Prescription for Modelling Time in Databases

Mehmet A. Orgun
Department of Computing
Macquarie University
Sydney, NSW 2109, Australia

[ABSTRACT] The prevailing approach to modelling the time dimension in databases is, in one form or another, the use of an explicit representation of time. In this paper we propose a temporal relational model and algebra, based on temporal semantics, to incorporate an *implicit* time dimension in databases. In temporal semantics, time-varying relations are an indexed collection of ordinary relations, one for each moment in time. A temporal database is modelled by a collection of time-varying relations. Temporal databases are queried using a temporal relational algebra (TRA) which extends the relational algebra point-wise upon the set of natural numbers. We also give examples of time-specific and time-relative queries, and of combining time-varying data by temporal intersection, union and splicing. Although TRA lacks the ability to explicitly manipulate time, we show how temporal aggregation and when-type queries can be formulated using a technique called "tagging".

1 Introduction

The relational algebra [5] 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. 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; Snodgrass [22] provides a summary of the status of research on temporal databases and Kline [13] provides a recent bibliography. The prevailing approach to modelling the time dimension in databases is, in one form or another, the use of an explicit representation of time at both the tuple level and the attribute level [3, 4, 7, 9, 20].

Most of the temporal algebras reported in the literature are not algebras in the mathematical sense, i.e., they are not *closed* since expressions do not always evaluate to time-varying relations of the underlying models. This is one of the consequences of the use of an explicit time dimension in databases: the operators provided in these algebras are in fact designed to exploit actual representations of time-varying data and to manipulate complex time-varying attributes and temporal elements. Some algebras are not unisorted; in other words, the results of expressions can be time-varying relations, snapshot relations, or even temporal elements such as intervals. For example, Clifford's [3] and Sarda's [20] algebras include an operator to drop the time components of a given time-varying relation to obtain *snapshot* relations, and Gadia's [9] algebra allows temporal elements. Very few algebras [16, 26] support the standard definitions of algebraic operators such as intersection and θ-join.

McKenzie and Snodgrass [17] give an extensive evaluation of relational algebras incorporating the time dimension in databases, and they outline a number of criteria for the comparison of algebras. Some of the important criteria are: (1) a temporal algebra should be a consistent extension of the relational algebra, (2) its formal semantics is well-defined, (3) it is an algebra, (3) it supports basic algebraic equivalences, (4) it reduces to the relational algebra (over moments in time), (5) it supports the standard definitions of intersection ($\cap$), θ-join, natural join ($\bowtie$), and quotient ($\div$), and (6) it

includes aggregates. Most of the proposed algebras do not satisfy all of these criteria, and only a few algebras include aggregates because of the complicated interval semantics. These criteria are our starting point.

In this paper, we consider an extension of the relational model based on temporal semantics. As a query language for the model, we introduce a temporal (relational) algebra, which we call TRA, as a *point-wise* extension of the relational algebra. The algebra TRA is a revised version of a temporal algebra, originally proposed by Orgun and Müller [18]. In the underlying relational model for temporal databases, time-varying relations are an (infinite) collection of finite relations indexed by moments in time. The algebra TRA by design has the relational algebra as a special case for each moment in time; hence it inherits all the properties of the relational algebra. Time-varying data from different moments in time are joined through the use of temporal operators, not by the use of explicit references to time. In other words, time is *implicit* in the model and algebra. However, temporal databases of the underlying model can only model *valid-time* [10, 23].

The temporal relational model corresponds to a representation of temporal data based on tuples extended with moments in time as time-stamps, hence it satisfies another criterion of McKenzie and Snodgrass [17]. In Clifford et al [2], temporal models that support 1NF time-varying relations, such as our temporal relational model, are called *temporally ungrouped* data models, as opposed to *temporally grouped* data models. Therefore TRA has the expressive power of a temporally ungrouped algebra. The model may introduce a fair amount of redundancies, but the representation at the implementation level need not be based on this abstract model; it can use event-based interval time-stamps to save space. If the representation is *homogeneous* [9], that is, attribute values coexist in any given tuple, we can switch from an event-based interval representation at the tuple or attribute level to time-stamping of tuples and vice versa.

Aggregation of time-varying data is an important operation in temporal databases, but it is supported by a few of the proposed algebras (e.g., Tansel [25]). Other attempts to provide temporal aggregation include TQUEL of Snodgrass [21], and the query language of Wuu and Dayal [29, 6]. TRA also supports point-wise extensions of the standard aggregation operators such as `sum`, `avg`, `max`, and so on. Another important operation in temporal databases is to ask queries to find times of when things happen. Since time dimension is implicit in TRA, it lacks the ability to directly formulate when-type queries. Therefore we provide an "indirect" method called *tagging* for manipulating time, which can be used to mimic when-type queries and formulate temporal aggregation accross time. Gabbay and McBrien [8] proposed a similar technique for expressing when-type queries. Another approach is discussed by Lorentzos and Johnson [14].

The sequel first outlines the temporal data model TRA is based on, and discusses its operations. We give examples of time-specific and time-relative queries, and of combining time-varying data by temporal intersection, union, and splicing; and then discuss tagging and temporal aggregation.

2 Temporal Data Model

The notion of a time-varying relation is not well-defined. In most of the reported works, time-stamps and event-based interval-stamps are usually used to represent time-varying data at both the tuple and attribute levels. As in the recent works of Tuzhilin and Clifford [26] and Gabbay and McBrien [8], we seek the answer in temporal logic [1]: the meaning of a predicate symbol in temporal semantics is a mapping from moments in time to relations over a given universe of discourse. The collection of moments in time is usually a discrete and linearly ordered set with an unbounded future.

In our model, a temporal database is modeled by a collection of time-varying relations where a time-varying relation is a mapping, just like the meaning of a predicate symbol in temporal semantics. We assume that the collection of moments in time is modeled by the set of natural numbers, denoted as ω. The elements of ω are uninterpreted, and they can be in terms of hours, days, weeks and so on, depending on the intended application domain. At this stage, we assume that the interpretation

is uniform with respect to the granularity of time. We refer the reader to Wiederhold et al [28] for the description of an algebra which addresses the issue of time granularity in time-varying relations.

Let $\mathcal{P}(S)$ denote the power set of a given set S. An n-ary relation over a given domain U is an element of $\mathcal{P}(U^n)$. An n-ary time-varying relation tr over the domain U is a mapping from the collection of moments in time to n-ary relations over U.

Definition 1 *A time-varying relation with arity n ($n \geq 1$) is an element of $[\omega \rightarrow \mathcal{P}(U^n)]$, that is, an element of the set of functions from ω to $\mathcal{P}(U^n)$.*

We write $\langle tr(0), tr(1), tr(2), \cdots \rangle$ for tr where for all $t \in \omega$, $tr(t) \in \mathcal{P}(U^n)$. We also have an ordering relation between time-varying relations defined as follows.

Definition 2 *Let r and s be time-varying relations with the same arity. We write $r \sqsubseteq s$ if and only if for all $t \in \omega$, $r(t) \subseteq s(t)$.*

Let *Rel* be a countable set of relation symbols and U a given domain over which relations are defined. We require that U have at least one element and be countable. In practice, we may require that U be finite. The set of temporal databases over U, denoted as $\mathcal{DB}$, is defined as follows:

$$\mathcal{DB} = [Rel \rightarrow \bigcup_{n \geq 1} [\omega \rightarrow \mathcal{P}(U^n)]].$$

It is also the case that $\mathcal{DB}$ is a complete lattice induced by the ordering relation $\sqsubseteq$. In more technical terms, members of $\mathcal{DB}$ are called *temporal interpretations* [1]. For any given temporal database $db \in \mathcal{DB}$ and an element p of *Rel* with arity n, the meaning of p in db is given as

$$db(p) \in [\omega \rightarrow \mathcal{P}(U^n)].$$

As complex time-varying attributes or (event-based) intervals are not employed in the abstract model, time-varying relations are in the First Normal Form (1NF) [15, 27]. 1NF is also preserved in some of the other temporal models, for example, those of Lorentzos and Johnson [14] and Jensen and Mark [11]. In Clifford et al [2], temporal models that support 1NF time-varying relations are called *temporally ungrouped* data models.

We now give an example of a temporal database.

Example 1 This database for a university keeps personal and professional data for faculty members and the departments they work for; it is a modified version of a database given in [12]. It consists of four relations:

`dept`(DEPT, SECRETARY, HEAD)
`pers`(NAME, SEX, STATUS, ADDRESS)
`prof`(NAME, DEPT, RANK, SALARY)
`publ`(NAME, JOURNAL, ISSUE)

Here `dept` holds information about each department, `pers` holds personal information for each faculty member, `prof` holds professional information for each faculty member, and `publ` holds information for publications of each faculty member. We have that `dept`, `pers`, `prof`, `publ` $\in$ *Rel*. Attribute names are self-explanatory. The DEPT attribute is the key for `dept` relation, and the NAME attribute for the `pers`, `prof`, and `publ` relations. The attribute HEAD is also used for faculty member names.

We assume that the database came into existence at time 0. Given a $db \in \mathcal{DB}$, it is possible that db(`prof`) is the time-varying relation whose portion from time 0 to time 3 is depicted in figure 1. In this example the logical time is interpreted as months. ∎

$0 \longmapsto$

Ali	Phil	Assist	2400
Carol	CompSci	Assoc	2750

$1 \longmapsto$

Ali	Maths	Assist	2500
Carol	Maths	Assoc	2750
Joy	CompSci	Assist	2350

$2 \longmapsto$

Ali	Maths	Assist	2500
Carol	Maths	Assoc	2750
Joy	CompSci	Assist	2350

$3 \longmapsto$

Ali	Phil	Assoc	2900
Carol	Maths	Full	3250
Joy	CompSci	Assist	2450
Kim	Ling	Full	3500

Figure 1: The `prof` relation from time 0 to time 3

Since the abstract model does not preclude any form of temporally grouped representation of time-varying data, we can use time-stamped tuples, event-based intervals, time-varying attributes and so on (or even a non-uniform representation within the same model). The choice for the representation does not affect TRA or its properties, because the temporal operators of TRA are not designed to exploit actual representations of time-varying data (see below). The representation only affects how TRA is going to be implemented. However, it should be noted that a realistic implementation of TRA is a challenging task due to the level of abstraction the algebra offers.

3 Temporal Relational Algebra

This section first introduces the temporal relational algebra TRA, and then provides the denotational semantics of its expressions. In this paper, we build on the work of Orgun and Müller [18]. Their algebra introduced four temporal operators, namely, `first`, `next`, `prev`, and `fby[.]`. We dispense with `prev` as a primitive operator and simplify the semantics of `fby`. From here on, we refer to the revised TRA simply as TRA. Example uses of the operators of TRA are given in the following section.

3.1 Point-wise Extensions

The algebra TRA is a point-wise extension of the relational algebra upon ω defined as follows. The signature of TRA includes all operators of the relational algebra [5], and the universe of TRA is the set of time-varying relations $\cup_{n>0}[\omega \to \mathcal{P}(U^n)]$. Let $\bigtriangledown$ be a unary operator of the relational algebra, and $\hat{\bigtriangledown}$ the corresponding point-wise operator of TRA. For any given time-varying relation tr, the following holds:

$$\text{For all } t \in \omega, \quad \hat{\bigtriangledown}\,(tr)(t) = \bigtriangledown\,(tr(t)).$$

We can give similar equations for the binary operators. Yaghi [30] introduced the notion of a point-wise operation in the context of intensional algebras.

An alternative characterization of point-wise operators is given as follows. Suppose that we are given a unary operator of the relational algebra, say σ_F. It is a mapping from relations into relations: $\sigma_F \in [P(U^n) \to P(U^n)]$ for any $n \geq 1$. Let `eval` be a time-slice (snapshot) function defined as

$$\texttt{eval} \in [\omega \times [\omega \to P(U^n)] \to P(U^n)]]$$

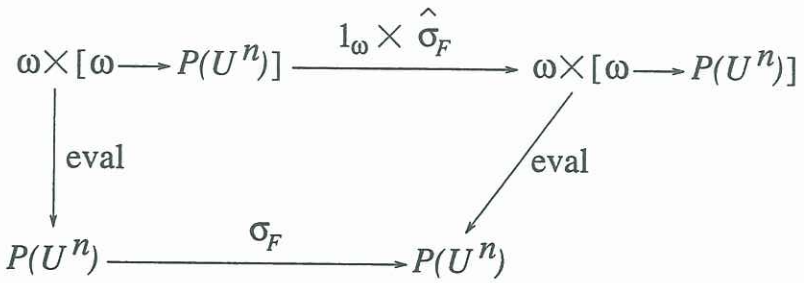

Figure 2: Commuting diagram for point-wise operators

where $\texttt{eval}(t, tr) = tr(t)$. Let $1_\omega \in [\omega \to \omega]$ be an identity mapping between moments in time in ω, i.e., $1_\omega(t) = t$ for any $t \in \omega$. The diagram given in figure 2 induces a *point-wise* operator, say $\hat{\sigma}_F$.

In other words, given a time-varying relation tr, let r be the snapshot of tr at time t (i.e., $\texttt{eval}(t, tr)$). Apply $\hat{\sigma}_F$ to tr and take the snapshot of the resulting time-varying relation at time t (i.e., $\texttt{eval}(t, \hat{\sigma}_F(tr))$). It is equal to the relation which is obtained by applying the corresponding operator σ_F to r. In other words, the diagram *commutes*. Similar commuting diagrams can be given for all the operators of the relational algebra and the corresponding point-wise operators of TRA.

We have the following lemma which states that the meaning of point-wise operators is the same at all moments in time. Therefore we can tell what a point-wise operator does to its argument by looking at the results from each moment in time. The lemma can be naturally extended to include binary point-wise operators.

Lemma 3 *Let $\hat{\nabla}$ be a unary point-wise operator of* TRA. *For all time-varying relations r and s with the same arity, and for all $t, u \in \omega$, if $r(t) = s(u)$, then $\hat{\nabla}(r)(t) = \hat{\nabla}(s)(u)$.*

3.2 Operations

An expression of TRA consists of compositions of algebraic operators and relation symbols. There are three kinds of algebraic operators: (1) point-wise operators, (2) temporal operators, and (3) aggregation operators. These operators are applied to time-varying relations and yield time-varying relations as a result. In the sequel, we adopt the infix notation for binary operators.

We now define the notion of a comparator. Given $x, y \geq 1$, the following are comparators: $x < y$, $x \leq y$, $x > y$, $x \geq y$ and $x \neq y$. We call x and y "comparator values". A comparator value is basically the ordinal number of an attribute in a relation. Given a temporal database *db*, we also allow comparators with one of their operands selected from the universe of *db*. A comparator formula is constructed out of conjunctions, disjunctions and comparators. A detailed discussion on the notions of a comparator and comparator formula can be found in Maier [15, section 3.5].

Point-wise Operators Point-wise operators are $\hat{\cap}$, $\hat{\cup}$, $\hat{\times}$, $\hat{-}$, $\hat{\pi}_X$, and $\hat{\sigma}_F$. Here X is a finite sequence of comparator values, and F is a comparator formula. All of $\hat{\cap}$, $\hat{\cup}$ and $(\hat{-})$ take two arguments with the same arity (compatible time-varying relations). At any given time t, the outcome of a point-wise operation depends *only* on the values of its arguments at time t. For instance, given the expression $r \,\hat{\cap}\, s$ where r and s are time-varying relations, the resulting time-varying relation is the point-wise intersection of r and s, i.e.,

$$r \,\hat{\cap}\, s = \langle r(0) \cap s(0), r(1) \cap s(1), r(2) \cap s(2), \cdots \rangle,$$

or, using the familiar λ-notation, $r \,\hat{\cap}\, s = \lambda t.r(t) \cap s(t)$.

Temporal Operators There are three (primitive) temporal operators in TRA: `first`, `next`, and binary `fby`. The temporal operator `first` freezes a time-varying relation at its initial value; `next` is the "tomorrow" operator; and `fby` (read as *followed by*) does temporal splicing. In general, at any given time t, the outcome of a temporal operation may depend on the values of its arguments possibly at time t and at other moments in time.

The formal semantics of temporal operators are given as follows:

- $\texttt{first}(r) = \lambda t.r(0)$.
- $\texttt{next}(r) = \lambda t.r(t+1)$.
- $r \texttt{ fby } s = \lambda t. \begin{cases} r(0), & t = 0 \\ s(t-1), & t > 0 \end{cases}$
  where r and s are compatible time-varying relations.

We now define another temporal operator, `prev` in terms of `fby` as follows:

$$\texttt{prev}(r) =_{df} \emptyset \texttt{ fby } r$$

In other words, `prev` is the "yesterday" operator (it was employed in the original TRA as a primitive operator). The value of any expression of the form `prev` A at time 0 is the empty relation, because we cannot go into the past beyond time 0.

From here on, we use the notation `next[`n`]` and `prev[`n`]` as syntactic sugar for n-folded applications of `next` and `prev`. In case $n = 0$, `next[`n`]` and `prev[`n`]` are the empty string.

Aggregation Operators Let x be a comparator value ≥ 1. The aggregation operators of TRA are $\hat{\texttt{sum}}_x$, $\hat{\texttt{avg}}_x$, $\hat{\texttt{count}}$, $\hat{\texttt{max}}_x$ and $\hat{\texttt{min}}_x$ with their obvious interpretations. These point-wise operators are applied to time-varying relations, and produce unary time-varying relations whose snapshots are single-valued relations. For instance, given the expression $\hat{\texttt{avg}}_x(tr)$ where tr is a time-varying relation whose arity is not less than x, the resulting time-varying relation is given as follows:

$$\hat{\texttt{avg}}_x(tr) = \langle \texttt{avg}_x\, tr(0), \texttt{avg}_x\, tr(1), \texttt{avg}_x\, tr(2), \cdots \rangle,$$

or in the λ-notation: $\hat{\texttt{avg}}_x(tr) = \lambda t.\texttt{avg}_x(tr(t))$. Here $\texttt{avg}_x(r) = \{\langle e \rangle\}$ where e is the average of the values of the x^{th} attribute from the relation r.

We also assume that aggregation operations can be associated with a *by-list*, producing several tuples determined by calculating the aggregate over a subset of the relation [21]. Examples of temporal aggregation are given in section 5.

3.3 Denotational Semantics

Given a temporal database db, an expression E over db contains only those relation symbols defined in db, and values from the domain of db. Let $[\![E]\!](db)$ denote the meaning (denotation) of E with respect to db. We assume that all expressions are legal, i.e., arities of arguments given in an expression match with respect to the operations involved, and so do the types of attributes over which aggregation operations are performed.

For a relation symbol $p \in Rel$ with arity n, we have that $db(p) \in [\omega \rightarrow \mathcal{P}(U^n)]$ where U is the domain of db. Thus $[\![E]\!](db)$ is also an element of $[\omega \rightarrow \mathcal{P}(U^k)]$ for some $k \in \omega$. In general, given an expression E, we have that

$$[\![E]\!] \in [\mathcal{DB} \rightarrow \bigcup_{n>0} [\omega \rightarrow P(U^n)]].$$

The following is the formal definition of the denotation function $[\![.]\!]$.

Definition 4 *Let $db \in \mathcal{DB}$, and A and B be* TRA *expressions. The denotations of each kind of expressions of* TRA *are defined as follows.*

1. $[\![p]\!](db) = db(p)$ *where* $p \in Rel$.
2. $[\![\hat{\nabla} A]\!](db) = \hat{\nabla}[\![A]\!](db)$ *where* $\hat{\nabla}$ *is any unary operator.*
3. $[\![A \hat{\nabla} B]\!](db) = [\![A]\!](db) \hat{\nabla} [\![B]\!](db)$ *where* $\hat{\nabla}$ *is any binary operator.*

For TRA to be an algebra in the mathematical sense, we need to show that all operations of TRA are *closed*; in other words, the denotation function $[\![.]\!]$ assigns a time-varying relation to each (legal) expression. The theorem can be proved by induction on the structure of expressions.

Theorem 5 *The denotation function $[\![.]\!]$ is well-defined, i.e., for any legal expression E over a temporal database db, $[\![E]\!](db) \in \cup_{n>0}[\omega \rightarrow \mathcal{P}(U^n)]$.*

We assume the usual point-wise definitions of *quotient* ($\hat{\div}$), *θ-join* ($\bowtie_F$), and *natural join* ($\bowtie$) in terms of other point-wise operators of TRA; see [27] for details. Intersection is employed as a primitive operator in TRA. We can import the definitions of these point-wise operators, because all the properties of the relational algebra are preserved in TRA [18].

Note that it can be shown that the original TRA and the revised TRA have the same expressive power. We omit the details.

4 Querying Time-Varying Data

In the following, we stipulate that all queries be evaluated at a given moment t in time (say, "now", "today", "this month" and so on) or over an event-based interval identified by its starting and ending times. In the examples given below, the logical time is interpreted as "months". The evaluation of a query over an interval is performed by evaluating the query at all moments over the interval. We regard an interval as a window through which time-varying data may be queried. Some of the examples are based on variations of the benchmark queries posed by Kalua and Robertson [12].

4.1 Temporal Selection

In many applications, a snapshot of time-varying data at a particular moment in time is of interest. Queries of this kind are time-specific. Suppose we are given the temporal database from example 1. It is possible that *db*(**dept**) is the time-varying relation whose portion from time 0 to 3 is depicted in Figure 3. From here on, we refer to the attributes in a given relation by ordinal numbers. For instance, NAME in **dept** is the 1^{st} attribute.

Example 2 Consider the query "What department did Kim head in month 3?"

$$\hat{\pi}_{3,1}(\hat{\sigma}_{3="Kim"}(\texttt{first next[3] dept}))$$

The temporal operator(s) **first next[3]** move the context to month 3. Since Kim headed in the Linguistics department in month 3, the answer is the relation $\{\langle Kim, Ling\rangle\}$ whenever the query is evaluated. Let E be the expression given above. Then we have that $[\![E]\!](db) = \lambda t.\{\langle Kim, Ling\rangle\}$. ∎

Sometimes we are interested in what happens relative to a reference point in time, say "now", "today" and so on. For instance, we may pose queries like "Is Ali going to get a raise next month?"

$0 \longmapsto$

Phil	Jenny	Ali
CompSci	Tasha	Carol

$1 \longmapsto$

Maths	Tony	Carol
CompSci	Tasha	Joy

$2 \longmapsto$

Maths	Tony	Carol
CompSci	Tasha	Joy

$3 \longmapsto$

Phil	Jenny	Ali
Maths	Tony	Carol
CompSci	Tasha	Joy
Ling	Cheri	Kim

Figure 3: The department relation from time 0 to time 3

In such a query, there is no explicit reference to time, thus the outcome of the query depends on *when* it is evaluated. Queries about the future are not problematic, because, if future data is not available yet, the answer would be the empty relation.

It is also possible to formulate queries about the past.

Example 3 Consider the query "Who was Joy's department head last month?".

$$\texttt{prev}(\hat{\pi}_{1,2,5}(\hat{\pi}_{1,2}(\hat{\sigma}_{1=\text{"}Joy\text{"}}\texttt{prof}) \underset{2=1}{\hat{\bowtie}} \texttt{dept}))$$

Here $\hat{\bowtie}_F$ is the point-wise θ-join operator. The comparator formula $2 = 1$ means that we compare the second and first attribute values from the corresponding tuples.

At time 0, the answer is the empty relation; at times 1 and 2, the answer is the relation $\{\langle Joy, CompSci, Joy\rangle\}$; and so on. We could also ask "Who will be Joy's department head next month?" by replacing `prev` by `next` in the expression. ∎

4.2 Temporal Intersection/Union

We may also be interested in what happened over a period of time in a given department. The period of interest may be time-specific or time-relative. In the following, we first introduce a notation for representing periods of time, i.e., event-based intervals. Let $x, y \geq 0$.

- Time-specific intervals are denoted as $[x, y]$ where $x \leq y$.
- Time-relative intervals are denoted as $[(+/-)x, (+/-)y]$ where $(+/-)x \leq (+/-)y$. The following are time-relative intervals: $[-3, +5]$, $[-6, -1]$ and $[+2, +4]$. The actual interval for any given time-relative interval is in fact determined by the time of evaluation, say *now*. For instance, $[+2, +4]$ represents the interval $[now + 2, now + 4]$.

The expression $[x, y]E$ is called *temporal intersection* and it is interpreted as finding all those tuples in E that are common at every moment in time over the interval $[x, y]$. With this interpretation, interval operators become a syntactic sugar for expressions involving temporal selections and intersections. For instance, the expression $[-1, +1]E$ is a shorthand notation for

$$\texttt{prev}\, E \,\hat{\cap}\, E \,\hat{\cap}\, \texttt{next}\, E.$$

Example 4 Consider the (time-relative) query "Find all those faculty members in the Mathematics department who did not get a raise in the last two months".

$$\hat{\pi}_{1,2}([-2,-1](\hat{\sigma}_{2="Maths"}\texttt{prof}))$$

If any faculty member in the mathematics department did not get a raise over the interval $[-2+now, -1+now]$, the corresponding faculty member tuples would be the same. That is why we can formulate the query by temporal intersection. ∎

We can also "accumulate" time-varying data from different moments in time by *temporal union*. We use a different notation for intervals over which time-varying data will be joined by unions. Let $\langle x, y\rangle$ denote an interval defined just like $[x, y]$. Then the expression $\langle 1, 3\rangle E$ is a syntactic sugar for

$$\texttt{first next}\, E \,\hat{\cup}\, \texttt{first next[2]}\, E \,\hat{\cup}\, \texttt{first next[3]}\, E.$$

We can also make use of the time of evaluation in queries.

Example 5 [Kalua and Robertson [12] query QB3] Consider the query "What departments were headed by Carol and Kim and who were their secretaries?" We assume that the query is restricted to the interval from the beginning up to "now", or up to the time of evaluation of the query:

$$\langle 0, now\rangle(\hat{\sigma}_{3="Carol" \vee 3="Kim"}\texttt{dept})$$

It is the responsibility of the implementation to replace *now* with the time of evaluation during the query evaluation process. ∎

4.3 Temporal Splicing

In some cases, we are interested in cutting out a portion of a time-varying relation, and pasting a portion of another one in its place. This is achieved by the binary temporal operator `fby`. We now consider a more general form of temporal splicing, that is, a parameterized `fby` operator defined in terms of the (primitive) operators of TRA as follows:

- r `fby[0]` $s =_{df} r$ `fby (next` s`)`,
- r `fby[`t`]` $s =_{df} r$ `fby ((next` r`) fby[`$t-1$`] (next` s`))`, for $t > 0$.

Read r `fby[`t`]` s as "r up to time t, and s from time $t+1$ on". We call time t as the *cut-point* for `fby[`t`]`.

Given time-varying relations r and s, we have that r `fby[`t`]` $s = < r_0, r_1, \ldots, r_t, s_{t+1}, s_{t+2}, \ldots >$. The parameterized `fby[.]` operator was employed as a primitive operator in the original TRA [18], because it has a straightforward implementation.

Example 6 Suppose that a 10% overall salary increase is planned as from month t and the university management wants to study its long term effects on the university. The temporal database is now extended with a new `prof` relation, say `projected-prof`, modeling the salary increase starting at month t. For analysis, all references to the `prof` relation before month t should be regarded as references to the original `prof` relation, and at and after month t as references to the `projected-prof` relation. In time-relative queries, it is impossible to determine which relation should be used since there are no explicit references to time. A solution to this problem can be provided by "temporal splicing". The following TRA expression can be used whenever the faculty member relation is required:

$$\texttt{prof fby[}t-1\texttt{] projected-prof}.$$

The operator `fby[`$t-1$`]` guarantees that `prof` is used up to and including the cut point at $t-1$. ∎

5 Explicit Manipulation of Time

It is very natural, especially in temporal databases, to ask queries to find times of when things happened. Since there is no explicit manipulation of time in expressions of TRA, such queries cannot be *directly* formulated. This is a tradeoff for the level of abstraction TRA offers. We adopt an indirect, but effective method, called *tagging*, which can be used to manipulate time *explicitly* in TRA. In other words, it is possible to have the best of the both worlds, i.e., the explicit and implicit manipulation of time, in an abstract algebra such as TRA.

5.1 Tagging and Temporal Aggregation

Suppose that we are interested in the average salary of a faculty member across time. A naive solution would be to use temporal union followed by the aggregation operation $\hat{\mathtt{avg}}_x$. There are problems with this solution, because the relational model does not support duplicate tuples (or *multi-sets*), but we need all the duplicate tuples to obtain a correct answer. For instance, the average salary of Ali from time 0 to time 3 is \$2575 (see figure 1). But using temporal union and aggregation as in the following expression

$$\hat{\mathtt{avg}}_4(\langle 0,3\rangle(\hat{\sigma}_{1=\text{"}Ali\text{"}}\mathtt{prof})),$$

we obtain the wrong result of \$2600. The problem is that the information about Ali stored in the `dept` relation is the same in months 1 and 2. What is needed is the ability in TRA to differentiate between tuples coming from different moments in time.

We provide a solution to the problem using the notion of a *tag*. Tags are basically time-stamps on data. Suppose that the following time-varying relation is given:

$$u = \langle\{\langle 0\rangle\},\{\langle 1\rangle\},\{\langle 2\rangle\},\cdots\rangle.$$

Then we can define a new point-wise operator, say `tag`, as follows:

- $\mathtt{tag}(r) =_{df} r \times u$.

Tags are used much in the same way as *user-defined time* [24], but they are never stored in the database, rather they are dynamically created on demand during the evaluation of TRA expressions. An expression of the form $\mathtt{tag}(E)$ can be evaluated by tagging each tuple resulting from the evaluation of E by the time of evaluation. As a result, temporal union on tagged time-varying relations will not "lose" any time-sensitive information. Ali's average salary can now be computed as

$$\hat{\mathtt{avg}}_4(\langle 0,3\rangle\,\mathtt{tag}\,(\hat{\sigma}_{1=\text{"}Ali\text{"}}\mathtt{prof})).$$

Tagging provides a remedy for the lack of ability of TRA to manipulate time, so that temporal aggregation (and even "when" queries) can be formulated. There are no extra operators in TRA for tagging (cross product is sufficient). Gabbay and McBrien [8] also considered answering when-type queries using a special relation called *time*. This relation is identical to the relation u. Another approach similar to tagging is discussed by Lorentzos and Johnson [14].

5.2 Expressing When-Type Queries

When tagging is directly applied to the base relations (those relations coming from the temporal database), tags represent valid-time. The following examples show how tags can be manipulated using selection, projection and aggregation operators so that certain types of when-type queries can also be formulated.

Example 7 [Kalua and Robertson [12] query QB8] Consider the query "When did the associate professors attain this rank?"

$$\langle 0, now\rangle(\hat{\pi}_{1,7,9}(\texttt{prev(prof)} \underset{1=1\wedge 3\neq 3}{\bowtie} \hat{\sigma}_{3=\text{"}Assoc\text{"}}\texttt{tag(prof)}))$$

We compare the rank of faculty members in consecutive months and if the rank has changed from "anything" to Associate Professor in any given month, the tagged faculty member tuple will be retained in the temporal union. The final result can be obtained by projection. If the time of evaluation is 3, the answer is the relation $\{\langle Ali, Assoc, 3\rangle\}$, because only Ali attained this rank in the period from time 0 to time 3. In order to include those faculty members who were associate professors at time 0, we can take union of the above query with the query

$$\texttt{first}(\hat{\pi}_{1,3,5}(\hat{\sigma}_{3=\text{"}Assoc\text{"}}\texttt{tag(prof)})).$$

Then the answer, again at time 0, is the relation $\{\langle Ali, Assoc, 3\rangle, \langle Carol, Assoc, 0\rangle\}$. ∎

With tagging, aggregation operators can be utilized to express a variety of when-type queries. The following example shows the use of `count` with *by-list*.

Example 8 [Kalua and Robertson [12] query QB7] Consider the query "List all faculty who published in the same journal at least twice, along with the journal issues and publication dates (months)." We first formulate a query to find the names of faculty members and journals that satisfy the condition of the query (call it A).

$$\hat{\pi}_{1,2}(\hat{\sigma}_{3>\text{"}2\text{"}}(\texttt{count}((\langle 0, now\rangle\,\texttt{tag}\,(\texttt{publ}))\ \texttt{by}\ 1,2)))$$

Here the resulting relation after `count` has the attributes: NAME, JOURNAL, and the number of tuples for each distinct pair of NAME and JOURNAL. We refer to these two attributes in the *by-list* using their corresponding ordinal numbers. Final query is formed by θ-join of A and a (sub-)query to obtain a relation with the names of faculty members, journals, issue numbers, and dates:

$$\hat{\pi}_{1,2,5,6}(A \underset{1=1\wedge 2=2}{\bowtie} \langle 0, now\rangle\,\texttt{tag}\,(\texttt{publ})).$$

We could easily define another query for finding the total number of publications per month for each department in the university using `count` and *by-list* on tagged relations (this would be an instance of a group-by-time aggregation). ∎

6 Concluding Remarks

The difference between our approach to temporal algebras and many others reported in the literature is that TRA is a "temporal algebra" in the mathematical sense, and that it is based on temporal semantics. TRA is not a specialized algebra to manipulate explicit representations of time-varying data or temporal elements such as intervals. Consequently, it has a much smoother semantics than many other proposed temporal algebras. We can easily introduce extra temporal operators into TRA and try out new ideas owing to the level of abstraction offered by it. For instance, Orgun and Wadge [19] proposed the use of the original TRA as an algebraic front-end to a modular, temporal extension of DATALOG. Temporal Datalog can naturally express recursion.

We believe that representation is an implementation issue, and should not influence the design of temporal algebras and the choice of temporal operators thereof. This is our "prescription" for modelling the time dimension in databases. We consider the property of separation from representation as an important aspect of a temporal algebra, because freedom from representation ensures portability. Query optimization strategies can be directly based on the formal properties of TRA [18], and they should be considered in conjunction with actual representations and storage structures. TRA can naturally serve as an appropriate target for a temporal query language processor.

Acknowledgements

This research has been supported in part by a Macquarie University Research Grant (MURG). Thanks are due to Rajiv Bagai for his comments on the properties of the temporal operators of TRA, and Lee Flax for his helpful comments on an earlier draft.

References

[1] J. P. Burgess. Basic tense logic. In D. M. Gabbay and F. Guethner, editors, *Handbook of Philosophical Logic, Vol. II*, pages 89–134. D. Reidel Publishing Company, 1984.

[2] J. Clifford, A. Croker, and A. Tuzhilin. On completeness of historical relational query languages. *ACM Transactions on Database Systems*, 19(1):64–116, 1994.

[3] J. Clifford and A. U. Tansel. On an algebra for historical relational databases: Two views. In S. Navathe, editor, *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data*, pages 247–265. ACM Press, 1985.

[4] J. Clifford and D. S. Warren. Formal semantics for time in databases. *ACM Transactions on Database Systems*, 8(2):214–254, June 1983.

[5] 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.

[6] Umeshwar Dayal and Gene T. J. Wuu. A uniform approach to processing temporal queries. In *Proceedings of the 18th Very Large Data Bases Conference*, pages 407–418, Vancouver, British Columbia, Canada, 1992. Morgan Kauffman, Los Altos, Calif.

[7] Soumitra Dutta. Generalized events in temporal databases. In *Proceedings of the Fifth International Conference on Data Engineering*, pages 118–125. IEEE Computer Society Press, 1989.

[8] D. Gabbay and P. McBrien. Temporal logic & historical databases. In *Proceedings of the 17th Very Large Data Bases Conference*, pages 423–430, Barcelona, Spain, September 1991. Morgan Kauffman, Los Altos, Calif.

[9] S. K. Gadia. A homogeneous relational model and query languages for temporal databases. *ACM Transactions on Database Systems*, 13(4):418–448, December 1988.

[10] C. S. Jensen et al. A consensus glossary of temporal database concepts. *SIGMOD RECORD*, 23(1):52–64, March 1994.

[11] Christian S. Jensen and Leo Mark. Queries on change in an extended relational model. *IEEE Transactions on Knowledge and Data Engineering*, 4(2):192–200, April 1992.

[12] Patrick P. Kalua and Edward L. Robertson. Benchmark queries for temporal databases. Technical Report TR379, Computer Science Department, Indiana University, Bloomington, Indiana 47405, USA, March 1993.

[13] Nick Kline. An update of the temporal database bibliography. *SIGMOD RECORD*, 22(4):66–80, December 1993.

[14] Nikos A. Lorentzos and Roger G. Johnson. TRA: A model for a temporal relational algebra. In C. Rolland, F. Bodart, and M. Leonard, editors, *Temporal Aspects in Information Systems*, pages 95–109. North-Holland, Amsterdam, 1988.

[15] David Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[16] E. McKenzie and R. Snodgrass. Extending the relational algebra to support transaction time. In U. Dayal and I. Traiger, editors, *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pages 467–478. ACM Press, 1987.

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

[18] M. A. Orgun and H. A. Müller. A temporal algebra based on an abstract model. In M. E. Orlowska and M. Papazoglou, editors, *Advances in Database Research: Proceedings of the 4th Australian Database Conference*, pages 301–316, Brisbane, Queensland, Australia, February 1–2 1993. World Scientific Singapore.

[19] M. A. Orgun and W. W. Wadge. A relational algebra as a query language for Temporal DATALOG. In A. M. Tjoa and I. Ramos, editors, *Proceedings of DEXA'92: The Third International Conference on Database and Expert Systems Applications*, pages 276–281, Valencia, Spain, September 2–4 1992. Springer-Verlag Wien.

[20] N. L. Sarda. Algebra and query language for a historical data model. *The Computer Journal*, 33(1):11–18, 1990.

[21] R. T. Snodgrass, S. Gomez, and L. Edwin McKenzie, Jr. Aggregates in the temporal query language TQuel. *IEEE Transactions on Knowledge and Data Engineering*, 5(5):826–842, October 1993.

[22] Richard Snodgrass. Temporal databases: Status and research directions. *SIGMOD RECORD*, 19:83–89, December 1990.

[23] Richard Snodgrass and Ilsoo Ahn. A taxonomy of time in databases. In *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data*, pages 236–246. ACM Press, 1985.

[24] Richard Snodgrass and Ilsoo Ahn. Temporal databases. *IEEE Computer*, pages 35–42, September 1986.

[25] A. U. Tansel. A statistical interface for historical relational databases. In *Proceedings of the International Conference on Data Engineering*, pages 538–546, Los Angeles, Calif., February 1987. IEEE Computer Society Press.

[26] A. Tuzhilin and J. Clifford. A temporal relational algebra as a basis for temporal relational completeness. In D. McLeod, R. Sacks-Davis, and H. Schek, editors, *Proceedings of the 16th International Conference on Very Large Data Bases*, pages 13–23, Brisbane, Australia, August 13–16 1990. Morgan Kaufmann Publishers Inc., Los Altos, Calif.

[27] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.

[28] Gio Wiederhold, Sushil Jajoida, and Witold Litwin. Dealing with granularity of time in temporal databases. In R. Andersen, J. A. Bubenko, and A . Sølvberg, editors, *Advanced Information Systems Enginering: Proceedings of the Third International Conference CAiSE'91*, pages 124–140, Trondheim, Norway, May 13–15 1991. Springer-Verlag.

[29] Gene T. J. Wuu and Umeshwar Dayal. A uniform model for temporal object-oriented databases. In *Proceedings of the Eighth International Conference on Data Engineering*, pages 584–593. IEEE Computer Society Press, 1992.

[30] A. A. Yaghi. *An Intensional Implementation Technique for Functional Languages*. PhD thesis, Department of Computer Science, University of Warwick, Coventry, England, 1984.