The VLDB Journal (1996) 5: 167–180

The VLDB Journal
© Springer-Verlag 1996

A complete temporal relational algebra

Debabrata Dey[1], **Terence M. Barron**[2], **Veda C. Storey**[3]

[1] Department of Information Systems and Decision Sciences, College of Business Administration, Louisiana State University, Baton Rouge, LA 70803, USA; e-mail: qmdey@lsuvm.sncc.lsu.edu
[2] Department of Information Systems and Operation Management, College of Business Administration, University of Toledo, Toledo, OH 43606, USA; e-mail: tbarron@utnet.utoledo.edu
[3] William E. Simon Graduate School of Business Administration, University of Rochester, Rochester, NY 14627, USA;
e-mail: storey@ssbibm.ssb.rochester.edu

Edited by Wesley Chu. Received February 1993 / Accepted April 1995

Abstract. Various temporal extensions to the relational model have been proposed. All of these, however, deviate significantly from the original relational model. This paper presents a temporal extension of the relational algebra that is not significantly different from the original relational model, yet is at least as expressive as any of the previous approaches. This algebra employs multidimensional tuple time-stamping to capture the complete temporal behavior of data. The basic relational operations are redefined as consistent extensions of the existing operations in a manner that preserves the basic algebraic equivalences of the snapshot (i.e., conventional static) algebra. A new operation, namely *temporal projection*, is introduced. The complete update semantics are formally specified and aggregate functions are defined. The algebra is closed, and reduces to the snapshot algebra. It is also shown to be at least as expressive as the calculus-based temporal query language TQuel. In order to assess the algebra, it is evaluated using a set of twenty-six criteria proposed in the literature, and compared to existing temporal relational algebras. The proposed algebra appears to satisfy more criteria than any other existing algebra.

Key words: Temporal databases – Historical databases – Relational algebra – Valid time – Transaction time

1 Introduction

Database systems store information about the real world they attempt to represent. However, any useful representation of the real world needs to address the issue of the temporal nature of information, since the real world is very dynamic, changing continually over time. In the relational model (Codd 1970, 1990) the temporal nature of data has been largely ignored, being reflected only through updates while ignoring the past states. In many real world applications where temporal data is critical, time has been modeled in an *ad hoc* fashion, primarily with the help of application programs. Clearly, this defeats the very purpose of the relational model – a high level of independence between the data and the application programs. Thus the need for a more comprehensive data model arises, where time is an intrinsic part of the model.

In recent years, a considerable amount of work has been undertaken in the area of temporal databases (Kline 1993; Snodgrass 1990). Most of these research efforts have been directed towards extending the relational model to incorporate time. However, due to differing points of view, these extended models are very diverse. Two approaches have been proposed in the literature for temporally extending the relational algebra: *tuple* time-stamping, and *attribute* time-stamping. Tuple time-stamping, where the timestamp is a special attribute of the relation scheme and hence is part of every tuple, was first proposed in LEGOL 2.0 (Jones et al. 1979) which uses two implicit time attributes, *start* and *stop*. Ben-Zvi (1982) uses both valid time and transaction time as timestamps of every tuple, and redefines the semantics of the basic relational operations. The historical relations proposed by Navathe and Ahmed (1989, 1993) use tuple time-stamping with valid time, and have a structure similar to that of LEGOL 2.0 (Jones et al. 1979). They extend SQL for temporal relations, and define three new operators, namely *TJOIN*, *TNJOIN* and *COMPRESS*. A similar structure for historical relations is proposed by Lorentzos and Johnson (1988) in their algebra which introduces three new operations that allow them to convert interval representation of timestamp to point representation, and *vice versa*. Their algebra also supports timestamps with nested granularity. Sarda's (1990, 1993) algebra uses a single nonatomic timestamp (time intervals called *period*) instead of the start and endpoints of an interval as in Lorentzos and Johnson (1988) and Navathe and Ahmed (1989), with the extra restriction that two tuples with the same values for the visible (or non-time) attributes and overlapping or adjacent time intervals must be *coalesced* into a single tuple. Basic set-theoretic operations retain their usual meaning, while selection, projection and Cartesian product operations are redefined, and four new operators are introduced.

Clifford and Tansel (1985) were the first to propose that timestamps should be part of the attributes, and not of the whole tuple. The historical algebra by Clifford and Croker (1987, 1993) is based on lifespans, where the basic relational operations are redefined to handle lifespans and four new operators are introduced. Tansel's model (Clifford and Tansel 1985; Tansel 1986, 1987), though similar to Clifford's, supports four types of attributes: time-varying, non-time-varying, set-valued, and atomic. This model uses time intervals, instead of time-points as in Clifford's algebra, to capture the temporal nature of the attributes. Eight new operators are introduced. Aggregate functions are also discussed by Tansel (1986, 1987). Gadia (1988) proposes his homogeneous temporal relational model based on attribute time-stamping, and a restriction called *homogeneity* that restricts the temporal domain to remain the same for all attributes within a single tuple. All the basic relational operations are redefined, and two new operators called *tdom* and *temporal selection* are introduced. Gadia (1986) relax the above *homogeneity* assumption in the multihomogeneous model, while Gadia and Yeung (1988) extends this model to support multiple time dimensions. Two different algebras are proposed by Tuzhilin and Clifford (1990) where the usual relational operations are redefined using snapshot semantics and several new operations are introduced. McKenzie and Snodgrass (1991a) propose a historical relational algebra based on attribute time-stamping. The algebra uses only valid time, and the attributes are single-valued. The basic relational operations are redefined. Three new operations called *historical derivation*, *snapshot* and *historical rollback* are introduced. Two more operators are provided for unique and nonunique aggregates.

Significant work has also been done in the area of object-oriented databases and their temporal extensions. Rose and Segev (1991) extend the object-based entity-relationship model into a temporal object-oriented model. They incorporate temporal structures and constraints in the model and propose an SQL-like query language. Rose and Segev (1992) propose an object-oriented temporal algebra for this model. Wuu and Dayal (1993) start with OODAPLEX, an object-oriented data model, and temporally extend it. Temporal properties of objects are modeled as functions of time-dependent functions. Dayal and Wuu (1992) describe the associated algebra and show how it can be used for processing temporal queries. Cheng and Gadia (1993) also use OODAPLEX as the base and build temporal support in an object-oriented environment. They call the model OOTempDBM and the associated query language OOTempSQL.

The above overview leads to the conclusion that previous research in temporal relational algebra suffers from several shortcomings.

1. Lack of treatment of multiple time dimensions in a unified way. A temporal data model should treat both valid, as well as transaction time,[1] in a uniform way, and should allow the user to chose any one or both based on the demands of the application. Unfortunately, other than the algebra of Gadia and Yeung (1988), no other proposal does this. More specifically, no temporal relational algebra with tuple time-stamping considers both the time domains. Sarda (1990, 1993) and Lorentzos and Johnson (1988) consider only valid time, whereas Navathe and Ahmed (1989,1993) model transaction time as a user-defined data type.
2. Introduction of a multitude of new operations. Due to the wide variations in the proposed data models, several new operations have been introduced. Many of these operators, such as *PACK*/*UNPACK* or *FOLD*/*UNFOLD*, are useful only in generating alternate views of the same data and do not necessarily add to the expressive power of a language.
3. Introduction of several objects. All proposals introduce a wide variety of objects such as static relations, temporal elements, temporal relations (with time points and with time intervals), etc. As a result, these algebras are multisorted. It should be noted that the snapshot algebra (i.e., conventional relational algebra) is unisorted, since it supports only one type of object, namely a relation. It would be desirable to have a general definition that can encompass all of the possible objects (McKenzie and Snodgrass 1991b).
4. Lack of precise definitions of aggregate functions. McKenzie and Snodgrass (1991a) provide a formal definition for aggregate functions for the attribute time-stamping view. However, such precise definitions do not exist for the tuple time-stamping view. Navathe and Ahmed (1989) describe aggregate functions for extending SQL temporally, but the associated relational algebra is not provided.
5. Lack of formal update semantic. Previous research has not considered algebraic operators for modifications in a temporal database in a formal way.

The objective of this research is to develop a temporal relational algebra based on tuple time-stamping. Unlike alternatives such as the object-oriented data model that lack a proper definition and widespread usage, the relational model has a formal basis in set theory and logic. Relational databases are very widely used because the relational model is well understood in theory and in practice.

Several reasons (e.g., simplicity, ease of implementation) have been provided in the literature claiming superiority of tuple time-stamping (Segev and Shoshani 1988; Navathe and Ahmed 1989). There are several more that we feel are important. We believe that the success of an extension to an already popular model lies in its compatibility with the original model – both theoretically, as well as with respect to a user's perception about it. Temporal relations with tuple time-stamping provide the same "look and feel" as the original, static relations. Attribute time-stamping, in a sense, provides a view similar to the nested relational model (Roth et al. 1988) which, despite its superior expressive power, does not enjoy widespread popularity due to implementation problems. Gadia (1988) raises the issue of *horizontal* and *vertical anomalies* to discredit tuple time-stamping. However, one must note that these anomalies are limitations of the relational model itself and would, therefore, persist in any logical extensions.[2]

Although there have been several proposals in the literature based on tuple time-stamping, each has been inadequate in some respects. In this paper, we propose a rela-

[1] Snodgrass and Ahn (1985) show that there could be two orthogonal time dimensions: *valid time*, when a change occurs in the real world, and *transaction time*, when such a change is recorded in the database

[2] In order to convince oneself, the reader is encouraged to try to represent a nested relation within the flat relational structure

tional algebra that makes the best use of existing results and overcomes all of the significant difficulties. We adopt multidimensional tuple time-stamping to capture the complete time-dependency of temporal data. Our algebra is based on a temporal extension of set theory, and reduces to the snapshot algebra when all relations are static. Valid timestamps are formed by application of a finite number of union operations on multidimensional time intervals. As a result, both valid and transaction time can be supported in this algebra. We provide a general definition of a relation that can represent both static and temporal relations. The traditional relational operations are redefined as consistent extensions, and a new one is introduced. The algebra obviates the need of supporting operations such as *PACK*/*UNPACK* by internalizing them. We also provide precise definitions of aggregate functions and update operations. The algebra is shown to be closed and as expressive as the temporal calculus based language TQuel (Snodgrass 1987).

We call this algebra the *Complete Temporal Relational Algebra*. The term "complete" has been used by different authors to mean different things, and can be misleading. We call ours "complete" to mean that it is at least as expressive as the temporal calculus. (See Sect. 4 for more details.) We believe that it is not significantly different from the original relational algebra, yet addresses all the above issues in a systematic way. This algebra is evaluated against the set of 26 criteria proposed by McKenzie and Snodgrass (1991b), and compared to several other existing algebras.

The remainder of the paper is divided into five sections. Section 2 outlines the basic assumptions and formally defines the relational structure for our algebra. The relational operations are described in Sect. 3. Section 4 discusses the most important properties of this algebra and evaluates it. Section 5 concludes the paper and offers directions for future research.

2 Relational structure

In this section, we formally define the structure of relations for our algebra.

2.1 Representation of time

We assume that the time line is continuous, i.e., isomorphic to the set of nonnegative real numbers (with a linear order "$\leq$"). In other words, it is the metric space of $[0, \infty)$. The most current time is denoted as t_{NOW}. The *time space* is formed by one or more orthogonal time lines. For example, if both transaction time and valid time are supported, then these time lines serve as the basis vectors of the two-dimensional time space. A *time point* t_p is any point in the time space. If the time space is m-dimensional, then a *time interval* t_I in the time space is the set of all points $\tau = (t_1, t_2, \ldots, t_m)$ such that

$$a_i \leq t_i \leq b_i,\ i = 1, 2, \ldots, m;$$

or the set of points which is characterized by the above inequality with any or all of the $\leq$ signs replaced by $<$. The limit a_i is a nonnegative real number, and b_i is either a nonnegative real number or ∞; $a_i = b_i$ is allowed, so a time point may also be represented as an interval. If for some i, $a_i = b_i$, then $t_i = a_i$, i.e., empty intervals are not allowed. If $b_i = \infty$, then $t_i < b_i$, i.e., the interval must be open from the right along the ith dimension. A subset of the time space is called an *elementary subset* if and only if it can be expressed as the union of a finite number of time intervals (Rudin 1976). Thus an elementary subset is a generalization of Gadia's (1988) temporal elements which are only one-dimensional. We use $\mathscr{T}$ to denote the family of all elementary subsets of the time space. By construction, $\mathscr{T}$ is closed under application of finite number of set theoretic operations, such as union, intersection, and difference.

In our algebra, $\mathscr{T}$ is the domain of time attributes – implicit or user-defined. The implicit time attribute is the timestamp, and is always denoted by TS. User-defined time attributes can have a name of a user's choice. If TS is renamed, then it behaves as a user-defined time attribute; i.e., it ceases to be the timestamp attribute.

Several Boolean operations are allowable on time attributes. We list them below.

1. If t_1 and t_2 are m-dimensional time points, then a partial order "$<$" is defined as: $t_1 < t_2$ if $t_{1_i} < t_{2_i}$, $i = 1, 2, \ldots, m$. Other operations such as "$\leq$" and "$=$" can be defined in a similar fashion. If the time points are one-dimensional, this is a total order.
2. If t is a time point and $ts \in \mathscr{T}$, then $t \in ts$ retains usual set-theoretic meaning.
3. If $ts_1, ts_2 \in \mathscr{T}$, then $ts_1 \subset ts_2$ retains its usual set-theoretic meaning, as do other operations such as "$\subseteq$" and "$=$."
4. If $ts_1, ts_2 \in \mathscr{T}$, then
   a) $ts_1 \sqcap ts_2$ (to be read as ts_1 overlaps ts_2) if $\exists t \in ts_1 (t \in ts_2)$.
   b) $ts_1 \prec ts_2$ (to be read as ts_1 precedes ts_2) if $(t_1 \in ts_1) \wedge (t_2 \in ts_2) \Rightarrow (t_1 < t_2)$. ("$\preceq$" can be defined by replacing "$<$" with "$\leq$" in the above definition.)
   c) $ts_1 \lhd ts_2$ (to be read as ts_1 partially precedes ts_2) if $\exists t_1 \in ts_1\ \exists t_2 \in ts_2 (t_1 < t_2)$. (Note that it possible to have $ts_1 \lhd ts_2$ and $ts_2 \lhd ts_1$ simultaneously.)

Example. In the following table, expressions on the left are "true," whereas expression on the right are "false:"

TRUE	FALSE
$[2,3) \subset [1,5)$	$[2,3) \subset [4,5)$
$[2,7) \sqcap [6,9)$	$[2,7) \sqcap [7,9)$
$[2,4) \prec [6,9)$	$[2,7) \prec [6,9)$
$[2,6) \lhd [3,7)$	$[6,9) \lhd [2,4)$

2.2 Representation of tuples and relations

Let $\mathscr{N} = \{1, 2, \ldots, n\}$ be an arbitrary set of integers. A *relation scheme* R is a set of *attribute names* $\{A_1, A_2, \ldots, A_n\}$, one of which may be a special timestamp attribute denoted TS. Corresponding to each attribute name A_i, $i \in \mathscr{N}$, there is a set D_i, called the *domain* of A_i; if A_i is a time attribute, then $D_i = \mathscr{T}$. The *multiset* $\boldsymbol{D} = \{D_1, D_2, \ldots, D_n\}$ is called the domain of R. A *tuple* x over R is a function from R to $\boldsymbol{D}$ such that $x(A_i) \in D_i$, $i \in \mathscr{N}$. In other words, a tuple x

over R can be viewed as a set of attribute name-value pairs: $x = \{\langle A_i, v_i\rangle | \forall i \in \mathscr{N}(A_i \in R \ \wedge \ v_i \in D_i)\}$. If $TS \notin R$, it is said that x has a zero-dimensional timestamp. The following definitions are needed before defining a relation.

Definition 2.1 Let R be any relation scheme. Two tuples x and y on R are *value-equivalent*[3] (written $x \simeq y$ or $y \simeq x$) if and only if all non-timestamp attribute values are the same in both the tuples. Symbolically, $x \simeq y \ \Leftrightarrow \ \forall A \in (R - \{TS\})(y(A) = x(A))$.

Value-equivalent tuples are analogous to *duplicates* in the snapshot algebra. Duplicates are not allowed in a legal relation in the conventional relational model, and similarly value-equivalent tuples are not allowed in a legal temporal relation; they must be *coalesced* into a single tuple.

Definition 2.2 The *coalescence* operation[3] (denoted by $\oplus$) on two value-equivalent tuples x and y on relation scheme R can be defined as

$$z = x \oplus y \ \Leftrightarrow \ (x \simeq y) \wedge (z \simeq x) \\ \wedge (TS \in R \Rightarrow z(TS) = x(TS) \cup y(TS))$$

Example. The tuples $\langle 3025, [1,5], 15\text{K}\rangle$ and $\langle 3025, [4,7], 15\text{K}\rangle$ are value-equivalent. If they are coalesced, the resulting tuple would be $\langle 3025, [1,7], 15\text{K}\rangle$.

The idea of value-equivalent tuples and coalescence operation need not be confined to just two tuples. Given m tuples $x_1, x_2, \ldots, x_m$, all of which are on the same relation scheme, they are said to be value-equivalent if $x_i \simeq x_j$ for all i, j; $1 \leq i, j \leq m$. The coalescence operation on all the m value-equivalent tuples will recursively coalesce all the tuples pairwise to produce a tuple that has a timestamp produced by the union of timestamps of all the tuples. In other words, if $x_1, x_2, \ldots, x_m$ are value-equivalent, then

$$\bigoplus_{i=1}^{m} x_i = (\ldots((x_1 \oplus x_2) \oplus x_3) \oplus \ldots \oplus x_{m-1}) \oplus x_m$$

Definition 2.3 A *temporal set* is a collection of tuples, such that no two of its members are value-equivalent.

A relation in this algebra can now be defined as follows.

[3] Snodgrass (1987) introduced the concept of *value-equivalent* tuples. He also proposed that such tuples be *coalesced* if their timestamps are overlapping or adjacent. Our coalescence operator does not have this extra restriction. Also note that the *FOLD* operation of Lorentzos and Johnson (1988), the *COMPRESS* operator of Navathe and Ahmed (1989), and the *CONTRACT* operator of Sarda (1990) – are all similar to the coalescence operator. However, there is a basic difference between their operators and ours. For example, Navathe and Ahmed use the *COMPRESS* operator on a relation (a set of tuples) so that all overlapping or adjacent value-equivalent tuples reduce to a single tuple. In other words, the *COMPRESS* operator is one of the allowable operations within the algebra. A similar observation is true for the other operators as well. On the other hand, a user of our algebra is not allowed to use coalescence as one of the relational operations. Every time there is a possibility of generating value-equivalent tuples (due to application of a relational operation), the coalescence operator is automatically used so that no value-equivalent tuples are present in the resulting relation. Thus, analogous to removal of duplicates in the static algebra, coalescence is an intrinsic part of the definition of all operations in our temporal algebra

Table 1. Example relations for an employee database

Relation: PERSON

SSN	LName	FName
086630763	Lyons	James
980678976	Kivari	Jack
229767329	Myers	Peter

Relation: PROJECT

PROJ#	Name	Client
12345	proj A	Mobil
11233	proj B	Kodak
11234	proj C	Kodak

Relation: EMPLOYEE

EMP#	ssn
3025	086630763
6637	980678976

Relation: SALARY

EMP#	TS	Salary
3025	[1,5)	15K
3025	[5,9)	20K
6637	[3,10)	17K
6637	[10,12)	19K

Relation: SALARY′

EMP#	TS	Salary
3025	[2,3]	15K
3025	(7,20)	20K
6637	[12,15)	19K

Relation: DEPT

EMP#	TS	Dept
3025	[1,3)	dep1
3025	[3,9)	dep2
6637	[3,5)	dep2
6637	[5,11)	dep3
6637	[11,12)	dep1

Relation: ASSIGNMENT

EMP#	PROJ#	TS
3025	12345	[1,4)
3025	11233	[4,6)
3025	11234	[5,10]
6637	11233	[3,10]
6637	11234	[8,12)

Definition 2.4 A *relation* r on the scheme R is a finite temporal set of tuples x on R.

Note that by definition of a temporal set, value-equivalent tuples are not allowed in a legal relation. A relation can be either *static* (i.e., a relation on a scheme without timestamps) or *temporal* (i.e., a relation on a scheme with a timestamp). The above definition of a relation is general enough to include both possibilities, and in the next section, the relational operations are defined in such a manner that both types of relations can be supported. A few sample relations are presented in Table 1; these relations will be used for illustrating some of the relational operations. Since TS, the timestamp of any tuple, can represent multiple orthogonal time domains, this algebra can support transaction as well as valid (one or more) time domains. For simplicity, however, only valid time is used in most of the illustrations.

There are two notions of a relation (Ullman 1988): (1) *set-of-lists* notion, where the order of the columns is important, and (2) *set-of-mappings* notion, where the order is not. In our definition of a relation, the set-of-mappings definition of a relation has been adopted. It is known, however, that one notion can be converted to the other rather easily (Ullman 1988). Before describing the temporal relational operations, the following definitions are necessary.

Defininition 2.5 Let R be any relation scheme. A tuple y on R is said to *temporally imply* a tuple x on R (written $y \xrightarrow{t} x$) if $y = y \oplus x$.

Definition 2.6 A tuple x is a *temporal member* of a relation r on scheme R (written $x \in_t r$) if $\exists y \in r(y \xrightarrow{t} x)$.

Note that, although a nonempty relation can possibly have infinitely many temporal members, the relation can always be represented as a finite collection of tuples. For example, the relation SALARY in Table 1 has an infinite number of temporal members; it can, however, be represented as a collection of four tuples.

Definition 2.7 A tuple x is a *partial temporal member* of a relation r on scheme R (written $x \in_{pt} r$) if $\exists y \in r \exists z((y \xrightarrow{t} z) \wedge (x \xrightarrow{t} z))$.

Definition 2.8 A relation r_1 on a scheme R is said to be a *temporal subset* of a relation r_2 on the same scheme (written $r_1 \subset_t r_2$) if and only if every tuple $x \in r_1$ is a temporal member of r_2.

Example. The tuple $\langle 3025, [1,5), 15\text{K}\rangle$ temporally implies the tuple $\langle 3025, [2,3], 15\text{K}\rangle$. The tuples $\langle 3025, [1,5), 15\text{K}\rangle$ and $\langle 3025, [2,3], 15\text{K}\rangle$ are both temporal members of the relation SALARY. The first tuple is not a temporal member of SALARY′, but is a partial temporal member of SALARY′.

3 Relational operations and the relational algebra

The basic relational operations – union, difference, join, projection, and selection – can now all be outlined with the help of the above definitions. One additional operation, namely temporal projection, is also introduced. Other relational operations (e.g., intersection, Cartesian product, division) can also be defined in terms of the basic operations. Each operation is defined to be a consistent extension of its counterpart in the snapshot algebra. It can also be verified that each operation reduces to the conventional definition of that operation in the snapshot algebra.

For economy of expression, the definitions are given in terms of temporal membership (as defined in the last section). However, since there can be an infinite number of temporal members in a nonempty relation, it may appear that these operations cannot be efficiently computed; this is why we also show how these operations can be computed in an efficient manner.

Temporal relational operations are distinguished from the conventional static relational operations by the use of a *hat* (ˆ) over the symbols of the former.

Union ($\hat{\cup}$). Let r_1 and r_2 be two relations on the relation scheme R. The *union* of these two relations, denoted by $r_1 \hat{\cup} r_2$, is a temporal set of tuples on R such that

$$x \in_t (r_1 \hat{\cup} r_2) \Leftrightarrow (x \in_t r_1) \vee (x \in_t r_2) \vee (\exists y \in_t r_1 \exists z \in_t r_2 (y \oplus z = x))$$

For example, the union of SALARY and SALARY′ is shown in Table 2a. It can easily be verified that the union operation is associative and commutative. The union operation can be easily computed using the following algorithm:[4]

[4] In this algorithm, and in all others that follow, we have assumed that the relations can fit in the main memory. If this is not the case, the algorithms can be easily modified

Table 2a,b. Examples of union and difference operations

a SALARY $\hat{\cup}$ SALARY′

EMP#	*TS*	Salary
3025	[1,5)	15K
3025	[5,20)	20K
6637	[3,10)	17K
6637	[10,15)	19K

b SALARY $\hat{-}$ SALARY′

EMP#	*TS*	Salary
3025	[1,2)∪(3,5)	15K
3025	[5,7]	20K
6637	[3,10)	17K
6637	[10,12)	19K

Algorithm union
input: relations r_1 and r_2 on scheme R
output: relation $s = r_1 \hat{\cup} r_2$ on scheme R

```
begin
    $s(R) \leftarrow r_1$;
    for all $x \in r_2$ do
        for all $y \in s$ do
            if $(x \simeq y)$ then
                $y \leftarrow (x \oplus y)$
            else $s \leftarrow s \cup \{x\}$;
end;
```

Difference ($\hat{-}$). Let r_1 and r_2 be as above. The *difference* of these two relations, denoted by $r_1 \hat{-} r_2$, is a temporal set of tuples on R such that

$$x \in_t (r_1 \hat{-} r_2) \Leftrightarrow (x \in_t r_1) \wedge (x \notin_{pt} r_2)$$

For example, the difference, SALARY $\hat{-}$ SALARY′ is shown in Table 2b. The following algorithm implements the difference operation:

Algorithm difference
input: relations r_1 and r_2 on scheme R
output: relation $s = r_1 \hat{-} r_2$ on scheme R

```
begin
    $s(R) \leftarrow r_1$;
    for all $x \in r_2$ do
        for all $y \in s$ do
            if $(x \simeq y)$ then
                if $(TS \in R) \wedge (y(TS) \not\subseteq x(TS))$ then
                    $y(TS) \leftarrow y(TS) - x(TS)$
                else $s \leftarrow s - \{y\}$;
end;
```

Natural join ($\hat{\bowtie}$). Let r_1 and r_2 be any two relations on schemes R_1 and R_2 respectively. The *natural join* of r_1 and r_2, denoted $r_1 \hat{\bowtie} r_2$, is a temporal set of tuples on the relation scheme $S = R_1 \cup R_2$ such that

$$x \in_t (r_1 \hat{\bowtie} r_2) \Leftrightarrow ; \exists y \in_t r_1 (y = x(R_1)) \wedge \exists z \in_t r_2 (z = x(R_2))$$

For example, the natural join between EMPLOYEE, SALARY and DEPT is shown in Table 3. It can easily be verified that the natural join operation is associative and commutative. The natural join between any two relations can be computed using the following algorithm:

Algorithm natural join
input: relations r_1 on scheme R_1 and r_2 on scheme R_2
output: relation $s = r_1 \hat{\bowtie} r_2$ on scheme $S = R_1 \cup R_2$

Table 3. Relation: EMPLOYEE′ = EMPLOYEE $\hat{\bowtie}$ SALARY $\hat{\bowtie}$ DEPT

EMP#	ssn	TS	salary	dept
3025	086630763	[1,3)	15K	dep1
3025	086630763	[3,5)	15K	dep2
3025	086630763	[5,9)	20K	dep2
6637	980678976	[3,5)	17K	dep2
6637	980678976	[5,10)	17K	dep3
6637	980678976	[10,11)	19K	dep3
6637	980678976	[11,12)	19K	dep1

```
begin
    $Q \leftarrow (R_1 \cap R_2) - \{TS\}$;
    $s(S) \leftarrow \emptyset$;
    for all $x \in r_1$ do
        for all $y \in r_2$ do
            if  $(x(Q) = y(Q))$  then
                if  $(TS \in (R_1 \cap R_2)) \wedge (x(TS) \cap y(TS) \neq \emptyset)$  then
                    $s \leftarrow s \cup \{\langle x(R_1 - \{TS\}), y(R_2 - R_1), x(TS)$
                        $\cap\, y(TS)\rangle\}$
                else if $(x(TS) \notin (R_1 \cap R_2))$  then
                        $s \leftarrow s \cup \{\langle x(R_1), y(R_2 - R_1)\rangle\}$;
end;
```

Projection ($\hat{\Pi}$). Let r be a relation on the scheme R. Let $S \subset R$ be any other relation scheme. The *projection* of r onto S, written $\hat{\Pi}_S(r)$ is the temporal set of tuples on S such that

$$x \in_t \hat{\Pi}_S(r) \;\Leftrightarrow\; \exists q \subset_t r \left(x = \bigoplus_{y \in q} y(S) \right)$$

For example,

$$\text{EMPLOYEE} = \hat{\Pi}_{\{\text{EMP\#,ssn}\}}(\text{EMPLOYEE}'),$$
$$\text{SALARY} = \hat{\Pi}_{\{\text{EMP\#},TS,\text{salary}\}}(\text{EMPLOYEE}'),$$
$$\text{DEPT} = \hat{\Pi}_{\{\text{EMP\#},TS,\text{dept}\}}(\text{EMPLOYEE}'),$$

where the relation EMPLOYEE′ is as given in Table 3. For relation schemes Q, R and S with $S \subset R \subset Q$, it can easily be verified that, if r is any relation on scheme Q, then

$$\hat{\Pi}_S(\hat{\Pi}_R(r)) = \hat{\Pi}_S(r)$$

The following algorithm computes the projection operation:

Algorithm projection
input: relation r on scheme R and a subscheme $S \subset R$.
output: relation $s = \hat{\Pi}_S(r)$ on scheme S

```
begin
    $s(S) \leftarrow \emptyset$;
    for all $x \in r$ do
        $s \leftarrow s \hat{\cup} \{x(S)\}$;
end;
```

Selection ($\hat{\sigma}$). Let r be a relation on the scheme R. Let Θ be a set of comparators over domains of attribute names in R. Let P be a predicate (called the selection predicate) formed by attributes in R, comparators in Θ, constants in dom(A), $A \in R$, and logical connectives. The *selection* on r for P, written $\hat{\sigma}_P(r)$, is a temporal set $\{x \in r | P(x)\}$. Since this definition is already in conventional set-theoretic notation, we do not provide an algorithm for selection.

Table 4. Example of selection and temporal projection operations

a $\hat{\sigma}_{\text{EMP\#=3025}}(\text{DEPT})$

EMP#	TS	Dept
3025	[1,3)	Dep1
3025	[3,9)	dep2

b $\hat{\Upsilon}_{[2,6)}(\hat{\sigma}_{\text{EMP\#=3025}}(\text{DEPT}))$

EMP#	TS	Dept
3025	[2,3)	Dep1
3025	[3,6)	Dep2

The selection on DEPT for “EMP# = 3025” is shown in Table 4a. The reader may verify that for any two selection predicates P_1 and P_2 involving attributes of R,

$$\hat{\sigma}_{P_1}(\hat{\sigma}_{P_2}(r)) = \hat{\sigma}_{P_2}(\hat{\sigma}_{P_1}(r)) = \hat{\sigma}_{P_1 \wedge P_2}(r)$$

In the temporal model, the selection predicate P may contain temporal conditions. Θ contains, in addition to the usual comparators, temporal comparators discussed in Sect. 2.1. A wide variety of selection predicates can be formulated using these comparators.

Temporal projection ($\hat{\Upsilon}$). Let r be a relation on the scheme R and $T \in \mathscr{T}$. The *temporal projection* of r during T, denoted $\hat{\Upsilon}_T(r)$, is a temporal set of tuples on R such that

$$x \in_t \hat{\Upsilon}_T(r) \;\Leftrightarrow\; (x \in_t r) \wedge (TS \in R \Rightarrow x(TS) \subset T)$$

For example, the temporal projection of the relation in Table 4a during [2,6) is shown in Table 4b. Clearly, if $TS \notin R$, then $\hat{\Upsilon}_T(r) = r$, where r is a relation on R. The algorithm for temporal projection is given below:

Algorithm temporal projection
input: relations r on scheme R and $T \in \mathscr{T}$
output: relation $s = \hat{\Upsilon}_T(r)$ on scheme R

```
begin
    $s(R) \leftarrow \emptyset$;
    for all $x \in r$ do
        if  $(TS \in R) \wedge (x(TS) \cap T \neq \emptyset)$  then
            $s \leftarrow s \cup \{\langle x(R - TS), x(TS) \cap T\rangle\}$
        else if $(x(TS) \notin (R_1 \cap R_2))$  then
                $s \leftarrow s \cup \{x\}$;
end;
```

Our temporal projection is similar to the time-slice operation of Clifford and Crocker (1987) and Navathe and Ahmed (1989) and the temporal selection operation of Gadia (1988). Since there is an added dimension of time in the temporal extension of the relational model, we need the temporal projection operation to restrict a relation along the time dimension.

Note that the rollback and snapshot operations (McKenzie and Snodgrass 1991b; Snodgrass and Ahn 1985) can be obtained by using the projection and temporal projection operations jointly. For example, consider a relation r on a scheme R such that $TS \in R$, with TS two-dimensional. Let $R' = R - \{TS\}$. Then the rollback operation on r to a transaction time point t would produce the relation $r' = \hat{\Upsilon}_{[0,+\infty)\times[t,t]}(r)$.

The snapshot of r' at a valid time-point t' as of t would be the relation $r'' = \hat{\Pi}_{R'}\left(\hat{\Upsilon}_{[t',t']\times[t,t]}(r)\right)$.

Rename ($\hat{\rho}$). The *rename* operation ($\hat{\rho}$) is used to change the names of some attributes of a relation. Let r be a relation on scheme R, where A and B are attributes satisfying $A \in R$ and $B \notin R$. Let A and B have the same domain, and let $R' = (R - A) \cup B$. Then r with A

renamed to B, written $\hat{\rho}_{A \leftarrow B}(r)$, is the set of tuples (on R') $\{y | \exists x \in r((y(R'-B) = x(R-A)) \wedge (y(B) = x(A)))\}$. Thus, the rename operation remains the same (as in snapshot algebra) in our algebra. One must be careful when renaming the timestamp attribute TS, because if it is renamed, it will no longer be treated as the implicit timestamp; rather it will be a user-defined time attribute.

3.1 The relational algebra

Given the above operations, the relational algebra can now be defined formally in a fashion similar to Maier (1983):

Definition 3.1 (Relational Algebra) Assume that $\boldsymbol{U}$ is a set of attribute names, called the *universe*. $\boldsymbol{U}$ may have timestamp TS as its element. Let $\mathscr{D}$ be a set of domains, and let **dom** be a total function from $\boldsymbol{U}$ to $\mathscr{D}$. Let $\boldsymbol{R} = \{R_1, R_2, \ldots, R_p\}$ denote a set of distinct relation schemes, where $R_i \subset \boldsymbol{U}$, for $1 \leq i \leq p$. Let $d = \{r_1, r_2 \ldots, r_p\}$ be a set of relations, such that r_i is a relation on R_i, $1 \leq i \leq p$. Θ denotes a set of comparators over domains in $\mathscr{D}$. The *relational algebra* over $\boldsymbol{U}$, $\mathscr{D}$, **dom**, $\boldsymbol{R}$, d and Θ is the seven-tuple $\mathscr{R} = (\boldsymbol{U}, \mathscr{D}, \mathbf{dom}, \boldsymbol{R}, d, \Theta, O)$, where O is the set of operators union, difference, natural join, projection, selection, temporal projection and rename using attributes in $\boldsymbol{U}$ and comparators in Θ, and logical connectives. An *algebraic expression* over $\mathscr{R}$ is any expression formed legally (according to the restrictions on the operators) from the relations in d and constant relations over schemes in $\boldsymbol{U}$, using the operators in O.

Let *sch*(E) be the *scheme* of an algebraic expression E. We can define *sch*(E) recursively as follows:

1. If E is r_i, then *sch*(E) = R_i.
2. If E is a constant relation, then *sch*(E) is the scheme for that relation.
3. If $E = E_1 \hat{\cup} E_2$, $E_1 \hat{-} E_2$, $\hat{\sigma}_P(E_1)$, or $\hat{\Upsilon}_T(E_1)$, where P is a selection predicate, and $T \in \mathscr{T}$, then *sch*(E)=*sch*(E_1).
4. If $E = \hat{\Pi}_S(E_1)$, then *sch*(E) = S.
5. If $E = E_1 \hat{\bowtie} E_2$, then *sch*($E$)=*sch*($E_1$)∪*sch*($E_1$).
6. If $E = \rho_{A \leftarrow B}(E_1)$, then *sch*($E$)=(*sch*($E_1$) $- A) \cup B$.

3.2 Other relational operations

It is also possible to express the other relational operations, such as intersection, Cartesian product, theta-join and division in terms of the basic relational operations discussed above.

Intersection ($\hat{\cap}$). Let r_1 and r_2 be two relations on the relation scheme R. The intersection of these two relations, denoted $r_1 \hat{\cap} r_2$, is a temporal set of tuples on R such that

$$x \in_t (r_1 \hat{\cap} r_2) \;\Leftrightarrow\; (x \in_t r_1) \wedge (x \in_t r_2)$$

Alternatively, it can easily be verified that

$$r_1 \hat{\cap} r_2 = r_1 \hat{-} (r_1 \hat{-} r_2)$$

Cartesian product ($\hat{\times}$). The Cartesian product of two relations can be defined as a special case of a natural join (Codd 1990, p. 66) where the relations do not have any common column (with the possible exception of the timestamp TS). Let R_1 and R_2 be two relation schemes such that $(R_1 \cap R_2) - \{TS\} = \emptyset$. Let r_1 and r_2 be any two relations on schemes R_1 and R_2 respectively. Then, the Cartesian product of r_1 and r_2 is a relation on the scheme $(R_1 \cup R_2)$ defined as:

$$r_1 \hat{\times} r_2 = r_1 \hat{\bowtie} r_2$$

Note that the criterion $(R_1 \cap R_2) - \{TS\} = \emptyset$, necessary in the definition of Cartesian product, is not a limitation, since any common attribute can be renamed before this operation is applied.

Theta-join ($\hat{\bowtie}_\Theta$). The theta-join operation can be thought of as a Cartesian product between two relations (with common attributes except timestamp renamed when necessary) followed by a selection operation, where the selection predicate contains θ-comparable comparators (Maier 1983). Let r_1 and r_2 be as above, and $R = R_1 \cup R_2$. Let Θ and P be as in the definition of the *selection* operation. Then the theta-join between r_1 and r_2, written $r_1 \hat{\bowtie}_P r_2$), is given as:

$$r_1 \hat{\bowtie}_P r_2 = \hat{\sigma}_P(r_1 \hat{\times} r_2)$$

Division ($\hat{\div}$). Let r_1 and r_2 be any two relations on schemes R_1 and R_2 respectively, where $R_2 \subset R_1$. Let $R' = R_1 - (R_2 - \{TS\})$. The division of r_1 by r_2, denoted $r_1 \hat{\div} r_2$, is a maximal temporal subset of $\Pi_{R'}(r_1)$ which, when joined (natural join) with r_2, produces a temporal subset of r_1. Alternatively, the division operation may be expressed as:

$$r_1 \hat{\div} r_2 = \hat{\Pi}_{R'}(r_1) \;\hat{-}\; \hat{\Pi}_{R'}\left(\left(\hat{\Pi}_{R'}(r_1) \hat{\bowtie} r_2\right) \hat{-} r_1\right)$$

3.3 Modification of the database

In addition to data extraction queries, a query language also has to support modification of the database. The relational algebra should then support operations such as insertion, deletion and updating. Each of these is discussed below.

Insertion. An insertion into a relation r is expressed as $r \leftarrow r \hat{\cup} E$, where E is any relational expression such that the schemes of r and E are the same. Since the union ($\hat{\cup}$) operation has been redefined, the above expression can handle insertion into static as well as temporal relations.

Deletion. A deletion from a relation r is expressed as $r \leftarrow r \hat{-} E$, where E is any relational expression on the same scheme as that of r. Since the difference operation has been redefined, such expressions will work with static, as well as temporal relations. The deletion operation need not necessarily delete all historical information. Careful specification of E can retain all the past information, while ensuring that no more information is added in the future. More specifically, for two-dimensional timestamps, E can be specified as $E = \hat{\Upsilon}_T(\sigma_P(r))$, where P is the selection predicate specifying the delete condition and $T = [0, \infty) \times [t_{\text{NOW}}, \infty)$; the first axis of T is the valid time, and the second the transaction time.

Consider, as an example, the relation DEPT1 shown in Table 5a, where the timestamp is two-dimensional with valid and transaction time. The first tuple in this relation should be interpreted as (Snodgrass and Ahn 1985): "*The Employee with EMP# 3025 worked in dep1 from 2 to ∞ (valid time) as of 1 through 10 (transaction time), and from 2 to 9 as of 10 through ∞.*" Now at time 15, if we know that the same employee has resigned, we could perform the following delete operation:

$$\text{DEPT1} \mathbin{\hat{-}} \hat{\Upsilon}_{[0,\infty)\times[t_{\text{NOW}},\infty)}(\sigma_{\text{EMP\#=3025}}(\text{DEPT1}))$$

The resulting relation is shown as DEPT2 in Table 5b. Note that the system should automatically assign a value of 15 to t_{NOW}.

Updating. The update operation is used when some values in a tuple need to be changed, while retaining the other values that we do not wish to change. Such changes cannot always be made by using only the *deletion* and *insertion* operations. The *update* operator (denoted by $\hat{\delta}$) is redefined in this algebra. This operation is expressed as $\hat{\delta}_{(A\leftarrow E),T}(r)$ where r is the name of a relation with attribute A, which is assigned, for all time points in $T \in \mathscr{T}$, the value of an arithmetic expression E involving constants and attributes in R (the scheme of r). The semantics of the update operation is given by:

$$\begin{aligned} x \in_t \hat{\delta}_{(A\leftarrow E),T}(r) \Leftrightarrow & \big(x \in_t (r \mathbin{\hat{-}} \hat{\Upsilon}_T(r))\big) \\ & \vee\big(\exists y \in_t \hat{\Upsilon}_T(r)(x(R') \simeq y(R')) \\ & \wedge(x(A) = E(y))\big) \end{aligned}$$

where $R' = R - \{A\}$. If r is static, $\hat{\Upsilon}_T(r) = r$, and the definition would reduce to the definition of static update where values are simply overwritten.

To illustrate, we reconsider the relation DEPT1 shown in Table 5a. Assume that a piece of information arrives at time 15 indicating that effective time 16, 3025 will be transferred to dep3. The update operation should be specified as:

$$\hat{\hat{\delta}}_{(\text{dept}\leftarrow\text{dep3}),[16,\infty)\times[t_{\text{NOW}},\infty)}(\text{DEPT1})$$

and would result in a relation DEPT3 as shown in Table 5c.

In the definition of deletion and update operations, we specified the transaction time component for clarity of expression; it could be automatically supplied by the system.

3.4 Aggregate functions

Commercial database management systems typically provide *aggregate* functions such as `max`, `min`, `sum`, `avg`, `count`, etc., and an *aggregate* clause `group by`. Obviously, it would also be useful to have temporal aggregate functions supported in a temporal database management system.

Two types of aggregates have been discussed in the literature (McKenzie and Snodgrass 1991a): (1) unique aggregates, where duplicate tuples are eliminated, and (2)

Table 5. Examples of delete and update operations

a Relation: DEPT1

EMP#	*TS*	Dept
3025	$[2,\infty)\times[1,10) \cup [2,9)\times[10,\infty)$	dep1
3025	$[9,\infty)\times[10,\infty)$	dep2
⋮	⋮	⋮

b Relation: DEPT2

EMP#	*TS*	Dept
3025	$[2,\infty)\times[1,10) \cup [2,9)\times[10,15)$	dep1
3025	$[9,\infty)\times[10,15)$	dep2
⋮	⋮	⋮

c Relation: DEPT3

EMP#	*TS*	Dept
3025	$[2,\infty)\times[1,10) \cup [2,9)\times[10,\infty)$	dep1
3025	$[9,\infty)\times[10,15) \cup [9,16)\times[15,\infty)$	dep2
3025	$[16,\infty)\times[15,\infty)$	dep3
⋮	⋮	⋮

nonunique aggregates, where duplicates are retained for calculation of aggregates. It must be noted that the case of unique aggregates does not need any special treatment, since our algebra, by default, removes duplicates and coalesces overlapping tuples. It will be shown, by way of examples, how unique aggregates can be calculated by slight variation of the approach used for nonunique aggregates.

Klug (1982) provides a formal basis for nonunique aggregates in the snapshot relational algebra. Most of his ideas can be extended in the temporal relational algebra. Aggregates in temporal databases have been examined by several researchers (Ben-Zvi 1982; Navathe and Ahmed 1989; McKenzie and Snodgrass 1991a; Tansel 1987). These efforts indicate the necessity for obtaining a distribution of values over time. This aspect is modeled by adopting the idea of a *moving aggregation window* from TSQL (Navathe and Ahmed 1989), and refining the overall scheme with precise definitions.

3.4.1 Basic aggregate functions

Let $\boldsymbol{g} = \{\texttt{min}, \texttt{max}, \texttt{sum}, \texttt{avg}, \ldots\}$ be a set of aggregate functions. Let r be any relation on the scheme R. Let $R' = R - \{TS\}$. For any $g \in \boldsymbol{g}$, $g(r)$ returns an aggregate tuple on R', i.e., g is a function from the set of instances of the relation scheme R to the domain of R'. If the aggregate function g is not defined on the domain of an attribute name $A \in R'$, then a special value Ω is returned for that attribute. Before these aggregate functions can be described, we must define a *measure function* (Rudin 1976) μ on $\mathscr{T}$ in the following manner:

1. If t_I is an m-dimensional interval, then t_I is the set of points $\tau = (t_1, t_2, \ldots, t_m)$ such that $a_i \leq t_i \leq b_i$, $i = 1, 2, \ldots, m$, and one or more of the $\leq$ signs may be

replaced by $<$. Then, $\mu(t_I) = \prod_{i=1}^{m}(b_i - a_i + \varepsilon)$, where ε is a very small number (say, 10^{-12}). In the case where $b_i = \infty$, b_i should be replaced by t_{NOW} in calculation of the function μ. If t_p is an m-dimensional time point, then it can be treated as a degenerate interval, and $\mu(t_p) = \varepsilon^m$.

2. If $T \in \mathscr{T}$, then T can be represented as the union of a finite number of disjoint intervals, i.e., $T = \cup_{i=1}^{k} t_{Ii}$, where $t_{Ii} \cap t_{Ij} = \emptyset$ if $i \neq j$. Then, $\mu(T) = \sum_{i=1}^{k} \mu(t_{Ii})$.

Now the aggregate functions can be defined in the following manner. The minimum function on r, written $\texttt{min}(r)$, returns a relation containing a single tuple x on R' such that $x(A) = \min_{y \in r} y(A)$, for all $A \in R'$. Similarly, $\texttt{max}(r)$ returns a relation with the tuple x on R' such that $x(A) = \max_{y \in r} y(A)$, and $\texttt{sum}(r)$ returns a relation with the tuple x on R' such that $x(A) = \sum_{y \in r} y(A)$, for all $A \in R'$. The average function on r is written as $\texttt{avg}(r)$, and returns a relation with a single tuple x on R' such that, for all $A \in R'$,

$$x(A) = \frac{\sum_{y \in r} y(A)\mu(y(TS))}{\sum_{y \in r} \mu(y(TS))}$$

It has been assumed that a relation r has zero-dimensional timestamps if its scheme R does not contain TS; as a result, the above definition is applicable to static relations as well. If any one of the above mentioned functions is not defined on an attribute $A \in R'$, then $x(A) = \varOmega$. Other aggregate functions can be defined in a similar manner and added to the set $\boldsymbol{g}$. Now, the aggregate function on a set of attributes is defined as $g_X(r)$ which represents the nonunique aggregate g for the attributes in $X \subset R'$ as

$$g_X(r) = \hat{\varPi}_X(g(r))$$

For $\texttt{count}$, a slightly different approach is taken. The $\texttt{count}$ function on r returns, for each non-timestamp attribute in its scheme, an integer k given by

$$k = \sum_{y \in r}(1)$$

In other words, $\texttt{count}$ is a function from the set of instances of the relation scheme R to the set of nonnegative integers. The $\texttt{count}$ function on r for an attribute in R' is the same number k.

3.4.2 Aggregate functions on partitions of relations

It may be desirable to calculate aggregates on several partitions of a relation that are partitioned based on a set of attributes. This is equivalent to the *group by* clause in SQL. Two types of partitioning is defined: (1) partitioning based on nontemporal attributes which is equivalent to *group by* in SQL, and (2) partitioning based on temporal attributes which may be viewed as a *temporal group by*.

Group by. The aggregate g for the attributes in $X \subset R'$ of relation r *grouped by* attributes in Y, $Y \subset R'$, $X \cap Y = \emptyset$, is written as $g_X^Y(r)$, and is given by the following relation on the scheme $(X \cup Y)$:

$$g_X^Y(r) = \hat{\bigcup_{x \in \hat{\varPi}_Y(r)}} \{x\} \hat{\bowtie} \{g_X(\{x\} \hat{\bowtie} r)\}$$

Temporal group by. This clause allows us to find a distribution of aggregate values over time, based on a *moving window* (Navathe and Ahmed 1989) of ω. The aggregate g for the attributes in $X \subset R'$ of relation r on scheme R, $TS \in R$, aggregated over an m-dimensional temporal window ω is written as $g_X^\omega(r)$, and is given by the following relation on the scheme $(X \cup \{TS\})$:

$$g_X^\omega(r) = \hat{\bigcup_{\tau \in \varGamma}} \{\tau\} \hat{\bowtie} \{g_X(\hat{\varUpsilon}_\tau(r))\}$$

where $\varGamma$ is the collection of time intervals $\{\tau_1, \tau_2, \ldots, \tau_q\}$ such that the following hold:

1. $\bigcup_{i=1}^{q} \tau_i = t_I$, where t_I is the smallest m-dimensional interval containing $\hat{\varPi}_{\{TS\}}(r)$.
2. $\inf(t_I) = \inf(\tau_1)$, and $\sup(t_I) = \sup(\tau_q)$.
3. $\tau_i \cap \tau_j = \emptyset$, $i \neq j$, $1 \leq i, j \leq q$.
4. If τ_{ij} is the j-th component interval of τ_i, and if ω_j is the j-th component of ω, then $\mu(\tau_{ij}) = \omega_j$, for all i, j; $i = 1, 2, \ldots, q-1$; $j = 1, 2, \ldots, m$.

The above conditions ensure that the entire time horizon of the relation is partitioned into smaller subintervals of the size of the moving window ω. The aggregate function is then performed repeatedly on the temporal projection of the relation during each of these subintervals. We allow an m-dimensional moving window; however, if the distribution of values is required only over valid time as of a time-point in the transaction time domain, it is possible to rollback the relation using projection and temporal projection operations, and then take the aggregate over the resulting relation.

We show a few examples now to illustrate how different types of aggregate functions can be expressed in terms of the formalism described above.

Example query 1. *How many clients are there?* This is equivalent to counting the total number of unique clients in the relation PROJECT.

$$\texttt{count}\left(\hat{\varPi}_{\{\text{client}\}}(\text{PROJECT})\right) = 2$$

Example query 2. *What was the minimum salary during* $[2, 3)$*?*

$$\texttt{min}_{\text{salary}}\left(\hat{\varUpsilon}_{[2,3)}(\text{SALARY})\right) = \{15K\}$$

Example query 3. *Find the average salary grouped by employee.*

$$\texttt{avg}_{\text{salary}}^{\text{EMP\#}}(\text{SALARY}) = \{\langle 3025, 17.50\rangle, \langle 6637, 17.44K\rangle\}$$

Example query 4. *Find the average salary of all employees over the moving window of 2.* This is a temporal group by, and can be expressed as $\texttt{avg}_{\text{salary}}^{2}(\text{SALARY})$. The interval [1,12), the time horizon of the relation, is first computed, and then partitioned to smaller subintervals of measure 2. The desired aggregate is then computed within each subinterval. The result is shown in Table 6.

Table 6. Distribution of average salary over time (moving window = 2)

TS	Salary
$[1,3)$	15K
$[3,5)$	16K
$[5,9)$	18.5K
$[9,11)$	18K
$[11,12)$	19K

4 Properties of complete temporal relational algebra

In this section, several properties of the proposed temporal relational algebra are described. This algebra is closed, meaning that all of the algebraic operators produce valid objects; in this case relations. It is also a consistent extension of the snapshot algebra, and reduces to the latter when time is not part of the relation schemes. It is as expressive as the temporal calculus based query language TQuel (Snodgrass 1987). This algebra also supports the basic algebraic equivalences of the snapshot algebra. Some of these algebraic equivalences are stated in the following theorem, and then the other properties examined.

Theorem 4.1 *Let Q, R and S be three relation schemes with $S \subset R$. Let q be a relation on scheme Q, and let r, r_1, r_2 be relations on scheme R. Let P be any selection predicate involving attributes of R, and let T be any value from $\mathscr{T}$, the domain of TS. Then, the following are identities:*

(a) $q \hat{\bowtie} (r_1 \hat{\cup} r_2) = (q \hat{\bowtie} r_1) \hat{\cup} (q \hat{\bowtie} r_2)$, (b) $q \hat{\bowtie} (r_1 \hat{-} r_2) = (q \hat{\bowtie} r_1) \hat{-} (q \hat{\bowtie} r_2)$,
(c) $\hat{\sigma}_P(r_1 \hat{\cup} r_2) = \hat{\sigma}_P(r_1) \hat{\cup} \hat{\sigma}_P(r_2)$, (d) $\hat{\sigma}_P(r_1 \hat{-} r_2) = \hat{\sigma}_P(r_1) \hat{-} \hat{\sigma}_P(r_2)$,
(e) $\hat{\Upsilon}_T(r_1 \hat{\cup} r_2) = \hat{\Upsilon}_T(r_1) \hat{\cup} \hat{\Upsilon}_T(r_2)$, (f) $\hat{\Upsilon}_T(r_1 \hat{-} r_2) = \hat{\Upsilon}_T(r_1) \hat{-} \hat{\Upsilon}_T(r_2)$,
(g) $\hat{\Upsilon}_T(q \hat{\bowtie} r) = \hat{\Upsilon}_T(q) \hat{\bowtie} \hat{\Upsilon}_T(r)$, (h) $\hat{\Pi}_S(r_1 \hat{\cup} r_2) = \hat{\Pi}_S(r_1) \hat{\cup} \hat{\Pi}_S(r_2)$.

Proof. Straightforward, by expanding both sides of each identity using the definitions of the operators.

Theorem 4.2 *The proposed temporal relational algebra is closed.*

Proof. We must show that all of the basic operations in this algebra result in a relation as defined in the algebra. A relation must satisfy three criteria: (1) the values must come from an appropriate domain, (2) no two tuples in a relation are value-equivalent, and (3) it must be a finite collection of tuples.

It is easy to see that the first criterion is satisfied for all attributes except timestamps. For timestamp attributes, the domain $\mathscr{T}$ has been constructed in such a fashion that the timestamps are closed under usual set theoretic operations. Also by definition of a tuple, it cannot contain an empty timestamp. For example, $\{\langle 3025, [3,5)\rangle\} \hat{-} \{\langle 3025, [3,5)\rangle\}$ would evaluate to an empty relation ("null set" of tuples) in our algebra, and not to $\{\langle 3025, \emptyset\rangle\}$.

The fact that no two value-equivalent tuples are produced when the basic operators are used is explicitly ensured in the definitions of the operators. If there is a possibility of generation of value-equivalent tuples, then those tuples would be automatically coalesced in this algebra.

One can prove that the third criterion is satisfied by constructing the resulting relation from the operands. Let us consider the union operation. Assume that r_1 and r_2 are relations on the same scheme. For every tuple $x \in r_1$, there are two possibilities. Either there is only one value-equivalent tuple $y \in r_2$, in which case $(x \oplus y) \in (r_1 \hat{\cup} r_2)$; or there is no such tuple in r_2, in which case $x \in (r_1 \hat{\cup} r_2)$. A similar observation is also true about every tuple in r_2. It is then clear that the total number of tuples in $(r_1 \hat{\cup} r_2)$ is no more than $(|r_1| + |r_2|)$, where $|r|$ denotes the number of tuples in a relation r. It can be shown in an analogous manner that (a) the number of tuples in $(r_1 \hat{-} r_2)$ cannot exceed $|r_1|$, (b) the number of tuples in $(r_1 \hat{\bowtie} r_2)$ cannot exceed $|r_1| \times |r_2|$, (c) the number of tuples in $\hat{\Pi}_S(r)$ cannot exceed $|r|$, and (d) the number of tuples in $\hat{\sigma}_P(r)$ cannot exceed $|r|$.

$$\begin{tikzcd}
r \arrow[r, "\text{temporal transform } \mathrm{TT}_\tau"] \arrow[d, "\text{snapshot operator } op"'] & r' = \mathrm{TT}_\tau(r) \arrow[d, "\text{analogous temporal operator } \hat{op}"] \\
op(r) \arrow[r, "\text{temporal transform } \mathrm{TT}_\tau"'] & \mathrm{TT}_\tau(op(r)) = \hat{op}(r')
\end{tikzcd}$$

Fig. 1. Outline of an equivalence proof (McKenzie and Snodgrass 1991b)

4.1 Correspondence with the snapshot algebra

This algebra is a consistent extension of the conventional snapshot algebra, and reduces to the snapshot algebra when time is not modeled. Before these properties can be proven, we first need to define two operators. These operators are not part of the algebra, but allow us to transform static relations to temporal relations and *vice versa*. For these definitions and the related discussion, we will use prime ($'$) for temporal relations exclusively.

Definition 4.1 (Temporal transform) Let R and R' be relation schemes satisfying $TS \notin R$ and $R' = R \cup \{TS\}$. Let r be any relation on scheme R. The **temporal transform** of r over $\tau \in \mathscr{T}$ (written $\mathrm{TT}_\tau(r)$) is a relation $r' = \{x | x(R) \in r, x(TS) = \tau\}$ on R'.

Definition 4.2 (Snapshot) Let R and R' be as above. Let r' be any relation on scheme R'. The **snapshot** of r' at time point t (written $\mathrm{SN}_t(r')$) is a relation $r = \{x(R) | x \in r', t \in x(TS)\}$ on R.

Theorem 4.3 *The proposed temporal relational algebra is a consistent extension of the snapshot algebra.*

Proof. A temporal algebra is said to be a consistent extension of the snapshot algebra if any relation or algebraic expression that can be represented in the snapshot algebra has a counterpart in the temporal algebra (McKenzie and Snodgrass 1991b). In other words, the algebra should be at least as powerful as the snapshot algebra. Figure 1 gives an outline of the equivalence proof for a unary operator.

Let Q and R be two relation schemes such that $TS \notin Q$ and $TS \notin R$, and let $q(Q)$, $r(R)$, $r_1(R)$ and $r_2(R)$ be

$$\begin{tikzcd}
r' \arrow[r, "\text{snapshot}\ \mathrm{SN}_t"] \arrow[d, "\text{temporal operator}\ \hat{op}"'] & r = \mathrm{SN}_t(r') \arrow[d, "\text{analogous snapshot operator}\ op"] \\
\hat{op}(r') \arrow[r, "\text{snapshot}\ \mathrm{SN}_t"'] & \mathrm{SN}_t(\hat{op}(r')) = op(r)
\end{tikzcd}$$

Fig. 2. Outline of a reduction proof (McKenzie and Snodgrass 1991b)

relations on these schemes. Also let $Q' = Q \cup \{TS\}$ and $R' = R \cup \{TS\}$. In order to establish this theorem, we must prove just the following identities for any $\tau \in \mathscr{T}$:

(a) $\mathrm{TT}_\tau(r_1)\hat{\cup}\mathrm{TT}_\tau(r_2) = \mathrm{TT}_\tau(r_1 \cup r_2)$,
(b) $\mathrm{TT}_\tau(r_1)\hat{-}\mathrm{TT}_\tau(r_2) = \mathrm{TT}_\tau(r_1 - r_2)$,
(c) $\mathrm{TT}_\tau(q)\hat{\bowtie}\mathrm{TT}_\tau(r) = \mathrm{TT}_\tau(q \bowtie r)$,
(d) $\hat{\sigma}_P(\mathrm{TT}_\tau(r)) = \mathrm{TT}_\tau(\sigma_P(r))$,
(e) $\hat{\Pi}_{S'}(\mathrm{TT}_\tau(r)) = \mathrm{TT}_\tau(\Pi_S(r))$, where $S \subset R$ and $S' = S \cup \{TS\}$

We will prove only the first identity; the others can be proven similarly.

If a tuple $x' \in_t \mathrm{TT}_\tau(r_1)\hat{\cup}\mathrm{TT}_\tau(r_2)$, then there are just two possibilities: (1) $x' \in_t \mathrm{TT}_\tau(r_1)$, or (2) $x' \in_t \mathrm{TT}_\tau(r_2)$. The third possibility of existence of value-equivalent tuples is not relevant, since all of the tuples have identical timestamps. This implies that $x'(R)$ is in r_1 or in r_2, or in both, i.e., $x'(R) \in (r_1 \cup r_2)$. Then $x' \in \mathrm{TT}_\tau(r_1 \cup r_2)$.

Alternatively, if a tuple x' is temporally in $\mathrm{TT}_\tau(r_1 \cup r_2)$, then $x'(R) \in (r_1 \cup r_2)$, which implies that either $x' \in_t \mathrm{TT}_\tau(r_1)$, or $x' \in_t \mathrm{TT}_\tau(r_2)$. Clearly, $x' \in_t (\mathrm{TT}_\tau(r_1) \hat{\cup}\mathrm{TT}_\tau(r_2))$.

Theorem 4.4 *The proposed temporal relational algebra reduces to the snapshot algebra.*

Proof. A temporal relational algebra is said to reduce to the snapshot algebra if the semantics of the algebra is consistent with that of the snapshot algebra (McKenzie and Snodgrass 1991b). The reduction proof for any unary operator is outlined in Figure 2.

Let Q' and R' be two relation schemes such that $TS \in Q'$ and $TS \in R'$, and let $q'(Q')$, $r'(R')$, $r_1'(R')$ and $r_2'(R')$ be relations on these schemes. Also let $Q = Q' - \{TS\}$ and $R = R' - \{TS\}$. We must prove the following identities for any time point t:

(a) $\mathrm{SN}_t(r_1') \cup \mathrm{SN}_t(r_2') = \mathrm{SN}_t(r_1'\hat{\cup}r_2')$,
(b) $\mathrm{SN}_t(r_1') - \mathrm{SN}_t(r_2') = \mathrm{SN}_t(r_1'\hat{-}r_2')$,
(c) $\mathrm{SN}_t(q') \bowtie \mathrm{SN}_t(r') = \mathrm{SN}_t(q'\hat{\bowtie}r')$,
(d) $\sigma_P(\mathrm{SN}_t(r')) = \mathrm{SN}_t(\hat{\sigma}_P(r'))$,
(e) $\Pi_S(\mathrm{SN}_t(r')) = \mathrm{SN}_t(\hat{\Pi}_{S'}(r'))$, where $S \subset R$, and $S' = S \cup \{TS\}$

The first identity is proven below. The rest follow in an analogous fashion.

If a tuple $x \in \mathrm{SN}_t(r_1') \cup \mathrm{SN}_t(r_2')$, then either $x \in \mathrm{SN}_t(r_1')$ or $x \in \mathrm{SN}_t(r_2')$. If $x \in \mathrm{SN}_t(r_1')$, then there exists $x' \in r_1'$ such that $x = x'(R)$ and $t \in x'(TS)$. Similarly, if $x \in \mathrm{SN}_t(r_2')$, then there exists $x' \in r_2'$ such that $x = x'(R)$ and $t \in x'(TS)$. In either case, $x' \in (r_1'\hat{\cup}r_2')$, $x = x'(R)$ and $t \in x'(TS)$. Clearly, $x \in \mathrm{SN}_t(r_1'\hat{\cup}r_2')$.

Now assume that a tuple x is in $\mathrm{SN}_t(r_1'\hat{\cup}r_2')$. Then there exists $x' \in (r_1'\hat{\cup}r_2')$, with $x = x'(R)$ and $t \in x'(TS)$. This means that x' is a partial temporal member in at least one of r_1' and r_2'. Clearly, x must be in at least one of $\mathrm{SN}_t(r_1')$ and $\mathrm{SN}_t(r_2')$, which implies that $x \in (\mathrm{SN}_t(r_1') \cup \mathrm{SN}_t(r_2'))$.

4.2 Correspondence with temporal calculus and TQuel

In this section, we show that the proposed algebra is equivalent to a temporal calculus in its expressive power. This, in a sense, would ensure the much desired completeness of the algebra.[5] We will use TQuel (Snodgrass 1987) (which is based on temporal calculus) for this purpose. The reasons for choosing TQuel over others proposed in the literature are simple. First, TQuel employs tuple time-stamping as does our algebra. Second, TQuel semantics is formally defined. Third, TQuel uses both valid and transaction time. Finally, TQuel is *temporally complete*.

The TQuel *retrieve* statement has the following syntax (Snodgrass 1987):

```
range of $x_1$ is $r_1$
:
:
range of $x_k$ is $r_k$
retrieve ($x_{i_1}.D_{j_1}, \ldots, x_{i_q}.D_{j_q}$)
    valid from $\nu$ to $\chi$
    where $\psi$
    when $\tau$
    as of $\alpha$ through $\beta$
```

The corresponding tuple calculus statement has the following form:

$$\begin{aligned}
&\{y^{(q+4)} | (\exists x_1)\ldots(\exists x_k)(r_1(x_1) \wedge \ldots \wedge r_k(x_k)\\
&\wedge y[1] = x_{i_1}[j_1] \wedge \ldots \wedge y[q] = x_{i_q}[j_q]\\
&\wedge y[q+1] = \Phi_\nu \wedge y[q+2] = \Phi_\chi \wedge \mathrm{Before}(y[q+1], y[q+2])\\
&\wedge y[q+3] = \text{current transaction id} \wedge y[q+4] = \infty\\
&\wedge \psi' \quad \wedge \Gamma_\tau\\
&\wedge (\forall l)(1 \le l \le k.(\mathrm{Before}(\Phi_\alpha, x_l[\mathrm{stop}]) \wedge \mathrm{Before}(x_l[\mathrm{stop}], \Phi_\beta)))\\
&)\},
\end{aligned}$$

where ψ' and Γ_τ are obtained from ψ and τ by replacing each occurrence of an attribute name by the value of that attribute for a tuple.

We, however, necessarily wish to deviate from Snodgrass's philosophy of updating the transaction timestamp (in the resulting relation) to the current transaction id (or, time). Recall that the transaction time of a tuple is the time when the information contained in the tuple was recorded in the database. Thus, the transaction time should only be inserted or changed when a new tuple is inserted or an old tuple is updated. It must be noted that no new information is recorded in the database during the execution of a retrieval query.

[5] McKenzie and Snodgrass (1991a) define a relational algebra to be *complete* if it is at least as expressive as the snapshot algebra. In that sense, this algebra is complete (see Theorem 4.3). However, we adopt a different criterion for the completeness in a temporal relational model. We feel that the completeness of a temporal relational algebra must require reducibility of every expression in a temporal relational calculus to an equivalent expression in that algebra

Table 7. Relation SAL

EMP#	TS	Salary
3025	$[1,3) \cup [5,7)$	18K
3025	$[3,5) \cup [7,9)$	20K
6637	$[3,7)$	17K
6637	$[7,12)$	19K

Then, it is clear that the transaction timestamp of any tuple in the resulting relation should not be $[t_{\text{NOW}}, \infty)$. Rather, it should be calculated from the tuples of the participating relations, much in the same way valid timestamp for the tuple is calculated. The modified tuple calculus statement would then assume the form:

$$
\begin{aligned}
&\{y^{(q+4)} | (\exists x_1) \ldots (\exists x_k)(r_1(x_1) \wedge \ldots \wedge r_k(x_k)\\
&\wedge y[1] = x_{i_1}[j_1] \wedge \ldots \wedge y[q] = x_{i_q}[j_q]\\
&\wedge y[q+1] = \Phi_\nu \wedge y[q+2] = \Phi_\chi \wedge \text{Before}(y[q+1], y[q+2])\\
&\wedge y[q+3] = \Phi_\alpha \wedge y[q+4] = \Phi_\beta \wedge \text{Before}(y[q+3], y[q+4])\\
&\wedge \psi' \quad \wedge \Gamma_\tau\\
&\wedge (\forall l)(1 \le l \le k.(\text{Before}(\Phi_\alpha, x_l[\text{stop}]) \wedge \text{Before}(x_l[\text{stop}], \Phi_\beta)))\\
&)\}
\end{aligned}
$$

Let us call this the **“modified TQuel semantics”**. It must also be noted that TQuel coalesces value-equivalent tuples only if they are adjacent or overlapping. Thus TQuel allows other value-equivalent tuples to be present. It will, however, be assumed that a resulting relation contains no value-equivalent tuples; that is, all value-equivalent tuples are coalesced.

Theorem 4.5 *Every TQuel* **retrieve** *statement, with the* **modified TQuel semantics** *can be expressed in the proposed temporal relational algebra.*

Proof. Consider a generalized retrieve statement of TQuel with the modified TQuel semantics as given above. It can easily be verified that this statement is equivalent to the following expression in our algebra:

$$\hat{\Pi}_{\{j_1,\ldots,j_q,TS\}} \left(\hat{\Upsilon}_{[\nu,\chi)\times[\alpha,\beta)} \left(\hat{\sigma}_{(\psi\wedge\tau)}(r'_1 \hat{\times} r'_2 \hat{\times} \ldots \hat{\times} r'_k)\right)\right)$$

where r'_i is basically r_i, $1 \le i \le k$, with some attributes renamed such that the Cartesian product operation is defined on these relations. The Cartesian product ensures that all the relevant relations take part in the expression. The selection operation ensures that the predicates ψ and τ are satisfied for all tuples in the resulting relation. The temporal projection allows us to select only those tuples that are defined during $[\nu,\chi) \times [\alpha,\beta)$. Finally, the projection operation allows one to display only the necessary attributes.

Clifford et al. (1994) show that TQuel is complete with respect to their calculus TC for ungrouped temporal relations. This implies that our algebra is also equivalent to TC and hence is TU complete. To illustrate the superior expressive power of this algebra, let us formulate a query that Clifford et al. (1994) show most algebras fail to express: “*Find the employees who have at some time received a salary cut.*” We consider the relation SAL shown in Table 7.

This query can be answered by using the θ-join operation between two instances of the same relation. However, before applying this operation, we need to rename the attribute names in the second instance. Let the renamed instance be SAL$'$ with attribute A being renamed to A'. Note that the timestamp TS in SAL is renamed to TS' in SAL$'$, and is no longer a timestamp. The answer can be found by evaluating the following relational expression:

$$\hat{\Pi}_{\text{EMP\#}} \left(\text{SAL} \,\hat{\bowtie}_P\, \text{SAL}'\right),$$

where

$$P = (\text{EMP\#} = \text{EMP\#}') \wedge (\text{salary} > \text{salary}') \wedge (TS \lhd TS').$$

If other attributes (such as name, department and manager) of these employees are needed, that can be easily found by joining the above expression with the relations which contain that information.

4.3 Evaluation of the proposed algebra

Recently Snodgrass et al. (1994) describe a temporal extension to SQL-92. They provide a list of desirable features for the data model and the language. Our model contains most of these features. First, it employs tuple time-stamping. As a result, all the tuples are automatically homogeneous. Second, our valid time support includes support for both the past and the future. Third, since we assume the time line is continuous, our timestamp values are not limited in range or precision. Fourth, our algebra is a consistent extension of the relational algebra. Fifth, our operations do not accord any explicit attribute special semantics. Sixth, in our model temporal support is optional; it is possible to have static relations as a special case. Seventh, we define temporal extensions of all common aggregate functions. Finally, our relations are implementable in terms of first normal form relations. The use of multidimensional elementary subsets as timestamps should not be considered as a violation of first normal form. In this algebra, each elementary subset is treated as a single value for a time attribute, not as a set of values (or as repeating groups). Valid operations on these values are also discussed so that the domain $\mathscr{T}$ can be supported in a temporal database management system.

Snodgrass et al. (1994) conclude that TSQL2 must have an efficiently implementable algebra. Since our algebra satisfies most of the desirable features, we feel that it is the most suitable for implementation of TSQL2.

Besides, McKenzie and Snodgrass (1991b) provide a set of 26 criteria for evaluating temporal algebras. The algebra proposed in this research is evaluated on these criteria.[6] It is also compared to five other existing algebras: (1) the time relational model by Ben-Zvi (1982), (2) the homogeneous model by Gadia (1988), (3) the temporal relational model by Navathe and Ahmed (1989), (4) the historical relational algebra by Sarda (1990) and (5) the historical relational algebra by McKenzie and Snodgrass (1991a). This evaluation and comparison is summarized in Table 8. From this table, it can be seen that the proposed algebra satisfies the maximum number of criteria. We believe that the primary strength of this algebra lies in its simplicity and its consistency with the snapshot algebra.

[6] McKenzie and Snodgrass (1991b) clearly show that, out of these 26 criteria, seven criteria are conflicting. In other words, no algebra can satisfy all 26 criteria

Table 8. Evaluation of the proposed complete temporal relational algebra based on the criteria of McKenzie and Snodgrass (1991b)

	Criteria	Ben	Gad	NvA	Sar	McS	CTR
1.	All attributes in a tuple are defined for same interval(s)	Y	Y	Y	Y	N	Y
2.	Consistent extension of the snapshot algebra	Y	Y	?	?	Y	Y
3.	Data periodicity is supported	N	N	N	N	N	N
4.	Each collection of legal attribute values is a legal tuple	N	N	N	Y	N	Y
5.	Each set of legal tuples is a legal relation	Y	Y	N	Y	N	N
6.	Formal semantics are well defined	P	Y	P	P	Y	Y
7.	Has the expressive power of a temporal calculus	P	Y	P	P	Y	Y
8.	Includes aggregates	Y	P	N	N	Y	Y
9.	Incremental semantics defined	N	N	N	N	Y	N
10.	Intersection, Θ-join, natural join, and quotient are defined	P	P	P	N	Y	Y
11.	Is, in fact, an algebra	Y	Y	P	?	Y	Y
12.	Model doesn't require null attribute values	Y	Y	Y	Y	Y	Y
13.	Multidimensional timestamps are supported	N	N	N	N	N	Y
14.	Reduces to the snapshot algebra	Y	Y	Y	Y	P	Y
15.	Restricts relations to first normal form	Y	N	Y	N	N	P
16.	Supports a 3D view of historical states and operations	N	N	N	N	Y	N
17.	Supports basic algebraic equivalence	Y	Y	?	?	P	Y
18.	Supports relations of all four classes	P	P	P	P	Y	Y
19.	Supports rollback operations	P	N	N	N	Y	Y
20.	Supports multiple stored schemas	N	N	N	N	Y	N
21.	Supports static attributes	N	N	Y	N	Y	Y
22.	Treats valid time and transaction time orthogonally	Y	?	?	?	P	Y
23.	Tuples are timestamped	Y	N	Y	Y	N	Y
24.	Unique representation for each temporal relation	N	N	Y	N	Y	Y
25.	Unisorted (not multisorted)	N	N	N	N	N	Y
26.	Update semantics are specified	P	N	N	N	Y	Y

The ratings for all the models except the one presented in this paper have been taken from (McKenzie and Snodgrass 1991b).
Y, criterion satisfied; N, criterion not satisfied; P, criterion partially satisfied; ?, unspecified in report; Ben, the time relational model by Ben-Zvi (1982); Gad, the homogeneous model by Gadia (1988); NvA, the temporal relational model by Navathe and Ahmed (1989); Sar, the relational algebra for a historical data model by Sarda (1990); McS, the relational algebra by McKenzie and Snodgrass (1991a); CTR, the complete temporal relational algebra proposed in this paper

5 Conclusion

Although temporal databases have been an active area of research in recent years, representations of temporal data in the relational model have been varied. The complete temporal relational algebra proposed in this paper resolves several open temporal relational issues, yet it is not significantly different from the original relational algebra.

This algebra employs tuple time-stamping with multidimensional timestamps. The basic relational operations have been redefined as consistent extensions of the snapshot operations in a manner that preserves the basic algebraic equivalences of the snapshot algebra. A new operation, called *temporal projection*, is introduced. The complete update semantics are formally specified and aggregate functions are defined. The algebra is closed and reduces to the snapshot algebra. It is also shown to be at least as expressive as the calculus-based temporal query language TQuel (Snodgrass 1987). In order to assess the algebra, it is evaluated using a set of 26 criteria proposed by McKenzie and Snodgrass (1991b), and compared to several existing temporal relational algebras. The proposed algebra satisfies the maximum number of criteria.

We believe that a temporal extension of the relational model should attach the timestamps at the tuple level. This is because the view is simpler and much closer to the original relational model. Although a number of research efforts have tried this approach, they all suffer from several limitations. A major contribution of this work is a concise and formal definition of the relational structure and relational operations.

One of the immediate future directions of this research is to implement this algebra in a temporal database management system; this is currently being pursued. Once this is done, we could build a more user-friendly interface such as TSQL2. We also plan to conduct a benchmark study that will evaluate the performance of a temporal database management system based on our algebra. One of the implicit assumptions of the proposed algebra is that a relation scheme does not change over time. This assumption, however, is too restrictive to model the real world accurately. McKenzie and Snodgrass (1990) and Roddick (1992) examine evolution of relation schemes. Future research will explore how the evolution of a relation scheme can be incorporated into our algebra.

Acknowledgements. This research was partially supported by the William E. Simon Graduate School of Business Administration at the University of Rochester. We wish to thank our anonymous reviewers for their insightful comments on a previous version of this manuscript.

References

1. Ben-Zvi J (1982) The time relational model (PhD dissertation) UCLA
2. Cheng TS, Gadia SK (1993) An object-oriented model for temporal databases. In: Proceedings of the International Workshop on an Infrastructure for Temporal Databases, Arlington, Tex, June

3. Clifford J, Croker A (1987) The historical relational data model (HRDM) and algebra based on lifespans. Proceedings of the Third International Conference on Data Engineering, February, Redwood City, Calif, pp 528–537
4. Clifford J, Croker A (1993) The historical relational data model (HRDM) revisited. In: Tansel AU, Clifford J, Gadia S, Jajodia S, Segev A, Snodgrass R (eds.), Temporal databases: theory, design, and implementation, Benjamin/Cummings, Redwood City, Calif
5. Clifford J, Croker A, Tuzhilin A (1994) On completeness of historical relational query languages. ACM Trans Database Sys **19**:64–116
6. Clifford J, Tansel AU (1985) On an algebra for historical relational databases: two views. ACM SIGMOD Rec **14**(4):247–265
7. Codd EF (1970) A relational model of data for large shared data banks. Commun of the ACM **13**:377–387
8. Codd EF The (1990) Relational model for database management: version 2. Addison-Wesley, Readings, Mass
9. Dayal U, Wuu GTJ (1992) A uniform approach to processing temporal queries. In: Proceedings of the 18th VLDB Conference, Vancouver, British Columbia, Canada
10. Gadia SK (1986) Toward a multihomogeneous model for a temporal database. In: Proceedings of the Second International Conference on Data Engineering, February, pp 390–397
11. Gadia SK (1988) A homogeneous relational model and query languages. ACM Trans Database Sys **13**:418–448
12. Gadia SK, Yeung CS (1988) A generalized model for a relational temporal database. ACM SIGMOD Rec **17**(3):251–259
13. Jones S, Mason P, Stamper R (1979) LEGOL 2.0: a relational specification language for complex rules. Inf Sys, **4**:293–305
14. Kline N (1993) An update of the temporal database bibliography. ACM SIGMOD Rec **22**(4):66–80
15. Klug A (1982) Equivalence of relational algebra and relational calculus languages having aggregate functions. J ACM **29**:699–717
16. Lorentzos NA, Johnson RG (1988) Extending relational algebra to manipulate temporal data. Inf Sys **13**:289–296
17. Maier D (1983) The theory of relational databases. Computer Science Press, Rockville, Md
18. McKenzie LE Jr, Snodgrass RT (1990) Schema evolution and the relational algebra. Inf Sys **15**:207–232
19. McKenzie LE Jr, Snodgrass RT (1991a) Supporting valid time in historical relational algebra: proofs and extensions. Tech Rep TR 91-15, Deptartment of Computer Science, University of Arizona, USA
20. McKenzie LE Jr, Snodgrass RT (1991b) Evaluation of relational algebras incorporating the time dimension in databases. ACM Comp Surv **23**:501–543
21. Navathe SB, Ahmed R (1989) A temporal relational model and a query language. Inf Sci **49**:147–175
22. Navathe SB, Ahmed R (1993) Temporal extensions to the relational model and SQL. In: Tansel AU, Clifford J, Gadia S, Jajodia S, Segev A, Snodgrass R (eds.), Temporal databases: theory, design, and implementation. Benjamin Cummings, Redwood City, Calif
23. Roddick JF (1992) SQL/SE – a query language extension for databases supporting schema evolution. ACM SIGMOD Rec **21**(3):10–16
24. Rose E, Segev A (1991) TOODM – a temporal object-oriented data model with temporal constraints. In: Proceedings of the Tenth International Conference on Entity-Relationship Approach, San Mateo, Calif, October, pp 205–229
25. Rose E, Segev A (1992) TO-algebra – a temporal object-oriented algebra. Tech Rep LBL-32013, University of California, Berkeley
26. Roth MA, Korth HF, Silberschatz A (1988) Extended algebra and calculus for nested relational databases. ACM Trans Database Sys **13**:389–417
27. Rudin W (1976) Principles of mathematical analysis. McGraw-Hill, New York
28. Sarda NL (1990) Algebra and query language for a historical data model. Comp J **33**:11–18
29. Sarda NL (1993) HSQL: A historical query language. In: Tansel AU, Clifford J, Gadia S, Jajodia S, Segev A, Snodgrass R (eds.) Temporal databases: theory, design, and implementation, Benjamin Cummings, Redwood City, Calif
30. Segev A, Shoshani A (1988) The representation of a temporal data model in the relational environment. In: Rafanelli M, Klensin JC, Svensson P (eds.) Proceedings of the Fourth International Working Conference on Statistical and Scientific Database Management (SSDBM). Rome, Italy, June
31. Snodgrass RT (1987) The temporal query language TQuel. ACM Trans Database Sys **12**:247–298
32. Snodgrass RT (1990) Temporal databases status and research directions. ACM SIGMOD Rec **19**(4):83–89
33. Snodgrass RT, Ahn I (1985) A taxonomy of time in databases. ACM SIGMOD Rec **14**:236–246
34. Snodgrass RT, Ahn I, Ariav G, Batory D, Clifford J, Dyreson CE, Elmasri R, Grandi F, Jensen CS, Käfer W, Kline N, Kulkarni K, Leung TYC, Lorentzos N, Roddick JF, Segev A, Soo MD, Sripada SM (1994) TSQL2 language specification. ACM SIGMOD Rec **23**(1):65–86
35. Tansel AU (1986) Adding time dimension to relational model and extending relational algebra. Inf Sys **11**:343–355
36. Tansel AU (1987) A statistical interface for historical relational databases. In: Proceedings of the Third International Conference on Data Engineering, February, pp 538–546
37. Tuzhilin A, Clifford J (1990) A temporal relational algebra as a basis for temporal completeness. In: Proceedings of the Conference on Very Large Data Bases, Brisbane, Australia
38. Ullman JD (1988) Principles of database and knowledge-base systems (Volume I: Classical Database Systems). Computer Science Press, Rockville, MD
39. Wuu GTJ, Dayal U (1993) A uniform model for temporal and versioned object-oriented databases. In: Tansel AU, Clifford J, Gadia S, Jajodia S, Segev A, Snodgrass R (eds.) Temporal databases: theory, design, and implementation. Benjamin Cummings, Redwood City, Calif

This article was processed by the author using the LaTeX style file *pljour2* from Springer-Verlag.