# An Evaluation of Historical Algebras

*TR87-020*

*October 1987*

*Edwin McKenzie*
*Richard Snodgrass*

The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

# Abstract

In this paper we survey *historical* algebras, extensions of the conventional relational algebra that support representation of the temporal dimension of real-world phenomena in databases. We identify twenty-one criteria for evaluating historical algebras. These criteria are well-defined, have an objective basis for being evaluated, and are arguably beneficial. We also identify incompatibilities among the criteria. Nine historical algebras are evaluated against the criteria.

# Contents

# List of Figures

# List of Tables

Time is an attribute of all real-world phenomena. Events occur at specific points in time; objects and the relationships among objects exist over time. The ability to model this temporal dimension of the real world is essential to many computer system applications (e.g., econometrics, banking, inventory control, medical records, and airline reservations). Yet, none of the three major data models — relational, network, hierarchical — supports the time-varying aspect of real-world phenomena. Conventional databases can be viewed as *snapshot* databases in that they represent the state of an enterprise at one particular time. As a database changes, out-of-date information, representing past states of the enterprise, is discarded. Although techniques for encoding time-varying information in conventional databases have been developed in many application areas, these techniques are necessarily ad hoc and application-specific.

The need for direct database support for time-varying information has received increasing attention recently. In the last five years, more than 80 articles relating time to information processing have been published [McKenzie 1986]. One area of continuing research interest is development of an *historical data model*, a data model capable of representing the temporal dimension of real-world phenomena. The primary focus has been extending the relational data model to support time-varying information.

Over the past decade, several historical algebras have been proposed. An historical algebra is essential to the formulation of an historical data model because it defines formally the types of objects and the operations on object instances allowed in the data model. The usefulness of an historical data model in representing the time-varying aspect of real-world phenomena depends on the power and expressiveness of its underlying historical algebra. Similarly, the algebra determines a data model's support of calculus-based query languages. Also, implementation issues, such as query optimization and physical storage strategies, can best be addressed in terms of the algebra.

In this paper we examine nine historical algebras. Each is an extension of the conventional relational algebra that supports representation of the temporal dimension of real-world phenomena in an historical data model. In the next section, we first review the conventional relational algebra and then describe briefly the historical algebras that have been proposed, emphasizing the types of objects that each defines and the operations on object instances that each provides. Although several historical algebras have been proposed, current research has not focused on defining criteria for evaluating the relative merit of these historical algebras. Hence, we next identify 21 criteria for evaluating historical algebras. These criteria are well-defined, have an objective basis for being evaluated, and are arguably beneficial. Finally, we evaluate the historical algebras against the criteria.

## 1 Historical Algebras

In Codd's relational algebra [Codd 1970], the only type of object is the set-theoretic *relation*. Assume that we are given a set of names $\mathcal{A} = \{A_1, \ldots, A_n\}$, where each $A_i$, $1 \leq i \leq n$, is called an *attribute name*, or simply an *attribute*. Also, assume that there is an arbitrary, non-empty, finite or denumerable set, $\mathcal{D}_i$, $1 \leq i \leq n$, called a *domain* corresponding to attribute $A_i$ [Maier 1983]. Then, a relation on these $n$ domains is a set of $n$*-tuples*, where each tuple is itself a set of ordered pairs $(A_i,\ D_i)$, $D_i \in \mathcal{D}_i$, $1 \leq i \leq n$ [Date 1986]. Hence, each element of a tuple maps an attribute name onto a value in its associated domain. The set of attributes $\mathcal{A}$ for a relation is called the *relation scheme*. Because relations are sets, tuples with duplicate attribute values are not permitted. Also, if the domains are sets of atomic values, then a relation is said to be in *first-normal-form*. Relations are most often displayed as tables in which the rows correspond to tuples and the columns correspond to attributes.

*EXAMPLE.* Assume that we are given the relation scheme $Student = \{Name, Course\}$ and corresponding domains of given names and college courses. Then R is a relation on the scheme *Student*.

R =

| *Name* | *Course* |
|---|---|
| Phil | English |
| Norman | English |
| Norman | Calculus |

□

There are five basic operations in the relational algebra: *union*, *set difference*, *cartesian product*, *selection*, and *projection* [Ullman 1982]. The union of two relations is the relation containing tuples that are in either of the two input relations. The set difference of two relations is the relation containing tuples that are in the first input relation but not in the second input relation. The cartesian product of a relation of $n$-tuples and a relation of $m$-tuples is the relation containing $n + m$-tuples that have $n$ elements that form a tuple in the first input relation and $m$ elements that form a tuple in the second input relation. We assume, without loss of generality, that the relation schemes of the input relations are disjoint [Maier 1983]. Selection and projection are unary operations. Selection maps an input relation onto an output relation containing only those tuples in the input relation that satisfy a specified predicate. Hence, selection reduces a relation "horizontally" by removing tuples. Projection maps each tuple in its input relation onto a tuple in its output relation having only a specified subset of the attributes of the input tuple. Hence, projection reduces a relation "vertically" by removing attributes. Other operations (e.g., intersection, divide, join) can be defined in terms of these five basic operations.

Before discussing the specific extensions to the relational algebra, we must consider two aspects of time that apply to all such extensions. The first aspect concerns the kinds of time the algebras support. There are three orthogonal kinds of time that a data model may support: valid time, transaction time, and user-defined time [Snodgrass & Ahn 1985, Snodgrass & Ahn 1986]. *Valid time* concerns modeling time-varying reality. The valid time of, say, an event is the clock time at which the event occurred in the real world, independent of the recording of that event in some database. Other terms found in the literature that have the same meaning as valid time include intrinsic time [Bubenko 1977], effective time [Ben-Zvi 1982], and logical time [Dadam et al. 1984, Lum et al. 1984]. *Transaction time*, on the other hand, concerns the storage of information in the database. The transaction time of an event is the transaction number (an integer) of the transaction that stored the information about the event in the database. Other terms found in the literature that have the same meaning as transaction time include extrinsic time [Bubenko 1977], registration time [Ben-Zvi 1982], and physical time [Dadam et al. 1984, Lum et al. 1984]. *User-defined time* is an uninterpreted domain for which the data model supports the operations of input, output, and perhaps comparison. As its name implies, the semantics of user-defined time is provided by the user or application program.

The relational algebra already supports user-defined time in that user-defined time is simply another domain, such as integer or character string. The relational algebra, however, supports neither valid time nor transaction time. A relation is time-varying in that it changes over time due to the insertion, deletion, and modification of tuples. Yet, with each change, out-of-date information is discarded. No record of the evolution of either the relation or the enterprise that it models is maintained. Only one version, the current version, of the relation exists and its contents represent the state of the enterprise being modeled at a single time. Hence, we refer to the relational algebra hereafter as the snapshot algebra, as it captures only a single snapshot in time of both a relation and the enterprise that the relation models.

The second aspect that we must consider is the conceptual model of time employed. There are two basic conceptual time models: the continuous model, in which time is viewed as being isomorphic to the real numbers, and the discrete model, in which time is viewed as being isomorphic to the natural numbers (or a discrete subset of the real numbers) [Clifford & Tansel 1985]. In the continuous model, each real number corresponds to a "point" in time whereas in the discrete model, each natural number corresponds to a non-decomposable unit of time having an arbitrary duration. In addition to "point," "instant," [Gadia 1986] "moment,"[Allen & Hayes 1985] "time quantum," [Anderson 1982] and "time unit" [Navathe & Ahmed 1986, Tansel 1986] are some of the terms used in the literature to describe a non-decomposable unit of time in the discrete model. To avoid confusion between a point in the continuous model and a non-decomposable unit of time in the discrete model, we refer to a non-decomposable unit of time in the discrete model as a *chronon* [Ariav 1986] and define an *interval* to be a set of consecutive chronons. Although the duration of each chronon in a set of times need not be the same, the duration of a chronon is usually fixed by the granularity of the measure of time being used (e.g., day, week, hour, second). A chronon is typically represented as an integer, corresponding to a single granularity, but may also be represented as a sequence of integers, corresponding to a nested granularity. For example, if we assume a granularity of a day relative to January 1, 1980, then the integer 1,901 represents March 15, 1985. If, however we assume a nested granularity of $\langle$ year, month, day $\rangle$, then the sequence $\langle$ 6, 3, 15 $\rangle$ represents March 15, 1985. Although the two time models represent time differently, they share one important property; they both require that time be ordered linearly. Hence, for two non-equal times, $t_1$ and $t_2$, either $t_1$ is "before" $t_2$ or $t_2$ is "before" $t_1$ [Anderson 1982, Clifford & Tansel 1985].

Although time itself is continuous, most proposals for adding a temporal dimension to the relational data model are based on the discrete time model. Several practical arguments are given in the literature for this preference for the discrete model over the continuous model. First, measures of time are inherently imprecise [Anderson 1982, Clifford & Tansel 1985]. Clocking instruments invariably report the occurrence of events in terms of chronons, not time "points." Hence, events, even so-called "instantaneous" events, can at best be measured as having occurred during a chronon. Secondly, most natural language references to time are compatible with the discrete time model. For example, when we say that an event occurred at 4:30 p.m., we usually don't mean that the event occurred at the "point" in time associated with 4:30 p.m., but at some time in the chronon (minute) associated with 4:30 p.m. [Anderson 1982]. Thirdly, the concepts of chronon and interval allow us to model naturally events that are not instantaneous, but have duration [Anderson 1982]. Finally, any implementation of a data model with a temporal dimension will of necessity have to have some discrete encoding for time [Snodgrass 1987]. All the historical algebras surveyed in this paper are compatible with the discrete time model.

In the remainder of this section we review briefly nine historical algebras, all extensions of the snapshot algebra that support valid time. These algebras differ in the types of objects they define and in the kinds of operations they provide. We consider only extensions of the snapshot algebra that support valid time; we do not consider extensions of the snapshot algebra that support transaction time (several extensions supporting transaction time have been proposed elsewhere [Ben-Zvi 1982, McKenzie & Snodgrass 1987A, McKenzie & Snodgrass 1987B]). Because valid time and transaction time are orthogonal, support for each type of time can be studied in isolation.

Three basic design decisions characterize the types of objects that each algebra defines.

- Is valid time associated with tuples (usually as additional implicit attributes) or attributes?
- How is valid time represented? Time-stamps, which represent valid time, may be either chronons, intervals, or sets of intervals, where a set of intervals is defined as a set of chronons, not all of which are consecutive.
- Are attributes atomic-valued? If attributes are not atomic-valued, then the first-normal-form property of the snapshot algebra cannot be satisfied.

Also, two basic design decisions characterize the different kinds of operations that each algebra provides.

- Does the algebra retain the set-theoretic semantics of the five basic relational operators and introduce new operators to deal with the temporal dimension of the real-world phenomena being modeled or does the algebra extend the semantics of the relational operators to account for the temporal dimension directly? If the semantics of the relational operators is extended to handle time, how do these operators compute the valid times of resulting tuples?
- How does the algebra handle *temporal selection* (i.e., tuple selection based on valid times) and *temporal projection* (i.e., computation of a new valid time for a tuple from its current valid time, if tuples are time-stamped, or computation of new valid times for a tuple's attributes from their current valid times, if attributes are time-stamped).

Each algebra is characterized by the choices made for these five key design decisions.

LEGOL 2.0 [Jones et al. 1979] is a language based on the relational model designed to be used in database applications where modeling the temporal dimension of real-world phenomena is important. Objects in the LEGOL 2.0 data model are relations of tuples as in the relational data model, with one distinction. Tuples in LEGOL 2.0 are assigned two implicit time attributes, *Start* and *Stop*. The values of these two attributes are the chronons corresponding to the end-points of the interval of existence (i.e., valid time) of the real-world object or relationship represented by a tuple.

*EXAMPLE.* R is an historical relation in LEGOL 2.0 on the relation scheme *Student* = {*Name*, *Course*}. For this and all later examples, assume that the granularity of time is a semester relative to the Fall semester 1980. Hence, 1 represents the chronon Fall semester 1980, 2 represents the chronon Spring semester 1981, etc. Later examples in this section will show the semantically equivalent representation of R in the other algebras.

R =

| *Name* | *Course* | *Start* | *Stop* |
|---|---|---|---|
| Phil | English | 1 | 1 |
| Phil | English | 3 | 4 |
| Norman | English | 1 | 2 |
| Norman | Calculus | 5 | 6 |

Note that two tuples are needed to record Phil's enrollment in English, as his enrollment was not continuous. □

Operations in LEGOL 2.0 are not defined formally, although the more important operations are described using examples. LEGOL 2.0 retains the standard set-theoretic operations and introduces several time-related operations to handle the temporal dimension of data. The new time-related operations are time intersection, one-sided time intersection, time union, time difference, and time-set membership. Time intersection acts as a temporal join, where the valid time of each output tuple is computed using *intersection semantics* (i.e., the valid time of each output tuple is the intersection of the valid times of two overlapping input tuples). Although the semantics of the other time-related operations is left unspecified, these operators appear to support a limited form of temporal selection as well as a temporal join using *union semantics* (i.e., the valid time of each output tuple is the union of the valid times of two overlapping input tuples).

The Time Relational Model [Ben-Zvi 1982] supports both valid time and transaction time. Two types of objects are defined: snapshot relations, as defined in the snapshot algebra, and temporal relations. Temporal relations are relations of tuples, each tuple having five implicit time attributes. *Effective-Time-Start* and *Effective-Time-Stop* are the end-points of the interval of existence of the real-world phenomena being modeled, *Registration-Time-Start* and *Registration-Time-Stop* are the end-points of the interval when the tuple is logically a tuple in the relation, and *Deletion-Time* records the time when erroneously entered tuples are logically deleted.

*EXAMPLE.* R is a temporal relation in the Time Relational Model on the relation scheme *Student* = {*Name*, *Course*}. For completeness, we assume that the tuples were inserted into the relation by the transaction corresponding to transaction number 423 and have yet to be deleted.

R =

| *Name* | *Course* | *Effective Time-Start* | *Effective Time-Stop* | *Registration Time-Start* | *Registration Time-Stop* | *Deletion Time* |
|---|---|---|---|---|---|---|
| Phil | English | 1 | 1 | 423 | — | — |
| Phil | English | 3 | 4 | 423 | — | — |
| Norman | English | 1 | 2 | 423 | — | — |
| Norman | Calculus | 5 | 6 | 423 | — | — |

□

A new *Time-View* operator, TV = ($T_e$, $T_s$), is introduced that maps a temporal relation onto a snapshot relation. The Time-View operator can be thought of as a limited form of temporal selection that selects from the relation's state at transaction time $T_s$ those tuples with a valid time of $T_e$. Once the specified tuples are selected, however, the Time-View operator discards their implicit time attributes to construct a snapshot relation.

*EXAMPLE.* If we let TV = (1, 423), then

TV(R) =

| *Name* | *Course* |
|---|---|
| Phil | English |
| Norman | English |

□

The semantics of the five relational operators union, difference, join, selection, and projection is extended to handle both the valid and transaction time of tuples directly. These operators, like the Time-View operator, are all defined in terms of a transaction time $T_s$ and a valid time $T_e$. Input tuples are restricted to those tuples in an input relation's state at transaction time $T_s$ having a valid time of $T_e$; the valid times of all tuples that participate in an operation are thus guaranteed to overlap at time $T_e$. Each operator computes the valid time of its output tuples from the valid times of qualifying tuples in its input relations using either union or intersection semantics. For example, the union operator is defined using union semantics and the join operator is defined using intersection semantics. The valid time of tuples resulting from the difference operator, however, is left unspecified.

The Temporal Relational Model [Navathe & Ahmed 1986] allows both non-time-varying and time-varying attributes, but all of a relation's attributes must be the same type. Objects are snapshot relations, whose attributes are all non-time-varying, and historical relations, whose attributes are all time-varying. The end-points of the interval of validity of tuples in historical relations are recorded in two mandatory time attributes, *Time-Start* and *Time-End*. Hence, the structure of an historical relation in the Temporal Relational Model is the same as that of an historical relation in LEGOL 2.0, as shown on page 5. The set theoretic operators are retained and five additional operators on time-varying relations are introduced. The operators *Time-Slice*, *Inner Time-View*, and *Outer Time-View* are all forms of temporal selection. *TCJOIN* and *TCNJOIN* are both join operators defined using intersection semantics. Two other join operators, *TJOIN* and *TNJOIN*, are discussed. They retain the time-stamps of underlying tuples in their resulting tuples but are, therefore, outside the algebra (the domain of the operators contains objects not defined by the model).

Unlike the algebras discussed above, the Temporal Relational Algebra [Lorentzos & Johnson 1987A] associates time-stamps with individual attributes rather than with tuples. Although a time-stamp is normally associated with all the attributes in a tuple, a time-stamp may be associated with any non-empty subset of attributes in a tuple. Furthermore, no implicit or mandatory time-stamp attributes are assumed. Time-stamps are simply explicit, numeric-valued attributes. They represent either the chronon during which one or more attribute values are valid or a *boundary point* of the interval of validity for one or more attribute values. Several time-stamp attributes may also be used together to represent a chronon of nested granularity.

*EXAMPLES.* First, let R be an historical relation in the Temporal Relational Algebra on the relation scheme *Student* = {*Name*, *N-Start*, *N-Stop*, *Course*, *C-Start*, *C-Stop*}. Unlike the other algebras, the time-stamp attributes appear as explicit attributes in the relation scheme. Here we assume that the attributes *N-Start* and *N-Stop* represent the boundary points of the interval of validity for the attribute *Name* and the attributes *C-Start* and *C-Stop* represent the boundary points of the interval of validity for the attribute *Course*. Note, however, that we could have specified the same time-stamp attributes for both *Name* and *Course* in this example.

R =

| *Name* | *N-Start* | *N-Stop* | *Course* | *C-Start* | *C-Stop* |
|---|---|---|---|---|---|
| Phil | 1 | 2 | English | 1 | 2 |
| Phil | 3 | 5 | English | 3 | 5 |
| Norman | 1 | 3 | English | 1 | 3 |
| Norman | 5 | 7 | Calculus | 5 | 7 |

Unlike the other algebras, a tuple in the Temporal Relational Algebra is not considered valid at its right-most boundary point. Hence, the first tuple signifies that Phil was enrolled in English during the Fall semester 1980, but not during the Spring semester 1981.

Now let $R_1$ be an historical relation in the Temporal Relational Algebra on the relation scheme *Student* = {*Name*, *Course*, *Semester-Start*, *Semester-Stop*, *Week-Start*, *Week-Stop*}, where all four time-stamp attributes are associated with both *Name* and *Course*. Assume here that the granularity for the time-stamp attributes *Week-Start* and *Week-Stop* is a week relative to the first week of a semester.

$R_1$ =

| *Name* | *Course* | *Semester-Start* | *Semester-Stop* | *Week-Start* | *Week-Stop* |
|---|---|---|---|---|---|
| Phil | English | 1 | 2 | 1 | 9 |
| Phil | English | 3 | 5 | 1 | 17 |
| Norman | English | 1 | 3 | 1 | 9 |
| Norman | Calculus | 5 | 7 | 9 | 17 |

In this example, we specify the weeks during a semester when a student was enrolled in a course. For example, Phil was enrolled in English during the Fall semester 1980 for only the first 8 weeks of the semester. Note that the meaning of the *Week-Start* and *Week-Stop* attributes is relative to the *Semester-Start* and *Semester-Stop* attributes. □

The standard set-theoretic operations are retained in the Temporal Relational Algebra unchanged. Although no new time-oriented operations are introduced, three new operators, *EXTEND*, *UNFOLD*, and *FOLD*, which are defined in terms of the conventional relational operators, are introduced. These operators allow conversion between relations whose tuples contain two time-stamp attributes representing the end-points of the interval of validity of one or more attributes to equivalent relations whose tuples contain a single time-stamp attribute representing a chronon during which the same attributes are valid. Relations whose tuples contain only time-stamp attributes representing the end-points of intervals of validity are considered to be *folded* while relations whose tuples contain only time-stamp attributes representing individual chronons of validity are considered to be *unfolded*. Relations R and $R_1$ in the above examples are folded relations.

*EXAMPLE*. Let $R_2$ be an equivalent representation of $R_1$ in which the two time-stamp attributes *Semester-Start* and *Semester-Stop* have been unfolded onto a single time-stamp attribute *Semester*.

$R_2$ =

| *Name* | *Course* | *Semester* | *Week-Start* | *Week-Stop* |
|---|---|---|---|---|
| Phil | English | 1 | 1 | 9 |
| Phil | English | 3 | 1 | 17 |
| Phil | English | 4 | 1 | 17 |
| Norman | English | 1 | 1 | 9 |
| Norman | English | 2 | 1 | 9 |
| Norman | Calculus | 5 | 9 | 17 |
| Norman | Calculus | 6 | 9 | 17 |

We could now apply *UNFOLD* once more to unfold the attributes *Week-Start* and *Week-Stop* onto a single time-stamp attribute *Week*. The resulting relation would have 72 tuples. □

The Historical Relational Data Model [Clifford & Croker 1987] allows two types of objects: a set of chronons, termed a *lifespan*, and an historical relation, where each attribute in the relation scheme and each tuple in the relation is assigned a lifespan. A relation scheme in the Historical Relational Data Model is an ordered four-tuple containing a set of attributes, a set of key attributes, a function that maps attributes to their lifespans, and a function that maps attributes to their value domains. A tuple is an ordered pair containing the tuple's value and its lifespan. Attributes are not atomic-valued; rather, an attribute's value in a given tuple is a partial function from the domain of chronons onto the attribute's value domain, defined for the attribute's valid time (i.e., the intersection of the attribute and tuple lifespans).

*EXAMPLE*. R is an historical relation in the Historical Relational Data Model on the relation scheme *Student*, where $\{Name \rightarrow \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}, Course \rightarrow \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}\}$ is the function assigning lifespans to attributes.

R =

| ⟨ *Tuple Value* | | *Tuple Lifespan* ⟩ |
|---|---|---|
| *Name* | *Course* | |
| 1 → Phil<br>3 → Phil<br>4 → Phil | 1 → English<br>3 → English<br>4 → English | {1, 3, 4} |
| 1 → Norman<br>2 → Norman<br>5 → Norman<br>6 → Norman | 1 → English<br>2 → English<br>5 → Calculus<br>6 → Calculus | {1, 2, 5, 6} |

Because tuple lifespans are sets and because both Phil and Norman were never enrolled in more than one course at the same time, we are able to record each of their enrollment histories in a single tuple. If one had been enrolled in two or more courses at the same time. however, his total enrollment history could not have been recorded in a single tuple as attribute values are functions from a lifespan onto a value domain. Note also that we have chosen the most straightforward representation for an attribute whose value is a function. Because attribute values in both Clifford's algebra and Gadia's algebra, which we describe next, are functions, they have an arbitrary number of other physical representations. □

The standard set-theoretic operations are retained and several new time-oriented operations are introduced. *WHEN* maps a relation into its lifespan, where the lifespan of a relation is defined to be the union of the lifespans of its tuples (e.g., $\{1, 2, 3, 4, 5, 6\}$ in the above example). *SELECT-IF* is a form of temporal selection that selects tuples that are both valid and satisfy a given selection criterion at a specified time and *TIME-SLICE* is a form of temporal projection that restricts the tuple lifespans of its resulting tuples to some portion of their original lifespans. The operator *SELECT-WHEN* possesses features of both temporal selection and temporal projection; it is a variant of *SELECT-IF* that restricts the tuple lifespans of its resulting tuples to the times when they satisfy the selection condition. Finally, four variants of temporal join are introduced, each defined using intersection semantics.

Gadia's homogeneous (H) model [Gadia 1986] also allows two types of objects: *temporal elements* and historical relations. A temporal element is a finite union of disjoint intervals (effectively a set of chronons) and attribute values are functions from temporal elements onto attribute value domains. The model requires that all attribute values in a given tuple be functions on the same temporal element. This property, termed ***homogeneity***, ensures that a snapshot of an historical relation at time $t$ always produces a conventional snapshot relation without nulls.

*EXAMPLE.* R is an historical relation in Gadia's homogeneous model over the scheme *Student* = {*Name*, *Course*}.

R =

| *Name* | *Course* |
|---|---|
| $[1, 2) \cup [3, 5) \rightarrow$ Phil | $[1, 2) \cup [3, 5) \rightarrow$ English |
| $[1, 3) \cup [5, 7) \rightarrow$ Norman | $[1, 3) \rightarrow$ English<br>$[5, 7) \rightarrow$ Calculus |

Here the interval $[t_1, t_2)$ is the set of chronons $\{t_1, \cdots, t_2 - 1\}$. Again, we are able to record the enrollment histories of Phil and Norman in single tuples only because they were never enrolled in more than one course at the same time. □

An historical version of each of the five basic conventional relational operators is defined using snapshot semantics. For each historical operator, the snapshot of its resulting historical relation at time $t$ is required to equal the result obtained by applying the historical operator's relational counterpart to the snapshot of the underlying historical relations at time $t$. Two new operators are also introduced. One, *tdom*, maps either a tuple or a relation into its temporal domain, where the temporal domain of a tuple is its temporal element and the temporal domain of a relation is the union of its tuples' temporal elements. For example, the temporal domain of R above is [1, 7). The other operator, termed *temporal selection*, is a limited form of both temporal selection and temporal projection; it selects from a relation those tuples whose temporal elements overlap a specified temporal element and restricts attribute values in the resulting tuples to the intersection of their temporal elements and the specified temporal element.

Gadia's multihomogeneous (MH) model [Gadia 1986] is an extension of his homogeneous model in which all attribute values in a given tuple need not be defined over the same temporal element. Attribute values are required to be defined over the same temporal element only for specified subsets of attributes. Extension of the snapshot semantics for operators to account for multihomogeneous tuples is, however, left unspecified.

Tansel's historical algebra [Tansel 1986] allows only one type of object: the historical relation. Four types of attributes, however, are supported, the attributes of a relation need not be the same type, and attribute values in a given tuple need not be homogeneous. Attributes may be either non-time-varying or time-varying and they may be either atomic-valued or set-valued. The value of a time-varying, atomic-valued attribute is represented as a triplet containing an element from the attribute's value domain and the boundary points of its interval of existence while the value of a time-varying, set-valued attribute is simply a set of such triplets.

*EXAMPLE*. R is an historical relation in Tansel's algebra over the scheme *Student* = {*Name*, *Course*}, where *Name* is a non-time-varying, atomic-valued attribute and *Course* is a time-varying, set-valued attribute.

R =

| *Name* | *Course* |
|---|---|
| Phil | {⟨ [1, 2), English ⟩,<br>⟨ [3, 5), English ⟩} |
| Norman | {⟨ [1, 3), English ⟩,<br>⟨ [5, 7), Calculus ⟩} |

Because Tansel does not define time-varying attributes as functions, the enrollment history of a student can be recorded in a single tuple, even if the student was enrolled in two or more courses at some time. Note, however, that each interval of enrollment, even for the same course, must be recorded as a separate element of a time-varying, set-valued attribute. □

The conventional relational operators are extended to account for the temporal dimension of data and several new time-related operations are introduced. *PACK* combines tuples whose attribute values differ for a specified attribute but are otherwise equal. Conversely, *UNPACK* replicates a tuple for each element in one of its set-valued attributes. *T-DEC* decomposes a time-varying, atomic-valued attribute in an historical relation into three non-time-varying, atomic-valued attributes, representing the three components of the time-varying, atomic-valued attribute. Conversely, *T-FORM* combines three non-time-varying, atomic-valued attributes, representing a value and the boundary points of the value's interval of validity into a single time-varying, atomic-valued attribute. *DROP-TIME* discards the time components of a time-varying attribute. Finally, *SLICE*, *USLICE*, and *DSLICE*, are limited forms of temporal projection in which the time-stamp of a time-varying attribute is recomputed as the intersection, union, and difference, respectively, of its original time-stamp and the time-stamp of another specified attribute. If the recomputed time-stamp is empty, the tuple is discarded. Tansel also introduces a new operation, termed *enumeration*, to support aggregation [Tansel 1987]. The enumeration operator derives, for a set of chronons or intervals and an historical relation, a table of data to which aggregate operators (e.g., count, avg, min) can be applied.

*EXAMPLES*. Let $R_1$ be the historical relation, resulting from the unpacking of attribute *Course* of relation R in the previous example, over the scheme *Student* = {*Name*, *Course*}, where *Name* is a non-time-varying, atomic-valued attribute and *Course* is a time-varying, atomic-valued attribute.

$R_1 =$

| *Name* | *Course* |
|---|---|
| Phil | ⟨ [1, 2), English ⟩ |
| Phil | ⟨ [3, 5), English ⟩ |
| Norman | ⟨ [1, 3), English ⟩ |
| Norman | ⟨ [5, 7), Calculus ⟩ |

Now, let $R_2$ be the historical relation, resulting from the decomposition (*T-DEC*) of attribute *Course* of relation $R_1$, over the scheme *Student* = {*Name*, *Course*, $Course_L$, $Course_U$}, where *Name*, *Course*, $Course_L$, and $Course_U$ are all non-time-varying, atomic-valued attributes.

$R_2 =$

| *Name* | *Course* | $Course_L$ | $Course_U$ |
|---|---|---|---|
| Phil | English | 1 | 2 |
| Phil | English | 3 | 5 |
| Norman | English | 1 | 3 |
| Norman | Calculus | 5 | 7 |

□

The only type of object in our historical algebra [McKenzie & Snodgrass 1987C] is the historical relation. The value of an attribute is always an ordered pair whose components are a value from the attribute's value domain and a set of chronons. There is no requirement that the time-stamps of any of the attributes in a relation be homogeneous but relations are not allowed to have two tuples with the same value component for all their attributes (termed *value-equivalence*).

*EXAMPLE.* R is an historical relation in our algebra over the over the scheme *Student* = {*Name*, *Course*}.

R =

| *Name* | *Course* |
|---|---|
| ⟨Phil, {1, 3, 4}⟩ | ⟨English, {1, 3, 4}⟩ |
| ⟨Norman, {1, 2}⟩ | ⟨English, {1, 2}⟩ |
| ⟨Norman, {5, 6}⟩ | ⟨Calculus, {5, 6}⟩ |

In our algebra, Phil's enrollment in English must be recorded in a single tuple, otherwise the value-equivalence property would be violated. Norman's enrollment history, however, cannot be recorded in a single tuple; an attribute may be assigned only one value from its value domain. □

The conventional relational operators are extended to account for the temporal dimension of data directly and preserve the value-equivalence property of historical relations. One new operator, *historical derivation*, is introduced specifically to handle temporal selection and temporal projection functions.

Table 1 and Table 2 are a summary of the features of the nine algebras described above. These tables show the range of solutions chosen by the developers of the algebras to the five basic design decisions introduced on page 4. Because several of the algebras have similar names and others are unnamed, we use the names of the developers to refer to the algebras hereafter for clarity. Table 1 categorizes the algebras according to their representation of time. Note that Clifford's algebra appears twice in Table 1 as it associates time-stamps with attributes in a relation scheme as well as tuples in a relation instance. Table 2 describes other basic features of the types of objects defined and operations allowed in the algebras. The second column lists object types and the third column describes the structure of attributes. The fourth column indicates whether the algebras retain the set-theoretic semantics of the five basic relational operators or extend the operators to deal with time directly. The fifth column lists new operators introduced specifically to handle the temporal dimension of the phenomena being modeled.

In the next section we discuss a set of criteria for evaluating historical algebras. Then, in Section 3, we evaluate these nine algebras against the criteria.

## 2 Criteria for Evaluating Historical Algebras

Although several historical algebras have been proposed, current research has not focused on defining criteria for evaluating the relative merit of these historical algebras. Only Clifford presents a list of specific properties desirable of an historical algebra [Clifford & Tansel 1985]. He identifies five fundamental, conceptual goals, which will be discussed in detail shortly. These goals alone are insufficient to evaluate the relative merit of proposed historical algebras. A more comprehensive set of specific, objective criteria is needed. In this section, we identify 21 such criteria for evaluating historical algebras. First, we introduce the criteria. With each criterion, we indicate its source, if relevant. Next, we discuss our reasons for not including as criteria several other properties of

|  | TIME-STAMP REPRESENTATION |  |  |
|---|---|---|---|
|  | single chronon | interval (two chronons) | set of chronons |
| Time-stamped Tuples | Jones, et al.<br>Ben-Zvi<br>Navathe & Ahmed |  | Clifford & Croker |
| Time-stamped Attributes | Lorentzos & Johnson | Tansel | Clifford & Croker<br>Gadia H<br>Gadia MH<br>McKenzie & Snodgrass |

Table 1: Representation of Time in the Algebras

historical algebras. Then, we examine incompatibilities among the criteria.

For clarity, we hereafter represent an historical operator as $\hat{op}$ to distinguish it from its snapshot algebra counterpart *op*.

## 2.1 Criteria

Table 3 is an alphabetical listing of criteria for evaluating historical algebras. Included in this list are properties of historical algebras that have been advocated by others as well as those properties that seem reasonable to us. The list is restricted to only those properties that are well-defined, have an objective basis for being evaluated, and are arguably beneficial. No historical algebra can have all these properties as certain subsets of the properties are incompatible. An historical algebra can, however, have a maximal subset of properties from Table 3 that are compatible.

*All attributes in a tuple are defined for the same interval(s)* [Gadia 1986]. This requirement, termed *homogeneity* by Gadia, assumes that attributes, rather than tuples, are time-stamped and that attributes are set-valued, rather than atomic-valued. Although attributes may change value at different times (i.e., asynchronous attributes), all attributes in a tuple must be defined for the same interval(s). Requiring that all attributes in a tuple be defined for the same interval(s) simplifies definition of the algebra. Operators need not be redefined to handle the time dimension directly. Rather, the algebra can be defined in terms of the conventional relational operators using snapshot semantics, even if set-valued attributes are allowed. Also, problems that arise when disjoint attribute time-stamps are allowed (e.g., how to handle non-empty time-stamps for some, but not all, attributes) need not be considered.

*Consistent extension of the snapshot algebra* [Clifford & Tansel 1985]. The expressive power of the historical algebra should subsume that of the snapshot algebra. The historical algebra should

|  |  |  | Standard | New |
|---|---|---|---|---|
| Algebra | Objects | Attributes | Operations | Operations |
| Jones, et al. | historical relations | atomic-valued | retained | time intersection,<br>one-sided time intersection,<br>time union,<br>time difference,<br>time-set membership |
| Ben-Zvi | snapshot relations,<br>historical relations | atomic-valued | extended | Time-View |
| Navathe & Ahmed | snapshot relations,<br>historical relations | atomic-valued | retained | Time-Slice,<br>Inner Time-View,<br>Outer Time-View,<br>TCJOIN, TCNJOIN |
| Lorentzos & Johnson | snapshot relations | atomic-valued | retained | Extend, Fold, Unfold |
| Clifford & Croker | lifespans,<br>historical relations | functional | retained | When, Select-If,<br>Select-When,<br>Time-Slice,<br>4 Join Operators |
| Gadia H,<br>Gadia MH | temporal elements,<br>historical relations | functional | snapshot semantics | tdom,<br>Temporal Selection |
| Tansel | historical relations | atomic-valued<br>set-atomic-valued<br>triplet-valued<br>set-triplet-valued | extended | Pack, Unpack,<br>T-Dec, T-Form,<br>Drop-Time,<br>Slice, Uslice, Dslice,<br>Enumeration |
| McKenzie & Snodgrass | historical relations | ordered pairs | extended | Temporal Derivation |

Table 2: Objects and Operations in the Algebras

- All attributes in a tuple are defined for the same interval(s)
- Consistent extension of the snapshot algebra
- Data periodicity is supported
- Each collection of valid attribute values is a valid tuple
- Each set of valid tuples is a valid relation
- Formal semantics is specified
- Has the expressive power of an historical calculus
- Historical data loss is *not* an operator side-effect
- Implementation exists
- Includes aggregates
- Is, in fact, an algebra
- Model doesn't require *null* attribute values
- Optimization strategies are available
- Reduces to the snapshot algebra
- Restricts relations to first-normal form
- Supports a 3-dimensional view of historical relations and operations
- Supports basic algebraic tautologies:

  $Q \hat{\cup} R = R \hat{\cup} Q$

  $Q \hat{\times} R = R \hat{\times} Q$

  $\hat{\sigma}_{F_1}(\hat{\sigma}_{F_2}(R)) = \hat{\sigma}_{F_2}(\hat{\sigma}_{F_1}(R))$

  $Q \hat{\cup} (R \hat{\cup} S) = (Q \hat{\cup} R) \hat{\cup} S$

  $Q \hat{\times} (R \hat{\times} S) = (Q \hat{\times} R) \hat{\times} S$

  $Q \hat{\times} (R \hat{\cup} S) = (Q \hat{\times} R) \hat{\cup} (Q \hat{\times} S)$

  $Q \hat{\times} (R \hat{-} S) = (Q \hat{\times} R) \hat{-} (Q \hat{\times} S)$

  $\hat{\sigma}_F(Q \hat{\cup} R) = \hat{\sigma}_F(Q) \hat{\cup} \hat{\sigma}_F(R)$

  $\hat{\sigma}_F(Q \hat{-} R) = \hat{\sigma}_F(Q) - \hat{\sigma}_F(R)$

  $\hat{\pi}_X(Q \hat{\cup} R) = \hat{\pi}_X(Q) \hat{\cup} \hat{\pi}_X(R)$

  $Q \hat{\cap} R = Q \hat{-} (Q \hat{-} R)$
- Supports static attributes
- Tuples, not attributes, are time-stamped
- Unique representation for each historical relation
- Unisorted (not multisorted)

Table 3: Criteria for Evaluating Historical Algebras

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\begin{matrix}\text{Snapshot}\\ \text{Relation}\\ S\end{matrix} \arrow[r, "T(S)"] \arrow[d, "\begin{matrix}\text{Snapshot}\\ \text{Operator}\\ op\end{matrix}"'] & \begin{matrix}\text{Historical}\\ \text{Relation}\\ R_1\end{matrix} \arrow[d, "\begin{matrix}\text{Analogous}\\ \text{Historical}\\ \text{Operator}\\ \widehat{op}\end{matrix}"] \\
op\ (S) \arrow[r, "T(op\ (S))"] & R_2 = \widehat{op}(R_1)
\end{tikzcd}$$

Figure 1: Outline of Equivalence Proof

be at least as powerful as the snapshot algebra. Any relation or algebraic expression that can be represented in the snapshot model should have a counterpart in the historical model. Thus the historical algebra should provide, as a minimum, an historical counterpart for each of the five operators that serve to define the snapshot algebra: union, difference, cartesian product, projection, and selection [Ullman 1982]. Furthermore, the historical relation resulting from the application of one of these snapshot operators to a snapshot relation and conversion of the resulting relation to its historical counterpart should be equivalent to the historical relation resulting from application of the snapshot operator's historical counterpart to the snapshot relation's historical counterpart. If we assume that the function T transforms a snapshot relation into its historical counterpart, then Figure 1 illustrates this equivalence proof.

*Data periodicity is supported* [Lorentzos & Johnson 1987A]. Periodicity is a property of many real-world phenomena. Rather than occurring just once in time or at randomly spaced times, these phenomena recur at regular intervals over a specific interval in time. For example, a person may have worked from 8:00 a.m. until 5:00 p.m. each day, Monday through Friday, for a particular month. Ideally, an historical data model should be able to represent such periodic phenomena without having to specify the time of each of their occurrences.

*Each collection of valid attribute values is a valid tuple.* In the snapshot model, the value of an attribute is independent of the value of other attributes in a tuple, except for key and functional dependency constraints. The same should be true of the historical model. If we extend the snapshot model so that a time-stamp is assigned to each attribute, we should extend the concept of attribute independence to include the time-stamp of the attribute as well as the value of the attribute. Within a tuple, the value or time-stamp of one attribute should not restrict arbitrarily the value or time-stamp of another attribute. Limiting valid tuples to some subset of the tuples that could be formed from valid attribute values adds a degree of complexity to the historical model not found in the snapshot model.

*Each set of valid tuples is a valid relation.* In the snapshot model, every set of tuples that satisfies key and functional dependency constraints is a valid relation. The same should be true of the historical model. Imposing additional inter-tuple constraints, which further restrict the set of valid relations, adds another degree of complexity to the historical model not found in the snapshot model.

*Formal semantics is specified.* Concise, mathematical definitions for all object types and operations are needed. Without such definitions, the meaning of algebraic operations is unclear. Also, evaluation of the algebra is impossible.

*Has the expressive power of an historical calculus* [Gadia 1986]. There should exist an historical calculus whose expressive power is subsumed by that of the algebra. Calculus-based historical query languages then can be developed for which the algebra can serve as the underlying evaluation mechanism.

*Historical data loss is not an operator side-effect.* Historical data are lost if an operator removes valid-time information, contained in underlying relations, from its resulting relation. Data loss becomes an operator side-effect if the removal of that valid-time information is not the purpose of the operator. For example, suppose an historical algebra allows attribute time-stamping but requires closure under Gadia's homogeneous restriction (i.e., the valid times associated with each attribute value in a tuple must be identical). To ensure closure under cartesian product, assume that cartesian product is defined using intersection semantics. Now consider the cartesian product of two relations with attribute time-stamping, relation A defined over the scheme *Student* = {*Name*, *Course*}, and relation B defined over the scheme *Home* = {*Name*, *State*}.

A =

| *Name* | *Course* |
|---|---|
| ⟨ Phil, {1, 3, 4} ⟩ | ⟨ English, {1, 3, 4} ⟩ |

B =

| *Name* | *State* |
|---|---|
| ⟨ Phil, {1, 2, 3} ⟩ | ⟨ Kansas, {1, 2, 3} ⟩ |

A$\tilde{\times}$B =

| $Name_S$ | *Course* | $Name_H$ | *State* |
|---|---|---|---|
| ⟨ Phil, {1, 3} ⟩ | ⟨ English, {1, 3} ⟩ | ⟨ Phil, {1, 3} ⟩ | ⟨ Kansas, {1, 3} ⟩ |

Note the loss of valid-time information associated with Phil's enrollment in English at time 4 and his residency in Kansas at time 2. Historical algebras that allow such loss of historical data as an operator side-effect cannot support historical queries. If the algebra supports historical queries, the algebra must not allow loss of historical data as an operator side-effect; all valid-time information input to an operator must be preserved in the operator's output unless the operation being performed (e.g., difference, intersection) dictates removal.

*Implementation exists.* Semantic deficiencies, inconsistencies, and inefficiencies are often revealed during implementation. Therefore, it is desirable that the algebra have been implemented.

*Includes aggregates.* The historical model should provide formal semantics for historical versions of standard aggregate (e.g., sum, count, min, max) operations.

*Is, in fact, an algebra* [Clifford & Tansel 1985]. This criterion is fundamental. Any historical algebra should define the types of objects supported and the allowable operations on object instances of each defined type. In addition, all legal operations should be closed.

*Model doesn't require null attribute values.* Restriction of attribute values to non-*null* values is consistent with the snapshot model and greatly simplifies the semantics of the algebra.

*Optimization strategies are available.* Except for semantics, implementation efficiency is the most important feature of an historical algebra. If an algebra cannot be implemented efficiently, it will have no practical application for the development of historical query languages. Strategies for simplification of algebraic expressions corresponding to queries should be available. Note that the availability of basic algebraic tautologies already provides algebraic transformation optimizations.

*Reduces to the snapshot algebra* [Snodgrass 1987]. The semantics of the algebra should be consistent with the intuitive view of a snapshot relation as a 2-dimensional slice of a 3-dimensional historical relation at a time $t$. Hence, for all historical operators, the snapshot relation obtained by applying an historical operator to an historical relation and then taking a snapshot should be equivalent to the relation obtained by taking a snapshot of the historical relation and applying the analogous relational operator to the resulting snapshot relation. Figure 2 illustrates this reduction proof.

*Restricts relations to first-normal form.* The snapshot algebra owes much of its simplicity to the restriction of relations to first-normal form. The historical algebra should retain this property.

*Supports a 3-dimensional view of historical relations and operations* [Ariav 1986, Ariav & Clifford 1986, Clifford & Tansel 1985]. Almost all proposals for extending the snapshot model to incorporate valid time represent historical relations as space-filling solids, where the additional, third dimension is valid time. Although these space-filling solids are not true cubes, they do possess geometric properties similar to those of cubes. For example, consider the historical relation R over the scheme $Home = \{Name, Course\}$ with attribute time-stamping.

R =

| *Name* | *Course* |
|---|---|
| $\langle$Phil, $\{1, 3, 4\}\rangle$ | $\langle$English, $\{1, 3, 4\}\rangle$ |
| $\langle$Norman, $\{1, 2\}\rangle$ | $\langle$English, $\{1, 2\}\rangle$ |
| $\langle$Norman, $\{5, 6\}\rangle$ | $\langle$Calculus, $\{5, 6\}\rangle$ |

Figure 3 is a graphical representation of this relation. Clearly, this representation of R can be viewed as a space-filling solid with geometric properties similar to that of a cube.

$$\begin{tikzcd}
\text{Historical Relation} \arrow[r, "\textbf{SNAPSHOT}_t"] \arrow[d, "\text{Historical Operator}"'] & \text{Snapshot Relation} \arrow[d, "\text{Analogous Snapshot Operator}"] \\
R \arrow[r, "\textbf{SNAPSHOT}_t"] & S_1 = S_2
\end{tikzcd}$$

Figure 2: Outline of Reduction Proof

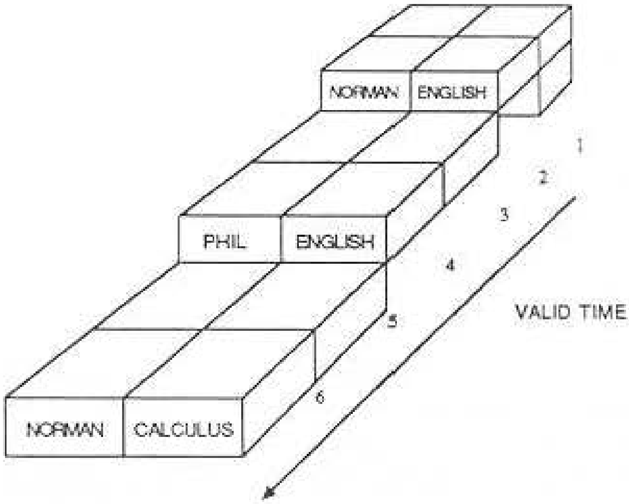

Figure 3: Historical Relation

If we accept this 3-dimensional representation as a high-level, user-oriented model of historical relations, then each operation defined on historical relations should have an interpretation, consistent with its semantics, in accordance with this conceptual framework. The definitions of operations should be consistent with the conceptual view that these operations manipulate cubic solids. For example, the difference operator should take two cubic solids (i.e., historical relations) and produces a third cubic solid that represents the mass (i.e., total historical information) present in the first cubic solid but not present in the second cubic solid. Likewise, the cartesian product operator should take two cubic solids and produce a third cubic solid such that each unit of mass (i.e., tuple) in the first cubic solid is concatenated with a unit of mass in the second cubic solid to form a unit of mass in the third cubic solid. This description of operations on historical relations as "volume" operations on cubic solids is consistent not only with the conceptual view of historical relations as cubic solids but also with the semantics of the individual snapshot algebraic operations as operations on 2-dimensional tables, extended to account for the additional dimension represented by valid time.

*Supports basic algebraic tautologies.* The following commutative, associative, and distributive tautologies, which hold for and in some sense define the snapshot operators, should also hold for their historical counterparts.

$$Q \hat{\cup} R = R \hat{\cup} Q$$

$$Q \hat{\times} R = R \hat{\times} Q$$

$$\hat{\sigma}_{F_1}(\hat{\sigma}_{F_2}(R)) = \hat{\sigma}_{F_2}(\hat{\sigma}_{F_1}(R))$$

$$Q \hat{\cup} (R \hat{\cup} S) = (Q \hat{\cup} R) \hat{\cup} S$$

$$Q \hat{\times} (R \hat{\times} S) = (Q \hat{\times} R) \hat{\times} S$$

$$Q \hat{\times} (R \hat{\cup} S) = (Q \hat{\times} R) \hat{\cup} (Q \hat{\times} S)$$

$$Q \hat{\times} (R \hat{-} S) = (Q \hat{\times} R) \hat{-} (Q \hat{\times} S)$$

$$\hat{\sigma}_F(Q \hat{\cup} R) = \hat{\sigma}_F(Q) \hat{\cup} \hat{\sigma}_F(R)$$

$$\hat{\sigma}_F(Q \hat{-} R) = \hat{\sigma}_F(Q) - \hat{\sigma}_F(R)$$

$$\hat{\pi}_X(Q \hat{\cup} R) = \hat{\pi}_X(Q) \hat{\cup} \hat{\pi}_X(R)$$

$$Q \hat{\cap} R = Q \hat{-} (Q \hat{-} R)$$

Included in this list are the commutative, associative, and distributive tautologies involving only union, difference, and cartesian product that are defined in set theory [Enderton 1977]. Also included in this list are the non-conditional commutative laws involving selection and projection presented by Ullman [Ullman 1982]. Finally, the definition of the intersection operator in terms of the difference operator, which holds for the snapshot algebra, should also hold for the historical algebra.

*Supports static attributes* [Clifford & Tansel 1985, Navathe & Ahmed 1986]. The algebra should allow for attributes whose role in a tuple is not restricted by time. This feature allows the historical model to be applied to environments in which the values of certain attributes in a tuple are time-dependent while the values of other attributes in the tuple are not time-dependent.

*Tuples, not attributes, are time-stamped.* Time-stamping tuples, rather than attributes, simplifies the semantics of the algebra. Operators need not be defined to handle disjoint attribute time-stamps but rather can be defined in terms of the conventional relational operators using snapshot semantics.

*Unique representation for each historical relation.* In the snapshot model, there is a unique representation for each valid snapshot relation. Likewise, there should be a unique representation for each valid historical relation. Failure of an algebra to satisfy this criterion can complicate the semantics of the operators, require inefficient implementations, and possibly restrict the class of database retrievals that can be supported. For example, consider the following relations on the scheme $Student = \{Name, Course\}$ with attribute time-stamping.

A =

| *Name* | *Course* |
|---|---|
| ⟨Phil, {1, 2}⟩ | ⟨English, {1, 2}⟩ |
| ⟨Phil, {3, 4}⟩ | ⟨English, {3, 4}⟩ |

B =

| *Name* | *Course* |
|---|---|
| ⟨Phil, {1, 2, 3, 4}⟩ | ⟨English, {1, 2, 3, 4}⟩ |

C =

| *Name* | *Course* |
|---|---|
| ⟨Phil, {5, 6}⟩ | ⟨English, {5, 6}⟩ |

D =

| *Name* | *Course* |
|---|---|
| ⟨Phil, {2, 3}⟩ | ⟨English, {2, 3}⟩ |

Clearly, the information content of relations A and B is identical; the information content of relation C is a continuation of the information in both A and B; and the information content of relation D is a subset of that contained in both A and B. However, what is the semantics of A ∪ C? Does the output relation contain three tuples, two tuples, or just one tuple? Similarly, what is the semantics of A ∪ D? Is the single tuple in D represented in the output relation or is it absorbed by the two tuples in A? Also, if we want to retrieve the name of all students who were enrolled in English from time 2 to time 4, do we get the same result if we apply this query to relations A and B? Retrieval of "Phil," which is the intuitively correct result when applying this query to A, requires tuple selection based on information contained in more than one tuple, a significant departure from the semantics of the selection operation in the snapshot algebra. Thus, a selection operator with significantly more complicated semantics would be required to produce results that are correct intuitively. Moreover, the implementation of such a selection operator may be impractical because of the many cases that would have to be considered during the selection process.

*Unisorted (not multisorted).* In the snapshot algebra, all operators take as input and provide as output a single type of object, the snapshot relation. If possible, an historical algebra should also be unisorted. A multisorted algebra would introduce a degree of complexity in the historical model not found in the snapshot model.

## 2.2 Properties not Included as Criteria

The following properties are either subsumed by properties in Table 3, are not well-defined, or have no objective basis for being evaluated. Hence, they are not included as criteria.

*Disallows tuples with duplicate attribute values.* If attributes are time-stamped, then this requirement is subsumed by the criterion that the algebra have a unique representation for each historical relation. There would be many different equivalent representations for most historical relations if tuples with duplicate attribute values were allowed. For example, the following are only two of several equivalent representations of a relation A over the scheme $Home = \{Name, State\}$ with attribute time-stamping.

A =

| *Name* | *State* |
|---|---|
| ⟨Norman, {1, 2, 5, 6}⟩ | ⟨Virginia, {1, 2, 5, 6}⟩ |

A =

| *Name* | *State* |
|---|---|
| ⟨Norman, {1, 2}⟩ | ⟨Virginia, {1, 2}⟩ |
| ⟨Norman, {5, 6}⟩ | ⟨Virginia, {5, 6}⟩ |

*Supports historical queries (valid time)* [Snodgrass 1987]. An historical algebra supports historical queries if information valid over a chronon can be derived from information in underlying relations valid over other chronons, much as the snapshot algebra allows for the derivation of information about entities or relationships from information in underlying relations about other entities or relationships. Satisfaction of this criterion implies that the algebra allows units of related information, possibly valid over disjoint chronons, to be combined into a single related unit of information possibly valid over some other chronon. Support for such a capability requires the presence, in the algebra, of a cartesian product or join operator that concatenates tuples, independent of their valid times, and preserves, in the resulting tuple, the valid-time information for each of the underlying tuples. Hence, this requirement is subsumed by the criteria that the algebra be a consistent extension of the snapshot algebra and historical data loss not be an operator side-effect.

*Supports non-homogeneous relations* [Gadia 1986]. If the algebra is closed and supports historical queries, it must support non-homogeneous relations (i.e., relations having tuples whose attribute values are allowed to have different valid times). Therefore, this requirement is subsumed by the criteria that the algebra, in fact, be an algebra, the algebra be a consistent extension of the snapshot algebra, and historical data loss not be an operator side-effect.

*Homogeneous tuples are valid tuples.* This requirement is subsumed by the requirement that the algebra support non-homogeneous relations.

*Update semantics* [Snodgrass 1987]. If the algebra provides union and difference operators, the algebra supports the replace, delete, and append operations found in all query languages. Therefore, this requirement is subsumed by the criterion that the algebra be a consistent extension of the snapshot algebra. Further, support for update requires the modeling of transaction time, which is orthogonal to valid time [Snodgrass & Ahn 1986].

*Minimal extension of the snapshot algebra.* This requirement is too vague to be considered a criterion, unless qualified. Criteria such as "consistent extension of the snapshot algebra," "reduces to snapshot algebra," and "unique representation for each historical relation," all imply a minimal extension to the snapshot algebra.

*Retains the simplicity of the snapshot model.* Again, this requirement is too vague to be considered a criterion, unless qualified. Note that specific aspects of simplicity are implied by other properties that are well-defined (e.g., "model doesn't require *null* attribute values" and "algebra is unisorted").

*The model is semantically complete* [Clifford & Tansel 1985]. The model should serve as a standard for defining historical completeness (i.e., an extension of Codd's notion of completeness in the snapshot model). This requirement has no objective basis for evaluating models as there is no consensus definition of historical completeness.

## 2.3 Incompatibilities

Not all the criteria listed in Table 3 are compatible. There are certain subsets of criteria that no historical algebra can satisfy. In this section, we examine the incompatibilities among criteria.

The criterion that the algebra support a 3-dimensional view of historical relations and operations is incompatible with the criteria that

- Tuples, not attributes, be time-stamped,
- All attributes in a tuple be defined for the same interval(s), and
- The tautology $Q \hat{\times} (R \hat{-} S) = (Q \hat{\times} R) \hat{-} (Q \hat{\times} S)$ hold.

First, no historical algebra can support a 3-dimensional view of historical relations and operations and also time-stamp tuples. For the algebra to support a 3-dimensional view of historical relations and operations, the algebra must support a cartesian product or join operator that concatenates tuples, independent of their valid times, and preserves, in the resulting tuple, the valid-time information for each of the underlying tuples. Yet, if the cartesian product operator assigns different time-stamps to attributes in its output tuples, the criterion that tuples, not attributes, be time-stamped cannot be satisfied. Hence, no historical algebra can satisfy both of these criteria.

Secondly, no historical algebra can support a 3-dimensional view of historical relations and operations and also require that all attributes in a tuple be defined for the same interval(s). If the cartesian product operator required that all attributes in a resulting tuple be defined over the same interval(s), arbitrary valid-time information associated with the attributes of the underlying tuples could not be preserved and the criterion that the algebra support a 3-dimensional view of historical relations and operations could not be satisfied. Yet, if the cartesian product operator preserved the valid-time information for the attributes of the underlying tuples in the resulting tuple, attributes in the resulting tuple would be defined for different intervals and the criterion that all attributes in a tuple be defined for the same interval(s) could not be satisfied.

Thirdly, no historical algebra can support a 3-dimensional view of historical relations and operations and also support the distributive property of cartesian product over difference. For example, consider the following single-tuple relations over the scheme *Student* = {*Name*, *Course*} with attribute time-stamping.

A =

| *Name* | *Course* |
|---|---|
| ⟨Phil, {1, 2, 3}⟩ | ⟨Math, {1, 2, 3}⟩ |

B =

| *Name* | *Course* |
|---|---|
| ⟨Norman, {1, 2}⟩ | ⟨English, {1, 2}⟩ |

C =

| *Name* | *Course* |
|---|---|
| ⟨Norman, {2}⟩ | ⟨English, {2}⟩ |

Figure 4 illustrates the representation of historical relations as 3-dimensional solids in calculating $A\hat{\times}(B\hat{-}C)$ and $(A\hat{\times}B)\hat{-}(A\hat{\times}C)$, respectively. The results of these calculations are shown below.

$A\hat{\times}(B\hat{-}C)$ =

| $\mathit{Name}_1$ | $\mathit{Course}_1$ | $\mathit{Name}_2$ | $\mathit{Course}_2$ |
|---|---|---|---|
| ⟨Phil, {1, 2, 3}⟩ | ⟨Math, {1, 2, 3}⟩ | ⟨Norman, {1}⟩ | ⟨English, {1}⟩ |

$(A\hat{\times}B)\hat{-}(A\hat{\times}C)$ =

| $\mathit{Name}_1$ | $\mathit{Course}_1$ | $\mathit{Name}_2$ | $\mathit{Course}_2$ |
|---|---|---|---|
| ⟨Phil, ∅⟩ | ⟨Math, ∅⟩ | ⟨Norman, {1}⟩ | ⟨English, {1}⟩ |

This example shows that the criterion that the distributive property of cartesian product over difference hold is incompatible with the criterion that the algebra support a 3-dimensional view of historical relations and operations.

There are two other incompatibilities among the criteria in Table 3. First, the criterion that each set of valid tuples be a valid relation is incompatible with the criterion that there be a unique representation for each historical relation. If every set of valid tuples were allowed to be a

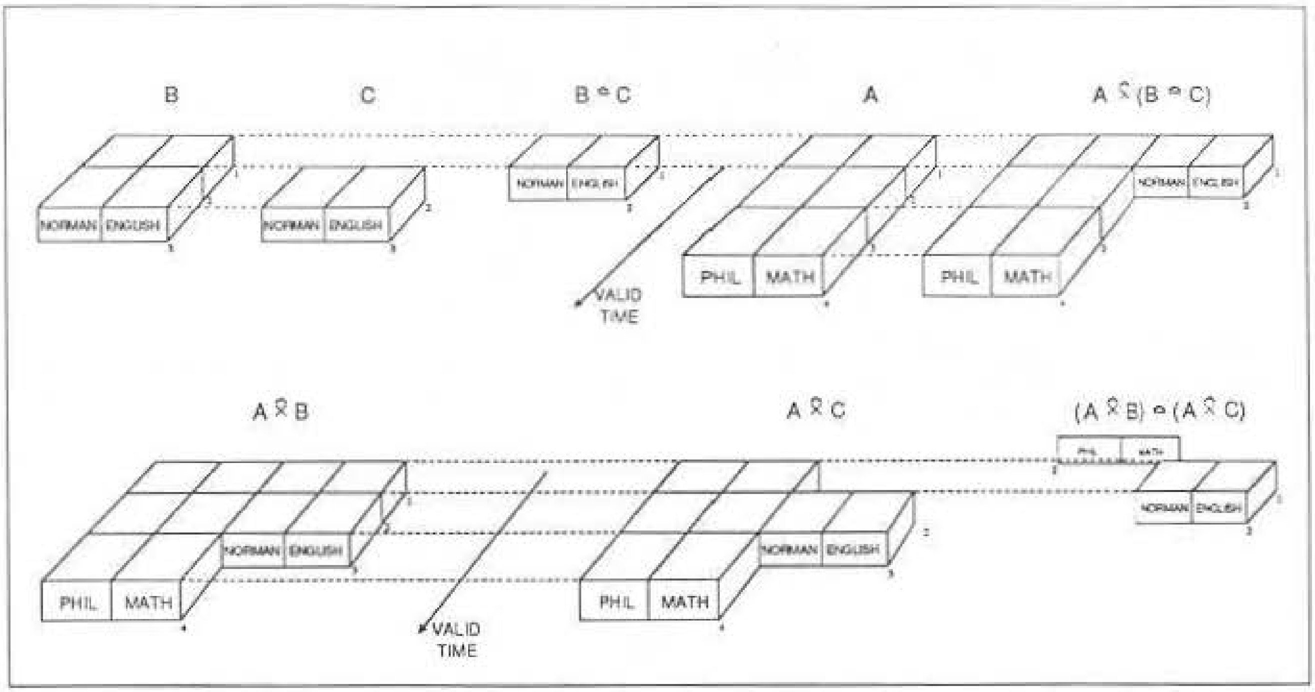

Figure 4: $A\hat{\times}(B\hat{-}C)$ and $(A\hat{\times}B)\hat{-}(A\hat{\times}C)$

valid relation, the algebra could not have a unique representation for each historical relation. For example, the following are only two of several equivalent representations of a relation A over the scheme $Home = \{Name, State\}$ with attribute time-stamping.

A =

| *Name* | *State* |
|---|---|
| ⟨Norman, {1, 2, 3, 4}⟩ | ⟨Virginia, {1, 2, 3, 4}⟩ |

A =

| *Name* | *State* |
|---|---|
| ⟨Norman, {1, 2}⟩ | ⟨Virginia, {1, 2}⟩ |
| ⟨Norman, {3, 4}⟩ | ⟨Virginia, {3, 4}⟩ |

Yet, if the algebra allowed only one of these representations, there would be sets of valid tuples that would not be valid relations. Hence, no historical algebra can satisfy both of these criteria.

Finally, the criteria that the algebra support a 3-dimensional view of historical relations and operations, have a unique representation for each historical relation, and restrict relations to first-normal-form are incompatible. An algebra can be defined that satisfies any two of these criteria, but no algebra can be defined that satisfies all three criteria. For example, consider the following two single-tuple relations over the scheme $Home = \{Name, State\}$ with attribute time-stamping.

A =

| *Name* | *State* |
|---|---|
| ⟨ Phil, {1, 2, 3} ⟩ | ⟨ Kansas, {1, 2, 3} ⟩ |

B =

| *Name* | *State* |
|---|---|
| ⟨ Phil, {2} ⟩ | ⟨ Kansas, {2} ⟩ |

To define difference so that $A \hat{-} B$ can be calculated consistent with the conceptual model of historical operators as 3-dimensional operators on cubic solids, the algebra must allow tuples with duplicate attribute values in a relation

$A \hat{-} B$ =

| *Name* | *State* |
|---|---|
| ⟨ Phil, {1} ⟩ | ⟨ Kansas, {1} ⟩ |
| ⟨ Phil, {3} ⟩ | ⟨ Kansas, {3} ⟩ |

or allow the time-stamp associated with a tuple to be non-atomic (i.e., a set of intervals rather than a single interval).

$A \hat{-} B$ =

| *Name* | *State* |
|---|---|
| ⟨ Phil, {1, 3} ⟩ | ⟨ Kansas, {1, 3} ⟩ |

Thus, to support a 3-dimensional view of historical relations and operations and disallow tuples with duplicate attribute values, which is implied by the criterion that the algebra have a unique representation for each historical relation (if attributes are time-stamped), the algebra must allow non-first-normal-form relations.

The five incompatibilities described above all involve at least one of the two criteria

- Supports a 3-dimensional view of historical relations and operations and
- Unique representation for each historical relation.

Table 4 summarizes the effect satisfaction of these two criteria has on the algebra's ability to satisfy other criteria. Note that if the algebra satisfies neither of these criteria, then it can satisfy all the other criteria. If, however, the algebra satisfies both of these criteria, then there are five criteria that it cannot satisfy.

|  |  | Supports a 3-dimensional view of historical relations and operations? |  |
|---|---|---|---|
|  |  | No | Yes |
| Unique representation for each historical relation? | No | No restrictions. | All attributes in a tuple cannot be defined over the same interval(s).<br>The distributive property of cartesian product over difference cannot hold.<br>Tuple time-stamping cannot be used. |
|  | Yes | Each set of valid tuples cannot be a valid relation. | All attributes in a tuple cannot be defined over the same interval(s).<br>The distributive property of cartesian product over difference cannot hold.<br>Tuple time-stamping cannot be used.<br>Each set of valid tuples cannot be a valid relation.<br>Relations cannot be restricted to first-normal-form. |

Table 4: Incompatibilities Among Criteria

## 3 An Evaluation of Historical Algebras

In this section we evaluate nine historical algebras against the criteria presented in the previous section. We consider Ben-Zvi's Time Relational Model [Ben-Zvi 1982], Clifford's Historical Relational Data Model [Clifford & Croker 1987], Gadia's homogeneous (H) and multihomogeneous (MH) models [Gadia 1986], Jones' extension to the snapshot algebra to support time-oriented operations for LEGOL [Jones et al. 1979], Lorentzos' Temporal Relational Algebra [Lorentzos & Johnson 1987A], our historical algebra [McKenzie & Snodgrass 1987C], Navathe's Temporal Relational Model [Navathe & Ahmed 1986], and Tansel's historical algebra [Tansel 1986]. Table 5 summarizes the evaluation of these nine proposals against the criteria. We did not include TERM [Klopprogge 1981] and PDM [Manola & Dayal 1986], both of which include support for time, in this evaluation as they are temporal extensions of other data models. TERM is an extension of the Entity-Relationship model and PDM is an extension of the entity-oriented Daplex functional data model.

### 3.1 Conflicting Criteria

We first evaluate the historical algebras against the seven criteria introduced in the previous section that are not all compatible. Because no algebra can satisfy all seven of these criteria, we term the criteria *conflicting* criteria.

|  | Ben-Zvi | Clifford | Gadia | Gadia | Jones | Lorentzos | McKenzie | Navathe | Tansel |
|---|---|---|---|---|---|---|---|---|---|
|  |  | & |  |  |  | & | & | & |  |
| *Conflicting* Criteria |  | Croker | H | MH | et al. | Johnson | Snodgrass | Ahmed |  |
| All attributes in a tuple defined for same interval(s) | △ | △ | √ | △ | △ | △ | △ | △ | △ |
| Each set of valid tuples is a valid relation | √ | √ | √ | √ | √ | √ | △ | △ | √ |
| Restricts relations to first-normal form | √ | △ | △ | △ | √ | √ | △ | √ | △ |
| Supports a 3-D view of historical relations & operations | △ | △ | △ | ? | △ | △ | √ | △ | ? |
| Supports basic algebraic tautologies | √ | ? | √ | ? | P | √ | P | ? | P |
| Tuples, not attributes, are time-stamped | √ | P | △ | △ | √ | △ | △ | √ | △ |
| Unique representation for each historical relation | △ | △ | △ | △ | △ | △ | √ | √ | △ |

| *Compatible* Criteria |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
| Consistent extension of the snapshot algebra | √ | ? | √ | √ | √ | √ | √ | ? | ? |
| Data periodicity is supported | △ | △ | △ | △ | △ | √ | △ | △ | △ |
| Each collection of valid attribute values is a valid tuple | △ | △ | △ | △ | △ | △ | △ | △ | √ |
| Formal semantics is specified | P | P | √ | △ | △ | √ | √ | P | P |
| Has the expressive power of an historical calculus | ? | ? | √ | ? | ? | ? | √ | P | √ |
| Historical data loss is *not* an operator side-effect | △ | ? | △ | ? | △ | √ | √ | ? | ? |
| Implementation exists | △ | △ | △ | △ | √ | √ | △ | △ | △ |
| Includes aggregates | √ | △ | P | P | P | √ | √ | △ | √ |
| Is, in fact, an algebra | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Model doesn't require *null* attribute values | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Optimization strategies are available | √ | P | P | P | P | P | P | P | P |
| Reduces to the snapshot algebra | √ | △ | √ | ? | √ | P | P | √ | P |
| Supports static attributes | △ | △ | △ | √ | △ | √ | √ | √ | √ |
| Unisorted (not multisorted) | △ | △ | △ | △ | √ | √ | √ | △ | √ |

√ satisfies criterion
P partial compliance

△ criterion not satisfied
? not specified in papers

Table 5: Evaluation of Historical Algebras Against Criteria

*All attributes in a tuple are defined for the same interval(s).* Only Gadia's homogeneous model satisfies this criterion. All the other algebras either time-stamp tuples or allow attribute time-stamps in a tuple to be disjoint.

*Each set of valid tuples is a valid relation.* The algebras proposed by Ben-Zvi, Clifford, Gadia, Jones, Lorentzos, and Tansel all satisfy this criterion. Our algebra fails to satisfy this criterion because it does not allow relations with *value-equivalent* tuples, that is, tuples with the same attribute values. Navathe's algebra also fails to satisfy this criterion. Navathe's algebra requires that tuples with identical values for the explicit attributes be *coalesced*; hence, tuples with identical values for the explicit attributes can neither overlap nor be adjacent in time.

*Restricts relations to first-normal form.* The algebras proposed by Ben-Zvi, Jones, Lorentzos, and Navathe restrict relations to first-normal form. The other algebras all fail to satisfy this criterion as they either allow set-valued attributes or set-valued time-stamps, or both.

*Supports a 3-dimensional view of historical relations and operations.* Our algebra supports the user-oriented conceptual view of an historical relation as a 3-dimensional space-filling solid in that it supports non-homogeneous attribute time-stamping and prevents historical data loss as an operator side-effect. It is unclear whether Gadia's multihomogeneous algebra and Tansel's algebra satisfy this criterion as all operations are not defined formally. The other algebras all fail to satisfy this criterion.

Clifford's algebra fails to satisfy this criterion because difference for historical relations is defined as a standard set difference operation. For example, consider the following single-tuple relations over the scheme *Student* = {*Name*, *Course*} with attribute time-stamping, valid in Clifford's algebra.

A =

| ⟨ *Tuple Value* | | *Tuple Lifespan* ⟩ |
|---|---|---|
| *Name* | *Course* | |
| 2 → Marilyn | 2 → Math | {2, 3, 4} |
| 3 → Marilyn | 3 → Math | |
| 4 → Marilyn | 4 → Math | |

B =

| ⟨ *Tuple Value* | | *Tuple Lifespan* ⟩ |
|---|---|---|
| *Name* | *Course* | |
| 1 → Marilyn | 1 → Math | {1, 2, 3} |
| 2 → Marilyn | 2 → Math | |
| 3 → Marilyn | 3 → Math | |

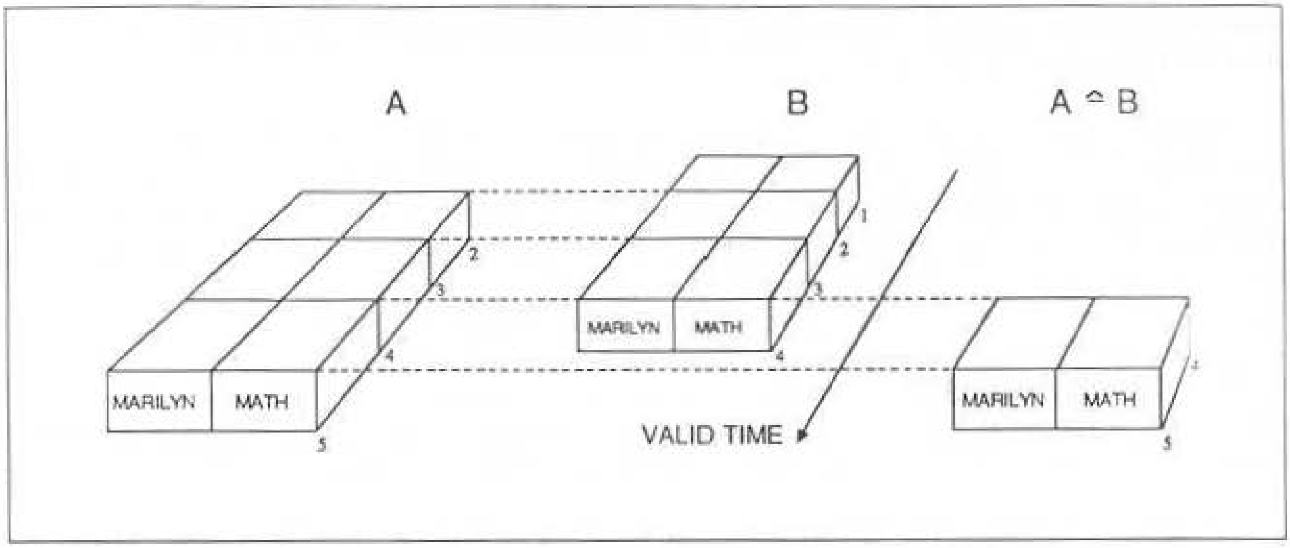

Figure 5: Conceptual View of the Difference Operator Applied to Historical Relations

Intuitively we would expect

$A \hat{-} B =$

| ⟨ *Tuple Value* | | *Tuple Lifespan* ⟩ |
|---|---|---|
| *Name* | *Course* | |
| 4 → Marilyn | 4 → Math | {4} |

consistent with the conceptual view of historical relations as 3-dimensional solids, as shown in Figure 5. However, application of a standard set difference operator yields

$A =$

| ⟨ *Tuple Value* | | *Tuple Lifespan* ⟩ |
|---|---|---|
| *Name* | *Course* | |
| 2 → Marilyn<br>3 → Marilyn<br>4 → Marilyn | 2 → Math<br>3 → Math<br>4 → Math | {2, 3, 4} |

producing nonintuitive results.

Lorentzos' algebra also fails to satisfy this criterion when relations have multiple attribute time-stamps. For example, consider the same single-tuple relations, valid in Lorentzos' algebra.

A =

| *Name* | *N-Start* | *N-Stop* | *Course* | *C-Start* | *C-Stop* |
|---|---|---|---|---|---|
| Marilyn | 2 | 5 | Math | 2 | 5 |

B =

| *Name* | *N-Start* | *N-Stop* | *Course* | *C-Start* | *C-Stop* |
|---|---|---|---|---|---|
| Marilyn | 1 | 4 | Math | 1 | 4 |

In Lorentzos' algebra, historical difference is defined in terms of the Unfold, set difference, and Fold operators. If we unfold both A and B, apply set difference to the unfolded relations, and then fold the result, we would get

A $\hat{-}$ B =

| *Name* | *N-Start* | *N-Stop* | *Course* | *C-Start* | *C-Stop* |
|---|---|---|---|---|---|
| Marilyn | 2 | 3 | Math | 4 | 5 |
| Marilyn | 3 | 4 | Math | 4 | 5 |
| Marilyn | 4 | 5 | Math | 2 | 5 |

Again, the result is inconsistent with the conceptual view of historical relations as 3-dimensional objects, as shown in Figure 5.

The homogeneous model proposed by Gadia and the algebras proposed by Ben-Zvi, Navathe, and Jones also fail to satisfy this criterion. None of these algebras provides a cartesian product operator that allows for the concatenation of two tuples containing arbitrary historical information without the loss of historical information or, in Jones' algebra, the possible implicit addition of historical information. In Gadia's homogeneous model, attributes are time-stamped but the time-stamps of individual attributes are required to be identical. This requirement necessitates the definition of cartesian product using intersection semantics. In Ben-Zvi's algebra, tuples rather than attributes are time-stamped and a *Time Join* operator is defined using intersection semantics. Likewise, in Navathe's algebra, tuples rather than attributes are time-stamped and two operators, *TCJOIN* and *TCNJOIN*, are defined using intersection semantics. Navathe also defines two operators, *TJOIN* and *TNJOIN*, that allow for the concatenation of tuples without loss of historical information. These operators, however, are outside Navathe's algebra; they produce tuples with two time-stamps (R. Ahmed, personal communication, 1987). In Jones' algebra, tuples are time-stamped and cartesian product operators are defined using both intersection and union semantics.

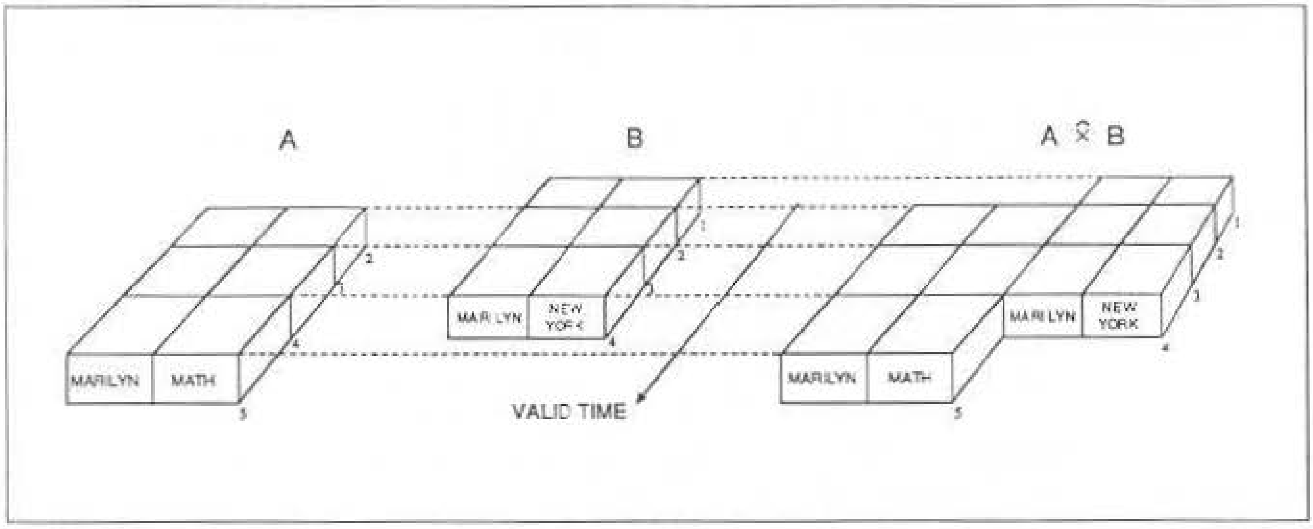

Figure 6: Cartesian Product of Historical Relations

Consider the following single-tuple relations over the schemes $Student = \{Name, Course\}$ and $Home = \{Name, State\}$ with attribute time-stamping.

A =

| *Name* | *Course* |
|---|---|
| ⟨Marilyn, {2, 3, 4}⟩ | ⟨Math, {2, 3, 4}⟩ |

B =

| *Name* | *State* |
|---|---|
| ⟨Marilyn, {1, 2, 3}⟩ | ⟨New York, {1, 2, 3}⟩ |

If cartesian product is represented conceptually as a "volume" operation on cubic solids, we would expect

$A \hat{\times} B =$

| $Name_1$ | *Course* | $Name_2$ | *State* |
|---|---|---|---|
| ⟨Marilyn, {2, 3, 4}⟩ | ⟨Math, {2, 3, 4}⟩ | ⟨Marilyn, {1, 2, 3}⟩ | ⟨New York, {1, 2, 3}⟩ |

as illustrated in Figure 6. However, since Gadia's homogeneous model and the algebras proposed by Ben-Zvi, Navathe, and Jones all define cartesian product using intersection or union semantics, none can support this conceptual view of cartesian product.

*Supports basic algebraic tautologies.* Ben-Zvi's algebra, Gadia's homogeneous model, and Lorentzos' algebra satisfy this criterion. Jones' algebra supports the tautologies, with one exception. The cartesian product operator defined using union semantics fails to support the distributive property of cartesian product over difference. All the tautologies, except the distributive property of cartesian product over difference, also hold for our algebra. Tansel's algebra does not support the commutative property of selection with union and difference. It is unclear whether Tansel's algebra satisfies the other tautologies as union and difference are not defined formally. Similarly, it is unclear whether all the tautologies hold for Clifford's algebra, Gadia's multihomogeneous model, and Navathe's algebra.

*Tuples, not attributes, are time-stamped.* Ben-Zvi, Jones, and Navathe all time-stamp tuples. Clifford also time-stamps tuples, but requires that the partial function from the time domain onto a value domain, representing an attribute's value, be further restricted to the attribute's time-stamp in the relation scheme. The other algebras all time-stamp attributes.

*Unique representation for each historical relation.* Because our algebra allows set-valued time-stamps and disallows value-equivalent tuples, it supports a unique representation for each historical relation. Because Navathe requires that value-equivalent tuples be coalesced, his algebra also supports this criterion. None of the other algebras satisfy this criterion. They all allow multiple representations of identical historical information within a relation.

## 3.2 Compatible Criteria

We how evaluate the algebras against the remaining 14 criteria. Because these criteria are compatible, an historical algebra can be defined that satisfies all these criteria.

*Consistent extension of the snapshot algebra.* Our algebra, along with those proposed by Ben-Zvi, Gadia, Jones, and Lorentzos, satisfy this criterion. Although formal definitions for all operators are not provided for the other algebras, they too are likely to satisfy this criterion.

*Data periodicity is supported.* Only Lorentzos' algebra satisfies this criterion. Lorentzos' algebra allows multiple time-stamps of nested granularity, which can be used to specify periodicity. None of the other algebras allow multiple time-stamps of nested granularity.

*Each collection of valid attribute values is a valid tuple.* Only Tansel's algebra satisfies this criterion. Tansel's algebra time-stamps attributes without imposing any inter-attribute dependence constraints; any collection of valid attribute values is a valid tuple.

The algebras proposed by Ben-Zvi, Jones, and Navathe fail to satisfy this criterion because all three use implicit attributes to specify the end-points of a tuple's time-stamp, implicitly requiring that the value of the start-time attribute be less than (or "$\leq$") the value of the stop-time attribute in all valid tuples. Lorentzos' algebra also requires that the values of attributes representing the boundary points of intervals be ordered. Clifford's algebra does not satisfy this criterion because the value of each attribute in a tuple is defined as a partial function from the time domain onto a value domain, where the function is restricted to times in the intersection of the tuple's time-stamp and the attribute's time-stamp in the relation scheme. Hence, the interval(s) for which an attribute is defined depends on both the tuple's time-stamp and the attribute's time-stamp in the relation scheme. Gadia's homogeneous model fails to satisfy this criterion, as does his multihomogeneous model, except in the degenerative case where each attribute is defined as a separate subscheme. In both models, the requirement that the time-stamp of multiple attributes in a tuple be identical implies that there are sequences of valid attribute values that are not valid tuples. Finally, our algebra fails to satisfy this criterion because it does not allow the time-stamps of all attributes in a tuple to be empty.

*Formal semantics is specified.* We, Gadia, and Lorentzos provide a formal semantics for our algebras. However, Gadia does not provide a formal semantics for operations in his multihomogeneous model; rather, he describes a methodology for converting operations in the homogeneous model to analogous operations in the multihomogeneous model. In the process, necessary details (e.g., how attributes with disjoint time-stamps and empty time-stamps are reconciled with the snapshot semantics) are left unspecified. Likewise, Jones provides no formal semantics for the time-oriented operations in LEGOL; she provides only a brief summary of time-oriented operations available in the language, along with examples illustrating the use of some of these operations.

Ben-Zvi, Clifford, and Tansel provide formal semantics for their algebras but all provide incomplete definitions for certain operators. For example, Ben-Zvi's definition of the difference operator does not include a definition of the *Effective-Time-Start* and *Effective-Time-End* of tuples in the resulting relation. Clifford's definition of the cartesian production operator does not include a definition of the lifespan of tuples in the resulting relation. Similarly, Clifford's definition of the union operator does not include a definition of the lifespan of an attribute, at the scheme level, in the resulting relation. Finally, Tansel does not provide formal definitions for his historical union and difference operators.

Navathe provides formal semantics for three new historical selection and four new historical join operators. He retains the five basic snapshot operators, although his model requires that value-equivalent tuples be coalesced. The semantics of these operators are left unspecified.

*Has the expressive power of an historical calculus.* Gadia has defined an equivalent calculus for his homogeneous model and we have shown that our algebra has the expressive power of the TQuel calculus. Likewise, Tansel has defined an equivalent calculus for his algebra [Tansel & Arkun 1985]. Ben-Zvi has augmented the SQL Query Language with a *Time-View* operator and shown that the resulting language has expressive power equivalent to that of his algebra [Ben-Zvi 1982]. Rather than modify the semantics of the SQL Query Language to handle the temporal dimension, Ben-Zvi uses the *Time-View* operator as a temporal preprocessor to construct snapshot relations that can then be manipulated the same as any other snapshot relations. Navathe has defined the temporal query language TSQL [Navathe & Ahmed 1986], which is a superset of SQL, for use in his model. He has not shown, however, that his algebra has the expressive power of TSQL. A calculus has yet to be defined for any of the other proposed algebras.

*Historical data loss is* not *an operator side-effect.* Historical data loss is not an operator side-effect in our algebra. All operators are defined to retain, in their resulting relations, the historical information found in their underlying relations, unless removal is specifically required by the operator. Historical data loss is also not an operator side-effect in Lorentzos' algebra; all historical information is embedded in snapshot relations and all operations are defined in terms of the basic snapshot operators. Ben-Zvi's algebra, Gadia's homogeneous model, and Jones' algebra all fail to satisfy this criterion because each time-stamps tuples and defines a cartesian product operator using intersection semantics. It is unclear whether the other algebras satisfy this criterion, as formal definitions for all operators are not provided.

*Implementation exists.* A prototype version of the algebra proposed by Jones has been implemented on the Peterlee Relational Test Vehicle [Jones et al. 1979]. Also, a prototype version of the algebra proposed by Lorentzos has been implemented on a PDP-11/44 as an extension of INGRES [Lorentzos & Johnson 1987B]. To the best of our knowledge, implementations do not exist for the other algebras.

*Includes aggregates.* We along with Ben-Zvi define historical aggregate operators formally as part of our algebras. Tansel also defines historical aggregate functions in his algebra in terms of a new operator, termed *enumeration*, and an aggregate formulation operator [Tansel 1987]. Aggregate functions, defined for the snapshot algebra, can be used to compute historical aggregates in Lorentzos' algebra. The algebra proposed by Jones includes aggregate operators, but these operators are not defined formally. Although Gadia does not include aggregates in his models, he does introduce "temporal navigation" operators (e.g., First), which act similarly to the TQuel temporally oriented aggregates. The other algebras do not include any aggregate operators.

*Is, in fact, an algebra.* Each of the nine algebras being evaluated satisfies this criterion.

*Model doesn't require null attribute values.* All nine algebras being evaluated satisfy this criterion.

*Optimization strategies are available.* Ben-Zvi describes an efficient implementation of his algebra, while Gadia presents a computational semantics, designed to aid efficient implementation of the algebra, for his homogeneous model. Also, optimization techniques based on the algebraic tautologies, with certain exceptions for some algebras, could be used in an implementation of any of the nine algebras.

*Reduces to the snapshot algebra.* Gadia's homogeneous model satisfies this criterion; operators are defined using a snapshot semantics thus guaranteeing that the algebra reduces to the snapshot algebra. Likewise, the descriptions of the algebras proposed by Ben-Zvi and Jones imply that the operators are defined using snapshot semantics. Because tuple time-stamping is assumed in Navathe's model, his algebra also satisfies this criterion. Although formal definitions have not been provided for operators in Gadia's multihomogeneous model, the algebra is likely to satisfy this criterion only through the introduction of distinguished *null's* when taking snapshots. Because we, along with Tansel and Lorentzos, allow non-homogeneous attribute time-stamps, our algebras also satisfy this criterion only through the introduction of distinguished *null's* when taking snapshots.

The algebra proposed by Clifford fails to satisfy this criterion. Consider the following single-tuple relations over the scheme $Student = \{Name, Course\}$ valid in Clifford's algebra.

A =

| ⟨ *Tuple Value* | | *Tuple Lifespan* ⟩ |
|---|---|---|
| *Name* | *Course* | |
| 3 → Phil | 3 → English | {3} |

B =

| ⟨ *Tuple Value* | | *Tuple Lifespan* ⟩ |
|---|---|---|
| *Name* | *Course* | |
| 2 → Phil | 2 → English | {2, 3} |
| 3 → Phil | 3 → English | |

Clifford defines difference as a standard set difference operation. Therefore, in Clifford's algebra,

$A \hat{-} B$ =

| ⟨ *Tuple Value* | | *Tuple Lifespan* ⟩ |
|---|---|---|
| *Name* | *Course* | |
| 3 → Phil | 3 → English | {3} |

If we were to take a snapshot of $A \hat{-} B$ at time 3, we would obtain the relation { (Phil, English) }. However, if we were to take a snapshot of relation $A$ at time 3 and a snapshot of relation $B$ at time 3 and take the difference of these snapshot relations, we would obtain the empty relation.

*Support static attributes.* Lorentzos', Navathe's, and Tansel's algebras satisfy this criterion by allowing both time-dependent and non-time-dependent attributes. Our algebra and Gadia's multihomogeneous model also can support static attributes. In these two algebras, the time-stamp of an attribute can be defined independently of the time-stamps of any of the other attributes in a tuple. In our algebra we would represent a static attribute as an attribute assigned the time domain. The other five algebras all require that the same valid time be associated with all attributes in a tuple; therefore, none of these algebras can support static and time-dependent attributes within the same tuple.

*Unisorted (not multisorted).* Our algebra, along with those proposed by Jones, Lorentzos, and Tansel is unisorted in that it concerns only one object type. Gadia defines a multisorted algebra whose object types are historical relations and temporal expressions. Clifford defines a multisorted algebra whose object types are historical relations and lifespans. Both Ben-Zvi and Navathe allow snapshot and historical relations.

## 4 Summary

In this paper, we have evaluated nine historical algebras against 21 criteria. We first described the algebras in terms of the types of objects they define and the operations on object instances they allow. Then, we introduced evaluation criteria for historical algebras, each of which is well-defined, has an objective basis for being evaluated, and is arguably beneficial. We omited properties from the list of criteria that were either subsumed by criteria, not well-defined, or had no objective basis for being evaluated. We also identified incompatibilities among the criteria. Finally, we evaluated the algebras against the criteria.

Supports a 3-dimensional view of historical relations and operations?

Unique representation for each historical relation?

| | No | Yes |
|---|---|---|
| No | Ben-Zvi<br>Clifford & Croker<br>Gadia H<br>Jones, et al.<br>Lorentzos & Johnson | Gadia MH<br>Tansel |
| Yes | Navathe & Ahmed | McKenzie & Snodgrass |

Table 6: Classification of Algebras According to Criteria Satisfied

Of the 21 criteria listed in Table 3, each is satisfied by at least one of the nine algebras and three are satisfied, at least partially, by all the algebras. As was shown in Section 2, the subset of conflicting criteria that an algebra satisfies is necessarily dependent on whether the algebra supports a 3-dimensional view of historical relations and operations and whether each historical relation in the algebra has a unique representation. For example, we and Navathe cannot satisfy the criterion that each set of valid tuples is a valid relation because our algebras satisfy the criterion that each historical relation has a unique representation. In Table 6 all nine algebras are classified according to their satisfaction of these two criteria. According to this classification and the summary of incompatibilities among criteria in Table 4, Navathe's algebra cannot satisfy one of the remaining conflicting criteria, Gadia's multihomogeneous model and Tansel's algebra cannot satisfy three of the remaining criteria, while our algebra cannot satisfy any of the remaining conflicting criteria. The other algebras are not restricted from satisfying the remaining conflicting criteria. There is no apriori reason any of the compatible criteria cannot be satisfied; one measure of the quality of the design of an algebra is the extent to which it satisfies these criteria.

As no algebra can satisfy all the criteria, an obvious future effort would identify a maximal subset of the criteria, requiring a ranking of the criteria by importance. Such a ranking is necessary to determine which quadrant of Table 4 is objectively superior. Also, the list of criteria presented here is not meant to be exhaustive; other properties of historical algebras likely merit consideration as evaluation criteria. Hence, one aspect of future work is the identification of other desirable criteria of historical algebras.

# 5 Acknowledgements

We would like to thank James Clifford, Nikos Lorentzos, and Abdullah Uz Tansel for their critique and suggested improvements to an earlier version of our list of criteria for evaluating historical algebras.

Research by the first author was sponsored in part by the United States Air Force. Research by the second author was sponsored in part by an IBM Faculty Development Award. The work was also supported by NSF grant DCR-8402339 and ONR grant N00014-86-K-0680.

# 6 Bibliography

[Allen & Hayes 1985] Allen, J.F. and P.J. Hayes. *A Common-Sense Theory of Time*, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Los Angeles, CA: Aug. 1985, pp. 528-531.

[Anderson 1982] Anderson, T.L. *Modeling Time at the Conceptual Level*, in *Proceedings of the International Conference on Databases: Improving Usability and Responsiveness*, Ed. P. Scheuermann. Jerusalem, Israel: Academic Press, June 1982, pp. 273-297.

[Ariav 1986] Ariav, G. *A Temporally Oriented Data Model. ACM Transactions on Database Systems*, 11, No. 4, Dec. 1986, pp. 499-527.

[Ariav & Clifford 1986] Ariav, G. and J. Clifford. *Temporal Data Management: Models and Systems*, in New Directions for Database Systems. Norwood, New Jersey: Ablex Publishing Corporation, 1986. Chap. 12. pp. 168-185.

[Ben-Zvi 1982] Ben-Zvi, J. *The Time Relational Model*. PhD. Diss. Computer Science Department, UCLA, 1982.

[Bubenko 1977] Bubenko, J.A., Jr. *The Temporal Dimension in Information Modeling*, in Architecture and Models in Data Base Management Systems. The Netherlands: North-Holland Pub. Co., 1977.

[Clifford & Tansel 1985] Clifford, J. and A.U. Tansel. *On an Algebra for Historical Relational Databases: Two Views*, in *Proceedings of ACM SIGMOD International Conference on Management of Data*, Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 247-265.

[Clifford & Croker 1987] Clifford, J. and A. Croker. *The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans*, in *Proceedings of the International Conference on Data Engineering*, IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1987, pp. 528-537.

[Codd 1970] Codd, E.F. *A Relational Model of Data for Large Shared Data Bank. Communications* *of the Association of Computing Machinery,* 13, No. 6, June 1970, pp. 377-387.

[Dadam et al. 1984] Dadam, P., V. Lum and H.-D. Werner. *Integration of Time Versions into a Relational Database System*, in *Proceedings of the Conference on Very Large Databases,* Ed. U. Dayal, G. Schlageter and L.H. Seng. Singapore: 1984, pp. 509-522.

[Date 1986] Date, C.J. *An Informal Definition of the Relational Model*, in Relational Database: Selected Writings. Reading, MA: Addison-Wesley, 1986. Chap. 2. pp. 21-31.

[Enderton 1977] Enderton, H.B. *Elements of Set Theory.* New York, N.Y.: Academic Press, Inc., 1977.

[Gadia 1986] Gadia, S.K. *Toward a Multihomogeneous Model for a Temporal Database*, in *Proceedings of the International Conference on Data Engineering,* IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1986, pp. 390-397.

[Jones et al. 1979] Jones, S., P. Mason and R. Stamper. *LEGOL 2.0: A Relational Specification Language for Complex Rules. Information Systems,* 4, No. 4, Nov. 1979, pp. 293-305.

[Klopprogge 1981] Klopprogge, M.R. *TERM: An approach to include the time dimension in the entity-relationship model*, in *Proceedings of the Second International Conference on the Entity Relationship Approach,* Oct. 1981.

[Lorentzos & Johnson 1987A] Lorentzos, N.A. and R.G. Johnson. *TRA: A Model for a Temporal Relational Algebra*, in *Proceedings of the Conference on Temporal Aspects in Information Systems,* AFCET. France: May 1987, pp. 99-113.

[Lorentzos & Johnson 1987B] Lorentzos, N.A. and R.G. Johnson. *Extending Relational Algebra to Manipulate Temporal Data.* Internal Report NL/1/87. Department of Computer Science, Birkbeck College, London University. Aug. 1987.

[Lum et al. 1984] Lum, V., P. Dadam, R. Erbe, J. Guenauer, P. Pistor, G. Walch, H. Werner and J. Woodfill. *Designing DBMS Support for the Temporal Dimension*, in *Proceedings of ACM SIGMOD International Conference on Management of Data,* Ed. B Yormark. Association for Computing Machinery. Boston, MA: June 1984, pp. 115-130.

[Maier 1983] Maier, D. *The Theory of Relational Databases.* Rockville, MD: Computer Science Press, 1983.

[Manola & Dayal 1986] Manola, F. and U. Dayal. *PDM: An Object-Oriented Data Model*, in *Proceedings of the International Workshop on Object-Oriented Database Systems,* 1986.

[McKenzie 1986] McKenzie, E. *Bibliography: Temporal Databases. ACM SIGMOD Record,* 15, No. 4, Dec. 1986, pp. 40-52.

[McKenzie & Snodgrass 1987A] McKenzie, E. and R. Snodgrass. *Extending the Relational Algebra to Support Transaction Time*, in *Proceedings of ACM SIGMOD International Conference on Management of Data,* Ed. U. Dayal and I. Traiger. Association for Computing Machinery.

San Francisco, CA: May 1987, pp. 467-478.

[McKenzie & Snodgrass 1987B] McKenzie, E. and R. Snodgrass. *Scheme Evolution and the Relational Algebra.* Technical Report TR87-003. Computer Science Department, University of North Carolina at Chapel Hill. May 1987.

[McKenzie & Snodgrass 1987C] McKenzie, E. and R. Snodgrass. *Supporting Valid Time: An Historical Algebra.* Technical Report TR87-008. Computer Science Department, University of North Carolina at Chapel Hill. Aug. 1987.

[Navathe & Ahmed 1986] Navathe, S.B. and R. Ahmed. *A Temporal Relational Model and a Query Language.* UF-CIS Technical Report TR-85-16. Computer and Information Sciences Department, University of Florida. Apr. 1986.

[Snodgrass & Ahn 1985] Snodgrass, R. and I. Ahn. *A Taxonomy of Time in Databases*, in *Proceedings of ACM SIGMOD International Conference on Management of Data,* Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 236-246.

[Snodgrass & Ahn 1986] Snodgrass, R. and I. Ahn. *Temporal Databases. IEEE Computer,* 19, No. 9, Sep. 1986, pp. 35-42.

[Snodgrass 1987] Snodgrass, R. *The Temporal Query Language TQuel. ACM Transactions on Database Systems,* 12, No. 2, June 1987, pp. 247-298.

[Tansel & Arkun 1985] Tansel, A.U. and M.E. Arkun. *Equivalence of Historical Relational Calculus and Historical Relational Algebra.* Technical Report. Bernard M. Baruch College, CUNY. 1985.

[Tansel 1986] Tansel, A.U. *Adding Time Dimension to Relational Model and Extending Relational Algebra. Information Systems,* 11, No. 4 (1986), pp. 343-355.

[Tansel 1987] Tansel, A.U. *A Statistical Interface for Historical Relational Databases,* in *Proceedings of the International Conference on Data Engineering,* IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1987, pp. 538-546.

[Ullman 1982] Ullman, J.D. *Principles of Database Systems, Second Edition.* Potomac, Maryland: Computer Science Press, 1982.