
        . . .
end;
```

1. What was John's pay 2 weeks after he started working?

*select    e->salary    (instant_translate    (begin (vtime_integer (e->t_salary)), 14))*

*from    e in emps*

*where    e->empname (now())="john"*

The t_salary is the temporal extension of the salary attribute, i.e., it is a data structure containing the salary values over the past present and the future. The *vtime_attrib_type* (vtime_integer, vtime_string, etc) function computes the valid time of a temporal attribute of type *attrib_type*. The *begin (vtime_integer (e->t_salary))* computes the starting time of the salary attribute. The *instant_translate (begin (vtime_integer (e->t_salary)), 14)* translates the beginning time by 14 days.

2. How often is John paid ?

*select    period (vtime_integer (e->t_salary))*

*from    e in emps*

*where    e->empname (now())="john"*

*vtime_integer (e->t_salary)* computes the valid time (lifespan) of the salary attribute in the form of a periodic element. The *period* function returns the period of a periodic element.

3. Who are employees who are paid weekly?

*select    e->empname (now())*

*from    e in emps*

*where    period (vtime_integer (e->t_salary)) = 7*

4. Display salary history of employees

*select    e->display_salary()*

*from    e in emps*

The *display_salary* attribute returns the salary over the lifespan of the attribute in text format.

5. Are all employees currently making more than $6 currently?

*for all e in emps: e->salary(now()) >= 6*

A query can be existentially or universally quantified in OQL. This query evaluates to true if all employees make more than $6 currently.

6. Will there be any employee making less than $6 in a year from now?

*exists e in emps: e->salary (instant_translate (now(), 365)) <= 6*

The *instant_translate (now(), 365)* expression computes the temporal expression *a year from now* by translating current time instant (*now()*) by 365 days.

7. What was the maximum salary paid as of 9 days ago?

*max (select e->salary (instant_translate (now(), −9))*
*from    e in emps)*

8. Who are those employees that have been in the company during the time John was in the company?

*define john as*

*intersect (element (select e->vtime() from   e in emps*
*where  e->empname(now()) = "john"),*
*until_now())*
*select f->empname(now()) from   f in emps*
*where  contain (f->vtime(), john)*

9. What time frame in which either John or Mary are working?

*union (element (select e->vtime() from   e in emps*
*where  e->empname(now()) = "john"),*
*element (select e->vtime() from   e in emps*
*where  e->empname(now()) = "tim"))*

## 5. CONCLUSION

A periodic type has to conform to some requirements such as simplicity of representation, ability to represent aperiodic, strictly-periodic or partially-periodic relative and absolute events uniformly. We proposed a temporal type called periodic element meeting these requirements and showed its applicability to temporal databases by extending the OQL with a temporal type hierarchy for periodic temporal data. Our extension defines temporal data as a function time through a set of abstract data types complete with relational comparison operators, set theoretic operators and temporal transformations. Temporal expressions involving periodic elements can be optimized for efficient query processing. Since instants, intervals and temporal elements are periodic elements by definition, the existing temporal models and their query languages based on instants, intervals and temporal elements can be extended for periodic data without changing the syntax and semantics. A set of OQL queries including aggregate functions and valid time on a periodic temporal object-oriented database in O2 is given

for an employee scheduling system. Calendars and different time granularity can be expressed easily with periodic element as well [38].

## References

1  Allen, J. F., *Maintaining Knowledge About Temporal Intervals, Communications of the ACM.* 26(11):832-843, November 1983.

2  Gadia, S., *A Homogeneous Relational Model and Query Languages for Temporal Databases.* ACM Transactions on Databases Systems, 13(4):418-448. December 1988.

3  Kurt, A., Modeling Periodic Time, Periodic Temporal Databases and Calendars, Ph.D. Thesis, CWRU, Cleveland, Ohio, 1997.

4  Gadia, S., et al., A Query Language for Homogeneous Temporal Data. ACM-SIGMOD 1985.

5 Tansel, A., *A Historical Database, Information Sciences,* No 53, pp. 101-133, 1991.

6 Snodgrass, R., et al., *TSQL2 Language Specification, Sigmod Record,* 23(1), pp. 65-86, March 1994.

7  Barbic, F. And Pernici B., Time Modeling in Office Information Systems, Proc. Of ACM-SIGMOD 1985, pp 51-62.

8  Leban, B., McDonald, D. D., and D. R. Forster, A *Representation for Collections of Temporal Intervals, AAAI* pp. 367-371, 1986.

9 Kabanza, F., *Handling Infinite Temporal Data,* Ninth Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 392-403, Nashville, TN, April 1990.

10  M. Niezette and J-M Stevenne, *An Efficient Symbolic Representation of Periodic Time,* Int Conf on Information and Knowledge Management, 1992 pp 161-168.

11 Wang, X. S., and S. Jajodia, *Temporal Mediators as a way to Support Multiple Temporal Representations, Proceedings of the ARPA/NFS International Workshop on an Infrastructure for Temporal Databases,* Arlington, Texas, June 14-16, 1993.

12 Tansel, A., et al., *Temporal Databases: Theory, Design and Implementation.* Benjamin/Cummings Publishing Company, Inc., series on databases systems and applications (book), 1993.

13 Chandra, R., et al., *Implementing Calendars and Temporal Rules in Next Generation Databases, Proceedings of the Third International Conference on Data Engineering,* IEEE 1994.

14 Soo, M., *Multiple Calendar Support for Conventional Database Management* Systems, *Proceedings of the ARPA/NFS International Workshop on an Infrastructure for Temporal Databases.* Arlington Texas June 14-16, 1993.

15 T. Lawson, C. Balthazaar, and A. Gray. *An object-Oriented Approach to Temporal Modelling. Proceedings of the ARPA/NFS Int Workshop on an Infrastructure for Temporal Databases.* Arlington Texas June 14-16, 1993.

[16] Jan Chomicky, Tomasz Imielinsky. *Temporal Deductive Databases and Infinite Objects. ACM Pods* 1988.

[17] Claudio Bettini, Roberto De Sibi, „Symbolic representation of user-defined time granularities, TIME 99

[18] Shubha Chakravarty and Yuval Shahar, "A Constraint-Based Specification of Periodic Patterns in Time-Oriented Data", TIME 99

[19] Lina Khatib and Robert A. Morris, "Generating Scenarios for Periodic Events with Binary Constraints ", TIME99

[20] Isabella Merlo, Elisa Bertino, Elena Ferrari, Giovanna Guerrini, "A Temporal Object-Oriented Data Model with Multiple Granularities", TIME99

[21] Robert A. Morris and Lina Khatib, "Optimization in Constraint Reasoning about Repeating Events", TIME 99

[22] Gerard Becher, Françoise Clérin-Debart, Patrice Enjalbert, "A Model for Time Granularity in Natural Language ", TIME98

[23] Robert A. Morris and Lina Khatib, "Quantitative Structural Temporal Constraints on Repeating Events", TIME 98

[24] Paolo Terenziani, "Generating Instantiations of Contextual Scenarios of Periodic Events", TIME98

[25] Diana Cukierman and James Delgrande, A language to express time intervals and repetition, TIME 95

[26] Robert Morris, Gerard Ligozat, and Lina Khatib, Generating Scenarios from Specifications of Repeating Events TIME 95

[27] Paolo Ternziani Reasoning about Periodic Events TIME 1995

[28] Jerome Euzenat An algebraic approach to granularity in time representation TIME 95

[29] Diana Cukierman and James Delgrande, Characterizing temporal repetition. TIME1996

[30] Edjard Mota & David Robertson , "Representing Interaction of Agents at Different Time Granularities",TIME96

[31] Carlo Combi , Francesco Pinciroli and Giuseppe Pozzii, "Managing Time Granularity of Narrative Clinical Information" TIME1996

[32] Claudio Bettini, "A General Framework and Reasoning Models for Time Granularity " TIME1996

[33] Rose, E., and A. Segev, *TOODM - A temporal Object-oriented Data Model with Temporal Constraints, Proceedings of the 10th International Conference on the Entity Relationship Approach*, October 1991

[34] Su, S. Y. W., and H. M. Chen, *A temporal Knowledge Representation Model OSQM\*/T and its Query Language OQL/T. Proceedings of the Conference on Very Large Databases*, Barcelona, Spain, December 1991

[35] Wuu, G., and U. Dayal, *A Uniform Model for Temporal Object-oriented Databases. Proceedings of the International Conference on Data Engineering*, Arizona, pp. 584-593, February 1992.

[36] Elisa Bertino, Elena Ferrari,Giovanna Guerrini, "An Approach to Model and Query Event-Based Temporal Data", TIME 98

[37] Carlo Combi and Giorgio Cucchi, "GCH-OSQL : A Temporally-Oriented Object-Oriented Query Language based on a three-valued logic" , TIME 97

[38] A Kurt, Z. M. Ozsoyoglu "Modeling Periodic Time and Calendars", The 1995 International Conference on Applications of Databases (ADB 95) 1995, San Jose, California, pp 221-234.