A Temporal Relational Algebra Based on Multiple Time-Lines

Mehmet A. Orgun

Department of Computing, Macquarie University
Sydney, NSW 2109, Australia
mehmet.orgun@mq.edu.au

Abstract

A clocked temporal relational algebra, called $\Re$, which supports temporal relations based on multiple time-lines is proposed. Temporal relations are defined over clocks which are subsequences of an assumed global time-line. The algebra is a consistent extension of the relational algebra, and it includes a number of temporal operators to combine data based on different time-lines. The meaning of an operation of $\Re$ depends on the clocks of the relations involved in the operation as well as the relations. We outline a formal interpretation of expressions of $\Re$, and sketch a naïve expression evaluation method.

1 Introduction

The relational algebra [3] operates on a model in which each relation reflects the current reality as it is best known of the enterprise being modeled. The model cannot deal with the notion of a history, or how each relation has evolved into what it is now in a natural manner. Having recognized the need to incorporate the time dimension into the relational model, a significant number of research efforts have been directed towards studying various aspects of this problem. Included in that effort are temporal extensions of Codd's relational algebra [11]. For more details on temporal databases and temporal query languages, we refer the reader to the literature [16, 2]. Most of the proposed algebras are based on a uniform representation of time (with regards to granularity and scale).

An important and unresolved issue is that the relations in a temporal database are not necessarily defined on the same granularity of time or scale (e.g., days, weeks, or months). Some events occur at irregular intervals, and it seems unnatural to force them all onto a prescribed notion of time. Wiederhold, Jajodia and Litwin [18] recognized the problem, and provided an algebra in which data with multiple granularities are converted to a uniform model of data based on time intervals. Dyreson and Snodgrass [5] extended SQL-92 to support mixed granularities with respect to a granularity lattice. They provided two operations, *scale* and *cast*, that move times within the granularity lattice. Euzenat [6] gives an algebraic approach to granularity in time representation, which uses granularity change operators for converting (upward and downward) qualitative time relationships from one granularity to another.

A related issue is the notion of time-varying relations defined over multiple time-lines, for instance, multiple time series [4] or financial data defined over time-lines with varying rates of sampling. Therefore there is a need for a model of time that can differentiate between different levels of time. Caspi and Halbwachs [1] proposed a model of events in reactive systems, in which there is a global time-line, but it is the actual events that define *time*. Events are time-stamped with their dates of occurrence. Such a model of time requires the synchronization of data based on multiple time-lines. In [8], the maintenance of historical data along multiple lines of time evolution has been extensively discussed, but little attention has been paid to representing and querying historical data based on multiple time-lines.

This paper proposes a clocked temporal algebra, called $\Re$, in order to deal with time-varying relations based on multiple time-lines. In the underlying data model, temporal relations are a collection of ordinary relations, defined over clocks that represent multiple time-lines. The set of possible moments in time is called the global time-line, whose interpretation depends on the application at hand. Our model of time is similar to that of [1, 14] in that the clock of a temporal relation is the collection of time-stamps (dates of occurrence or validity) from its defining events (tuples).

In $\Re$, data values from different moments in time and from different time-lines are combined through the use of temporal operators, not by explicit references to time. The meaning of an operation (temporal or otherwise) depends on the clocks of the relations involved in the operation as well as the relations; however, no new times are produced as a result of an operation. The data model of $\Re$ is based on a temporal logic called TLC [9] in which each predicate symbol is assigned a clocked temporal relation and tempo-

ral operators are used to combine values from different moments in time. At this stage, $\Re$ does not deal with relations based on multiple granularities of time.

There are other proposals for temporal algebras based on temporal logic. Tuzhilin and Clifford [17] proposed a temporal algebra (called **TA**) which offers the operators of the relational algebra plus two temporal linear recursive operators. **TA** is equivalent in expressive power to a temporal calculus based on a temporal logic with operators **since** and **until**. Gabbay and McBrien [7] considered a refinement of **TA** which is also based on a temporal logic with **since** and **until**. These algebras do not deal with multiple granularity of time or multiple time-lines; their main motivation is to propose a temporal relational algebra which can be used as a basis for temporal relational completeness. Orgun and Müller [13, 12] proposed a temporal algebra called TRA based on a discrete temporal logic with operators `first` and `next`. When all relations are defined over the given global time-line, $\Re$ degenerates into TRA.

The rest of the paper is organized as follows. Section 2 introduces a model of time based on clocks and the notion of a clocked database. Section 3 discusses the operators of $\Re$ and their semantics. It is in particular shown that the meaning of a given expression of $\Re$ over a given clocked database is a clocked relation defined over the clock of the expression. Section 4 discusses temporal integrity constraints. Section 5 concludes the paper with a brief summary.

2 Clocked Data Model

2.1 Clocks

The set of all moments in time (i.e., the time-line) is modeled by the set of natural numbers $\omega = \{0, 1, 2, 3, \ldots\}$. Then clocks are defined as sequences over ω, each of which representing a different time-line.

Definition 1 (Clocks) *A clock $C : \omega \to \omega$ is a strictly increasing sequence of natural numbers. (it must satisfy the condition that $C(0) < C(1) < C(2) < \ldots$).*

We use the notation $\langle 0 \mapsto t_0, 1 \mapsto t_1, 2 \mapsto t_2, \ldots \rangle$ where $i \mapsto t_i$ means that the value i is mapped to the value t_i. We simply write $\langle t_0, t_1, t_2, \ldots \rangle$ to denote a clock where t_i is the time indexed by i. For finite clocks, we write $\langle t_0, t_1, \ldots t_n \rangle$. The global (time-line) clock is the sequence $\langle 0, 1, 2, 3, \ldots \rangle$. The empty clock is the empty sequence $\langle\,\rangle$.

Given a clock C, we write $\mathbf{dom}(C)$ to denote the domain (index set) of C, and $\mathbf{range}(C)$ to denote the range (image) of C. The notation $\mathbf{rank}(t, C)$ refers to the index of time t on clock C, i.e., $C(\mathbf{rank}(t, C)) = t$.

Let C be a clock. We write $t \in C$ if time t occurs in C, or $t \in \mathbf{range}(C)$ (t is a moment in time on clock C, not an index value).

We now define an ordering relation on clocks as follows.

Definition 2 ($\sqsubseteq$) *For any given clocks C_1 and C_2, we write $C_1 \sqsubseteq C_2$ if for all $t \in C_1$, we have $t \in C_2$.*

It can be shown that the set of clocks, denoted by CK, is a complete lattice in which the global clock is the maximum element and the empty clock is the minimum element.

We now define two operations on clocks that are analogous to set intersection and union. Let $C_1, C_2 \in \mathsf{CK}$.

$$C_1 \sqcap C_2 \equiv g.l.b.\{C_1, C_2\} \qquad C_1 \sqcup C_2 \equiv l.u.b.\{C_1, C_2\}$$

These operations can be generalized to arbitrary sets of clocks. Note that the *g.l.b.* (greatest lower bound) of two given clocks can be obtained by taking the common moments in time from both clocks whereas the *l.u.b.* (least upper bound) of two given clocks can be obtained by taking all the moments from each clock.

2.2 Clocked Databases

Let a *relation scheme* (or just *scheme*) Σ be a finite set of *attribute names*, where for any attribute name $A \in \Sigma$, *dom*(A) is a non-empty *domain* of values for A. A *tuple* on Σ is any map $t : \Sigma \to \cup_{A \in \Sigma} dom(A)$, such that $t(A) \in dom(A)$, for each $A \in \Sigma$. Let $\tau(\Sigma)$ denote the set of all tuples on Σ.

Definition 3 *A* relation R *on scheme Σ is any subset of $\tau(\Sigma)$. We let $\mathcal{P}(\Sigma)$ be the set of all relations on Σ.*

For computability reasons, we stipulate that relations are finite sets of tuples.

Clocked relations are an indexed collection of ordinary relations defined over a given clock:

Definition 4 *A* clocked relation R *on scheme Σ is a tuple $\langle C, T \rangle$ where $C : \omega \to \omega$ is a clock and $T : \omega \to \mathcal{P}(\Sigma)$.*

Note that $T(0)$ is the value (or extension) of R at time $C(0)$, $T(1)$ at time $C(1)$, and so on. For instance, if $C = \langle 1, 5, 7, 18, \ldots \rangle$, then the clocked relation has defined values at moments in time $C(0) = 1, C(1) = 5, C(2) = 7, C(3) = 18$ and so on. However, it is not defined at those moments in time which are not on its clock.

We define an ordering relation on clocked relations as follows: If $\langle C, T \rangle$ and $\langle B, S \rangle$ are clocked relations on the same scheme, then we say that $\langle C, T \rangle \sqsubseteq \langle B, S \rangle$ if $C \sqsubseteq B$ and $T(i) \subseteq S(i)$ for all $i \in C$.

A clocked database is a collection of clocked relations. In the following exposition, we use the notation $[X \to Y]$ to denote the set of functions from set X to set Y.

Definition 5 *Let Rel be a countable set of relation symbols, such that for any symbol $p \in Rel$, Δ_p is a scheme. Then a* clocked database db_{Rel} *for Rel is a tuple $\langle Rel, \mu \rangle$ where for all $p \in Rel$, $\mu(p)$ is a clocked relation on scheme Δ_p (i.e., $\mu(p) : \mathsf{CK} \times [\omega \to \mathcal{P}(\Delta_p)]$.)*

We also write $db_{Rel}(p)$ to refer to the clocked relation assigned to $p \in Rel$ by the clocked database db.

Example 1 Suppose that there is a library which only holds books. Book acquisitions are dealt with once each week. Once a book is in stock, it remains in stock forever. The library also has customers (borrowers) whose names and addresses are also kept on record.

The library database has the following relation schemas:

`stock` (CALLNO, NAME)
`acquisition` (CALLNO, NAME, PRICE)
`onloan` (CALLNO, CUST_NAME)
`customer` (CUST_NAME, ADDRESS)

The set of relation symbols in the database is $Rel = \{$`stock`, `acquisition`, `onloan`, `customer`$\}$ with their associated relation schema; keys are underlined.

The clocked database $db_{Rel} = \langle Rel, \mu \rangle$ assigns a clocked relation to each of the given relations. Suppose that in db_{Rel} the relations `stock`, `onloan` and `customer` are all assigned the clock $\langle 0, 1, 2, 3, \ldots \rangle$ and `acquisition` is assigned the clock $\langle 0, 7, 14, \ldots \rangle$. Here the first clock may represent the days of the week (monday, tuesday etc) and the second clock mondays. Suppose that db_{Rel} assigns the clocked relations shown partially in figure 1 to `stock` and in figure 2 to `acquisition`. In the figures, the notation $C(i) \mapsto T(i)$ is used to show the value of the relations at moment $C(i)$, not at an index value i. ∎

0 ⟼

TK17	War and Peace
QA75	Oliver

1 ⟼

TK17	War and Peace
QA75	Oliver
HF97	The Hobbit

...

6 ⟼

TK17	War and Peace
QA75	Oliver
HF97	The Hobbit

7 ⟼

TK17	War and Peace
QA75	Oliver
HF97	The Hobbit
QA12	Mobbydick

...

14 ⟼

TK17	War and Peace
QA75	Oliver
HF97	The Hobbit
QA12	Mobbydick
BC52	Germinal
CH91	HTML Guide

...

Figure 1. The `stock` relation

0 ⟼

TK17	War and Peace	158.95
HF97	The Hobbit	25.05

7 ⟼

TK17	Mobbydick	33.35
BC52	Germinal	85.10

14 ⟼

CH91	HTML Guide	299.99

21 ⟼ {}

...

Figure 2. The `acquisition` relation

3 Clocked Relational Algebra

An expression of $\Re$ consists of relation symbols, and their compositions using point-wise extensions of the operators of the relational algebra and temporal operators.

3.1 Point-wise Operators

An operator of $\Re$ is called *pointwise* if the value of any expression involving that operator at any time t depends entirely on the operand values at the same time t. Note that the notion of a point-wise operation has been extensively discussed in the literature [14, 13].

In the following we write $\Delta(\Theta)$ to refer to the clock component of a given operation and $\Psi(\Theta)$ to the relation component of a given operation.

For any n-ary operator Θ on relations defined in the relational algebra [3], we let $\vec{\Theta}$ be an n-ary pointwise operator of $\Re$, such that

$$\Psi(\vec{\Theta}) = \lambda R_1, \ldots, R_n.\lambda s.\Theta(\Psi(R_1)(s), \ldots, \Psi(R_n)(s))$$
$$\Delta(\vec{\Theta}) = \lambda R_1, \ldots, R_n.\lambda s.\sqcap\{\Delta(R_1), \ldots, \Delta(R_n)\}$$

In other words, at each moment in time, the operator of the relational algebra is applied to operands at that moment in time, and the resulting clocked relation is an indexed collection of the results from each moment in time.

Thus, $\vec{\pi}_\Sigma$ and $\vec{\sigma}_F$ are unary operators and $\vec{\cup}$, $\vec{\cap}$, $\vec{-}$, $\vec{\times}$ and $\vec{\bowtie}$ are binary operators of $\Re$.

Let $x \geq 1$. The aggregation operators are $\vec{\mathtt{sum}}_x$, $\vec{\mathtt{avg}}_x$, $\vec{\mathtt{count}}$, $\vec{\mathtt{max}}_x$ and $\vec{\mathtt{min}}_x$ with their obvious interpretations. For example, if $< C, T >$ is a time-varying relation, then

$$\vec{\mathtt{sum}}_x(< C, T >) = \langle C, \langle \mathtt{sum}_x(T(i)) \mid i \in \mathbf{dom}(C \sqcap C') \rangle \rangle$$

where each $\mathtt{sum}_x(T(i))$ is a single-valued relation.

Example 2 The expression $\vec{\mathtt{sum}}_{PRICE}(\mathtt{acquisition})$ can be used to find the total cost of the books acquired in a given week. The result depends on when the query is evaluated. For instance, if the time of evaluation is 7, then the answer

is the relation $\{\langle 118.45\rangle\}$. If the time of evaluation is 14, then the answer is the relation $\{\langle 299.99\rangle\}$. ∎

An important criterion for a temporal algebra is that it should be a consistent extension of Codd's relational algebra [11]. It can be easily shown that $\Re$ satisfies the criterion. Therefore $\Re$ inherits (pointwise) analogues of all properties of the relational algebra, such as distributive and associative laws, and the definitions of the other relational operators such as $\div$, θ-join, and $\bowtie$, and so on. These properties, together with some other properties involving temporal operators, can be used in query optimization.

3.2 Temporal operators

An operator of $\Re$ is called *temporal* if it allows looking into the future or past values of the operands in arriving at the *current* value of an expression. $\Re$ offers four temporal operators explained below.

The unary operator `first` results in the propagation of the value of its operand at the first moment of its clock. Thus, for any clocked relation $R = \langle C, T\rangle$, `first` R is the tuple

$$\langle C, \langle T(0), T(0), T(0), \ldots\rangle\rangle.$$

More formally we can define the meaning of `first` by providing the definitions of its relation component, $\Psi(\texttt{first})$ and its clock component, $\Delta(\texttt{first})$. The clock of the resulting relation is the same as the clock of the input relation.

$$\Psi(\texttt{first}) = \lambda R.\lambda s.\Psi(R)(0)$$
$$\Delta(\texttt{first}) = \lambda R.\lambda s.\Delta(R)(s)$$

The operator `next` is the *tomorrow* operator as it permits looking one step in the future of its operand. So, `next` R is the tuple

$$\langle C, \langle T(1), T(2), T(3), \ldots\rangle\rangle.$$

When C is an infinite clock, the clock of the resulting relation is the same as the clock of the input relation:

$$\Psi(\texttt{next}) = \lambda R.\lambda s.\Psi(R)(s+1)$$
$$\Delta(\texttt{next}) = \lambda R.\lambda s.\Delta(R)(s+1)$$

We need to consider the case that, when the clock of the given relation is finite, the last moment on the clock will not have a next moment defined for it. Let $k = \mathbf{card}\{\Delta(R)\}$, that is, the number of moments in time on a finite clock $\Delta(R)$. Then we have:

$$\Psi(\texttt{next}) = \lambda R.\lambda s < k-1.\ \Psi(R)(s+1) \text{ if } \Delta(R) \text{ is finite}$$
$$\Delta(\texttt{next}) = \lambda R.\lambda s < k-1.\ \Delta(R)(s+1) \text{ if } \Delta(R) \text{ is finite}$$

From the definition, we can see that the clock of `next` R will have one less moment then the clock of R.

We write `next`$[k]$ for *k*-folded applications of `next`. In case $k = 0$, `next`$[k]$ is the empty string.

Example 3 Consider the query "Which books were in the stock in moment 6?" The index of moment 6 is also 6 on the clock of `stock`, so we have the following expression:

$$\texttt{first next}[6]\ (\vec{\pi}_{\textit{NAME}}(\texttt{stock}))$$

The temporal operator(s) `first next`$[6]$ move the context to moment indexed by 6. Then the answer is the relation $\{\langle'\text{War and Peace}'\rangle, \langle'\text{Oliver}'\rangle, \langle'\text{The Hobbit}'\rangle\}$ at any given moment at which the query is evaluated. ∎

We would also like to refer to *the most recent or current* value of a clocked relation (e.g., most recent employees) with respect to a given time of evaluation. This is achieved by the `current` operator that samples its operand onto the global time-line (clock):

$$\Psi(\texttt{current}) = \lambda R.\lambda s.\ \textbf{if } k = 0 \textbf{ then } \emptyset \textbf{ else } \Psi(R)(k-1)$$
$$\Delta(\texttt{current}) = \lambda R.\lambda s.s$$

where $k = \mathbf{card}\{\Delta(R)(i) \leq s \mid i \in \mathbf{dom}(\Delta(R))\}$. The main idea is to find the value of R at a time on its clock that is closest to time s.

Example 4 Consider the query "Which books have been acquired this week but not yet placed in stock?" Since book acquisitions are only handled on a certain day of each week, this information would not be accessible on other days of the week unless the `acquisition` relation is sampled using `current`.

$$\vec{\pi}_{NAME}((\texttt{current acquisition}) \vec{-} \texttt{stock})$$

There is no need to apply `current` operator to `stock` because it already runs on the global clock. ∎

Note that the sequence $\langle 0, 1, 2, \ldots\rangle$ of all time points is infinite only to the right and not to the left. If, in addition, it were also infinite to the left, for example, if the sequence of valid time points were the set $\mathcal{Z}$ of integers $\langle \cdots, -2, -1, 0, 1, 2, \cdots\rangle$, we could have had a unary operator `prev` (*yesterday* operator) defined as

$$\Psi(\texttt{prev}) = \lambda R.\lambda s.\Psi(R)(s-1)$$
$$\Delta(\texttt{prev}) = \lambda R.\lambda s.\Delta(R)(s-1)$$

where $\Delta(R)$ is assumed to be infinite in the negative direction. We could also provide the definition of `prev` when $\Delta(R)$ is finite in the negative direction; we omit the details. Observe that `prev` is the complete inverse of `next`. However, such an operator is not possible now, because the above functions are not defined for $s = 0$ over ω.

We solve the problem by introducing a binary operator `fby`, which uses the index 0 value of its first operand only in

that special case $s = 0$. Let R_1 and R_2 be clocked relations on the same scheme. Then we have:

$$\Psi(\texttt{fby}) = \lambda R_1, R_2.\lambda s.$$
$$\quad \textbf{if } s = 0 \textbf{ then } \Psi(R_1)(0) \textbf{ else } \Psi(R_2)(s + k - 1)$$

$$\Delta(\texttt{fby}) = \lambda R_1, R_2.\lambda s.$$
$$\quad \textbf{if } s = 0 \textbf{ then } \Delta(R_1)(0) \textbf{ else } \Delta(R_2)(s + k - 1)$$

where $k = \mathbf{card}(\{x \mid x \in \Delta(R_2)\ \&\ x > \Delta(R_1)(0)\})$.

Now `prev` can be defined in terms of `fby` as follows:

$$\texttt{prev}\ R\ =_{df}\ \emptyset\ \texttt{fby}\ R$$

where $\emptyset$ is the empty clocked relation.

Example 5 Consider the query "Which books were in the stock yesterday?" Here is the time-dependent expression: $\vec{\pi}_{NAME}(\texttt{prev stock})$. At time 0, there is no yesterday, so the answer is the empty relation; at time 1, the answer is the relation $\{\langle'\text{War and Peace}'\rangle, \langle'\text{Oliver}'\rangle\}$ and so on. ∎

3.3 Interpretation of Expressions

An expression of $\Re$ is interpreted over a given clocked database. Then the meaning of an expression is a clocked relation over some clock determined by the clocks of relation symbols appearing in the expression.

Definition 6 *Given a clocked database* $\mathsf{DB} = \langle Rel, \mu\rangle$*, and an expression* E *over* DB*, then the meaning of each kind of expression of* $\Re$ *over* DB *is defined as follows.*

- $\mathsf{DB}(r) = \mu(r)$ *for all* $r \in Rel$.
- $\mathsf{DB}(\vec{\Theta}(E_1, \ldots, E_n)) = \vec{\Theta}(\mathsf{DB}(E_1) \ldots \mathsf{DB}(E_n))$ *for any operator* $\vec{\Theta}$ *of* $\Re$ *with arity n.*

It can be shown that all operations of the algebra are closed, that is, the meaning of a legal expression over a given clocked database is a clocked relation. The closure property is one of the important criteria that temporal algebras should satisfy [11].

Theorem 1 *Given a clocked database* $\mathsf{DB} = \langle Rel, \mu\rangle$*, and a legal expression* E *over* DB*, then* $\mathsf{DB}(E)$ *is a clocked relation.*

Ignoring the storage structures for storing clocked relations, expressions of $\Re$ can be evaluated using the naive evaluation method discussed below. Since clocked relations are inherently infinite when defined over infinite clocks, we restrict the evaluation method to particular moments in time or event-based intervals.

Given a clocked database $\mathsf{DB} = \langle Rel, \mu\rangle$, an expression E over DB, and a time t, we want to find out the value of $\mathsf{DB}(E)$ at time t. We first have to check if $t \in \Delta(\mathsf{DB}(E))$ using the clocks of the relation symbols that occur in E. For this purpose, we need to consult the clock component definitions of each operator that appear in E (we omit the details). If $t \in \Delta(\mathsf{DB}(E))$, the expression E is guaranteed to have a defined value at time t. If not, we stop the evaluation of the expression.

If $t \in \Delta(\mathsf{DB}(E))$, then we first transform t into appropriate moments in time for the immediate subexpressions of E, again using the clock component definitions, then obtain the values of the sub-expressions at those moments in time, and finally apply the primary operator to the values to obtain the value of E at time t.

Example 6 Consider the query given in example 3. If $t \in \Delta(\mathsf{DB}(\alpha(\texttt{first next}[6]\ (\vec{\pi}_{NAME}\ \texttt{stock}))))$, we can evaluate the expression. The clock of the expression is the clock of `stock`, so $t \in \Delta(\texttt{stock})$. We now compute the rank of t in the clock of the expression, that is, find $k = \mathbf{rank}(t, \mathsf{DB}(E))$. Then we proceed to the evaluation of the expression:

$$\begin{aligned}
&\Psi(\mathsf{DB}(\texttt{first next}(\vec{\pi}_{NAME}\ \texttt{stock})))(k)\\
&\quad = \Psi(\texttt{first}\ \mathsf{DB}(\texttt{next}[6](\vec{\pi}_{NAME}\ \texttt{stock})))(k)\\
&\quad = \Psi(\mathsf{DB}(\texttt{next}[6](\vec{\pi}_{NAME}\ \texttt{stock})))(0)\\
&\quad = \Psi(\texttt{next}[6](\mathsf{DB}(\vec{\pi}_{NAME}\ \texttt{stock})))(0)\\
&\quad = \Psi(\mathsf{DB}(\vec{\pi}_{NAME}\ \texttt{stock}))(6)\\
&\quad = \Psi(\vec{\pi}_{NAME}\ \mathsf{DB}(\texttt{stock}))(6)\\
&\quad = \vec{\pi}_{NAME}(\Psi(\mathsf{DB}(\texttt{stock}))(6))\\
&\quad = \vec{\pi}_{NAME}(\Psi(\mu(\texttt{stock}))(6))\\
&\quad = \pi_{NAME}\,(\langle\text{TK17}, '\text{War and Peace}'\rangle, \ldots\})\\
&\quad = \{\langle'\text{War and Peace}'\rangle, \langle'\text{Oliver}'\rangle, \langle'\text{The Hobbit}'\rangle\}
\end{aligned}$$

Again the resulting value will be the same whenever the query is evaluated. ∎

4 Integrity Constraints

Recall that $\Re$ is based on the temporal logic TLC [9]. TLC would also allow us to express temporal integrity constraints that the data in a clocked database must satisfy (see [15] for a more extensive discussion on temporal relationships and constraints). A clocked database provides the basis for an interpretation for formulas in TLC, because predicate symbols in a given formula would correspond to relation symbols in a clocked database. The meaning of a predicate symbol is just like a clocked relation in a clocked database; we omit the details.

Suppose that TLC is also extended with temporal modalities $\Box$ (from now on) and $\Diamond$ (now or sometime in the future). These modalities would allow us to express temporal relationships and temporal constraints succinctly. For instance, an integrity constraint that says "all acquired books

must be available in stock soon after they are acquired" could be expressed by the following formula:

$$\Box(\forall C, N, P)(\texttt{acquisition}(C, N, P) \rightarrow \Diamond\texttt{stock}(C, N))$$

And the constraint that says "once a book in in stock, it remains in stock forever" could be expressed by the formula:

$$\Box(\forall C, N)(\texttt{stock}(C, N) \rightarrow \Box\texttt{stock}(C, N))$$

A clocked database satisfies given integrity constraints if the formulas representing the constraints are true under the interpretation that corresponds to the given clocked database. For the library database, it is easy to see that integrity constraints given above are both valid. However, as the database is being updated, there may be a period during which the integrity constraints are not true, for instance, when a book is just acquired, but not immediately processed. In this case, the constraints may be flagged for future processing.

5 Concluding Remarks

We have outlined a temporal algebra $\Re$ that supports relations based on different time-lines such as multiple time series data [4]. Temporal operators are used to navigate through multiple time-lines, and change the clocks of resulting relations. Temporal operators behave in a uniform manner over their operands, keeping their intuitive meanings, but the end result varies with the clocks of the operands involved. All of this is transparent to the end-user.

Future work includes some extensions of $\Re$ to deal with clock alignments and multiple granularities. Sometimes various relations may be based on different clocks with the same periodicity, for instance, clocks $\langle 0, 7, 14, 21, \ldots\rangle$ and $\langle 1, 8, 15, 22, \ldots\rangle$. It may be desirable to align such relations without using the sampling operator. Liu and Orgun [10] suggested an embedding of timing systems with multiple granularities into TLC so that it can be used to represent and reason about relations based on multiple levels of granularity. We may be able to use their approach to embed a timing system into $\Re$ so that it can be used to manipulate clocked relations with multiple granularities.

Acknowledgements

This work has been supported in part by The Australian Research Council (ARC) and The Department of Education, Training and Youth Affairs (DETYA). Thanks are also due to L. Flax and C. Swe for many useful discussions.

References

[1] P. Caspi and N. Halbwachs. A functional model for describing and reasoning about time behavious of computing systems. *Acta Informatica*, 22:595–627, 1986.

[2] J. Chomicki. Temporal query languages: A survey. In *Proc. of The First International Conference on Temporal Logic*, volume 827 of *LNAI*, pp.506–534, 1994. Springer-Verlag.

[3] E. F. Codd. A relational model of data for large shared data banks. *Communications of the Association for Computing Machinery*, 13(6):377–387, 1970.

[4] W. Dreyer, A. K. Dittrich, and D. Schmidt. Research perspectives for time series management systems. *SIGMOD Record*, 23(1):10–15, 1994.

[5] C. E. Dyreson and R. T. Snodgrass. Temporal granularity. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, pages 347–383. Kluwer Academic Press, 1995.

[6] J. Euzenat. An algebraic approach to granularity in time representation. In *Proc. of TIME'95: 2nd International Workshop on Temporal Representation and Reasoning*, Melbourne Beach, Florida, USA, April 1995.

[7] D. Gabbay and P. McBrien. Temporal logic & historical databases. In *Proc. of the 17th Very Large Data Bases Conference*, pages 423–430. Morgan Kauffman, 1991.

[8] G. M. Landau, J. P. Schmidt, and V. J. Tsotras. Historical queries along multiple lines of time evolution. *VLDB Journal*, 4:703–726, 1995.

[9] C. Liu and M. A. Orgun. Dealing with multiple granularity of time in temporal logic programming. *Journal of Symbolic Computation*, 22(5&6):699–720, 1996.

[10] C. Liu and M. A. Orgun. Embedding a timing system into TLC. In *Proc. of TIME'98: 5th International Workshop on Temporal Representation and Reasoning*, pages 105–112. IEEE Computer Society Press, 1998.

[11] L. E. McKenzie Jr. and R. Snodgrass. Evaluation of relational algebras incorporating the time dimension in databases. *ACM Computing Surveys*, 23(4):501–543, 1991.

[12] M. A. Orgun. Incorporating an implicit time dimension into the relational model and algebra. *RAIRO Theoretical Informatics and Applications*, 30(3):231–260, 1996.

[13] M. A. Orgun and H. A. Müller. A temporal algebra based on an abstract model. In *Proc. of the 4th Australian Database Conference*, pages 301–316. World Scientific, 1993.

[14] J. Plaice. Nested clocks: The LUSTRE syncronous dataflow language. In *Proc. of the 1989 International Symposium on Lucid and Intensional Programming*, pages 1–17, Arizona State University, Tempe, U.S.A., 1989.

[15] E. Rose and A. Segev. Toodm - a temporal object-oriented data model with temporal constraints. In *Proc. of the 10th International Conference on the Entity Relationship Approach*, oct 1991.

[16] A. U. Tansel et al., editors. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings Publishing Company, Redwood City, CA, 1993.

[17] A. Tuzhilin and J. Clifford. A temporal relational algebra as a basis for temporal relational completeness. In *Proc. of the 16th International Conference on Very Large Data Bases*, pages 13–23, August 13–16 1990. Morgan Kaufmann.

[18] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with granularity of time in temporal databases. In *Proc. of the Third International Conference CAiSE'91*, pages 124–140, May 13–15 1991. Springer-Verlag.