Evaluation of Relational Algebras Incorporating the Time Dimension in Databases

L. EDWIN MCKENZIE, JR.

AFCAC/SY Bldg 1320F Hanscom AFB, Massachusetts 01731

RICHARD T. SNODGRASS

Department of Computer Science, University of Arizona, Tucson, Arizona 85721

The relational algebra is a procedural query language for relational databases. In this paper we survey extensions of the relational algebra that can query databases recording time-varying data. Such an algebra is a critical part of a temporal DBMS. We identify 26 criteria that provide an objective basis for evaluating temporal algebras. Seven of the criteria are shown to be mutually unsatisfiable, implying there can be no perfect temporal algebra. Choices made as to which of the incompatible criteria are satisfied characterize existing algebras Twelve time-oriented algebras are summarized and then evaluated against the criteria. We demonstrate that the design space has in some sense been explored in that all combinations of basic design decisions have at least one representative algebra. Coverage of the remaining criteria provides one measure of the quality of each algebra We argue that all of the criteria are independent and that the criteria identified as compatible are indeed so. Finally, we list plausible properties proposed by others that are either subsumed by other criteria, are not well defined, or have no objective basis for being evaluated. The algebras realize many different approaches to what appears initially to be a straightforward design task.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design—*data models, normal forms, schema and subschema*; H.2.3 [**Database Management**]: Languages—*data manipulation languages* (*DML*); H.2.4 [**Database Management**]: Systems—*query processing, transaction processing*; H.4 1 [**Information Systems Applications**]: Office Automation

General Terms: Languages, Performance

Additional Key Words and Phrases: Aggregate, chronon, historical relation, homogeneity, query optimization, snapshot relation, transaction time, valid time

INTRODUCTION

Time is an important aspect 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, such as econometrics, banking, inventory control, accounting, law, medical records, cartographic data, and airline reserva-

CONTENTS

tions. Yet the relational data model [Codd 1970, 1990] does not support 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, generally now. 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 nevertheless ad hoc and application specific.

The need for application-independent database management system (DBMS) support for time-varying information is receiving increasing attention, primarily for two reasons. First, as the research and business communities became comfortable with relational database theory and practice, a plethora of new applications for databases emerged, among them temporal applications. Second, the advent of optical storage technology provided an answer to the question of where to store all the historical data. In the last decade alone, more than 320 articles relating time to information processing have been published [McKenzie 1986; Soo 1991; Stam and Snodgrass 1988].

One area of continuing research interest is the development of a *temporal data model* capable of representing the temporal dimension of real-world phenomena. Such a model would store, along with information on entities and relationships, both when that information was valid in the real world being modeled and when that information was recorded in the database. The primary focus has been extending the relational data model to support time-varying information. Included in that effort is the extension of the relational algebra.

An *algebra* is a set of objects and a collection of operations over those objects. In the relational algebra, the objects are *relations*, and the operations take as arguments one or two relations and produce a relation. An example is the *selection* operator that yields a relation whose components, supplied by the underlying relations, satisfy a specified predicate.

The relational algebra was originally defined as an operational equivalent of a calculus-based declarative relational query language such as SQL. The DBMS translates SQL queries into an algebraic expression, which is then optimized and executed. Implementation issues, such as query optimization and physical storage strategies, can best be addressed in terms of the algebra. In fact, one of the reasons for the success of the relational model is that it lends itself to an algebraic execution paradigm [Elmasri and Navathe 1989; Vandenberg and DeWitt 1991].

Example

Let R be a relation specifying the courses each student is taking (see Figure 1). A simple query on this relation, "what courses are offered," is expressed in SQL as follows:

```
SELECT course
FROM R
```

This may be translated into the algebraic *projection* operation, $\pi_{\text{course}}(R)$. The

R =

sname	course
"Phil"	"English"
"Norman"	"English"
"Norman"	"Math"

Figure 1. Snapshot relation example.

course
"English"
"Math"

Figure 2. Result of a snapshot query.

resulting relation is shown in Figure 2.

In Figure 3 we record time-varying information by augmenting each row with the starting and stopping semesters. 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 fall semester 1980, 2 represents the spring semester 1981, and so on. Note that two tuples are needed to record Phil's enrollment in English, as his enrollment was not continuous. We can now query "when was each course taken," with the result shown in Figure 4. If Norman had *not* taken English during semesters 1 and 2, the result relation would contain two tuples with a course of English, as in Figure 3. □

Such a query on a time-varying relation cannot be translated into the relational algebra. Time-oriented aspects, such as the fact that consecutive but otherwise identical intervals should be coalesced into a single interval, as in the example for "English," are not provided for in the relational algebra. The alternative, *not* coalescing consecutive intervals, is undesirable because it allows multiple, equivalent representations for each temporal relation, which is not faithful to the conventional relational model and can complicate the semantics of the operations or require inefficient implementations. Hence, even this very simple query raises difficulties with the relational algebra. Also, new operations, such as the ability to apply a temporal predicate, such as *overlap*, or to compute a time-varying aggregate, such as *count*, are required. What is needed is a temporal algebra that incorporates these aspects and operations. An extended algebra can serve as an appropriate target for a temporal query language translator, an appropriate structure on which to perform optimization, and an appropriate executable formalism for the DBMS to interpret to execute queries. A temporal algebra is thus a critical part of a DBMS if it is to support time-varying information.

Desirable characteristics for a temporal algebra derive from the uses just listed and include being a consistent extension of the relational algebra, supporting time as an additional dimension, supporting existing optimization strategies, and having the expressive power of a temporal calculus-based query language. These and many other characteristics will be explored below.

In this paper we examine algebras that are extensions of the conventional relational algebra and that support manipulation of the temporal dimension of real-world phenomena. In the next section, we first review the conventional relational algebra, then briefly describe the temporal algebras that have been proposed, emphasizing the types of objects that each defines and the operations on objects that each provides. Some algebras use as objects snapshot relations with additional time-stamp attributes, with the time notion appearing primarily in new algebraic operators and/or new semantics of the existing operators. Other algebras are defined on objects that less closely resemble conventional relations.

To evaluate these algebras, we identify 26 criteria, each of which is well defined, has an objective basis, and is arguably beneficial. We separate these criteria into 7 *conflicting* and 19 *compatible* criteria. We show that no algebra can simultaneously satisfy all conflicting

R =

sname	course	start	stop
"Phil"	"English"	1	1
"Phil"	"English"	3	4
"Norman"	"English"	1	2
"Norman"	"Math"	5	6

Figure 3. Relation with time example.

course	start	stop
"English"	1	4
"Math"	5	6

Figure 4. Result of a temporal query

criteria. Hence, there can be no perfect algebra, which is perhaps why so many temporal algebras have been defined. We argue that all algebras should be expected to satisfy the compatible criteria. The conflicting criteria give rise to four major "approaches," each identifying the (generally implicit) mindset of the algebra's designer(s) as to which of the conflicting criteria are most important to satisfy. We show that this design space has in some sense been explored, in that each approach has at least 1 (in actuality, at least 2) representative algebras. We also touch on 10 plausible properties that have been previously proposed that we do not include as criteria because they have no objective basis for being evaluated or are subsumed by other criteria. Hence, we provide a comprehensive means of evaluating a temporal algebra: First, determine to which approach it adheres, and, second, determine which compatible criteria it satisfies. We evaluate 12 algebras using this strategy.

In the next section we first examine the kinds of time the algebras support and the conceptual model of time employed. We summarize the temporal algebras, considering the objects each defines and the operations on those objects that each provides. We then explain the criteria and show how seven are mutually incompatible. We evaluate the algebras on the conflicting and compatible criteria. Finally, we argue informally and empirically that the criteria are independent; specifically, that they are not pairwise logically equivalent and that the criteria termed compatible are not pairwise incompatible.

Those interested in databases in general will find Sections 1.1 (aspects of temporal modeling), 1.3 (summary of temporal algebras), and 4 (summary) and Table 3 and Figure 13 most relevant. Those interested in temporal databases, whether concerning modeling and logical design, query optimization, or implementation should read Sections 2 (criteria) and 3 (evaluation). Those interested in a specific temporal algebra should at least peruse Section 2.1 and should read Sections 3.1 (conflicting criteria) and 3.2 (compatible criteria). We recommend that those designing a new temporal algebra study the entire paper carefully to understand the many subtle issues at play.

There are three contributions of this paper, in addition to the survey itself: providing an objective basis for evaluating a temporal algebra, demonstrating that an algebra that satisfies all desirable criteria cannot be defined, and showing that the major areas of the design space (termed "approaches" above) have all been explored.

1. TEMPORAL ALGEBRAS

In Codd's [1970] relational algebra, the only type of object is the *relation*. Assume that we are given a set $\mathscr{A} = \{A_1, \ldots, A_n\}$, where each A_i, $1 \le i \le n$, is called an *attribute name*, or simply an *attribute*. Also assume that there is an arbitrary, nonempty, finite, or denumerable set, $\mathscr{D}_i$, $1 \leq i \leq n$, called a *domain* corresponding to each 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), $D_i \in \mathscr{D}_i$, $1 \leq i \leq n$ [Date 1986]. Hence, each element of a tuple maps an attribute name onto a value in its associated domain. The set of attributes $\mathscr{A}$ for a relation is called the *relation schema*. Because relations are sets, tuples with duplicate attribute values are not permitted. 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 we are given the relation schema *Student* = {**sname, course**} and corresponding domains of student names and college courses. Then R is a relation on the schema *Student*, as shown in Figure 1. □

There are five basic operations in the relational algebra: *union*, *set difference*, *Cartesian product*, *selection*, *and projection* [Ullman 1988]. 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. Union and difference require argument relations over the same schema. The Cartesian product ($\times$) of a relation of n-tuples and a relation of m-tuples is the relation containing $(n + m)$-tuples that have n elements from a tuple in the first input relation and m elements from a tuple in the second input relation. We assume, without loss of generality, that the relation schemas of the input relations are disjoint [Maier 1983]. The last two are unary operations. Selection (σ) maps an input relation to 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 (π) maps each tuple in its input relation to 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.

1.1 Aspects of Temporal Modeling

Before discussing specific temporal 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 concern. There are three orthogonal kinds of time that a data model may support: valid time, transaction time, and user-defined time [Snodgrass and Ahn 1985, 1986]. *Valid time* concerns modeling a 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 a similar meaning 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 (perhaps represented as an integer) identifies the transaction that stored the information about the event in the database. Other terms found in the literature that have a similar meaning are 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.

Valid time, unlike transaction time, is a multifaceted aspect of time. Different times may be used in defining the existence of a single object or relationship; for example, the time a student completes all degree requirements and the time of the student's graduation ceremony may both be used in specifying the student's graduation from college. Also, the various properties of an entity or relationship need not change at the same time; for example, an employee's promotion may, but need not, be accompanied by a change in salary or office address.

The relational algebra already supports user-defined time because it is simply another domain, such as integer or character string [Overmyer and Stonebraker 1982]. The relational algebra, however, supports neither valid time nor transaction time. 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. The application programmer can, however, encode time-varying information in a conventional database by using attributes with the domain of user-defined time, but the DBMS does not interpret such attributes differently from attributes of other domains, as illustrated with the SQL query given previously. Hence, we refer to the relational algebra hereafter as the *snapshot* algebra, since it captures only a single snapshot in time of both a relation and the enterprise that the relation models.

An algebra, data model, or relation supporting only valid time is termed *historical*; one that supports only transaction time is termed *rollback*; and one that supports both valid and transaction time is termed *temporal* [Snodgrass and Ahn 1986]. Most of the algebras and their underlying data models surveyed here are historical. Transaction time was introduced later than valid time and is easier to support in an algebra. Most of the hard problems are encountered in supporting valid time, so that is the aspect addressed in most algebras.

Figure 5 illustrates a *single* temporal relation composed of a sequence of historical states indexed by transaction time. It is the result of four transactions starting from a null relation: (1) three tuples were added, (2) one tuple was added, (3) one tuple was added and an existing one terminated, and (4) the starting time of a previous tuple [the middle one added in transaction (1)] was changed to a somewhat later time (presumably the original starting time was incorrect) and a recently added tuple (the bottom one) was deleted (presumably it should not have been there in the first place.) Each update operation involves copying the historical state, then applying the update to the newly created state. Of course, less redundant representations than the one shown are possible.

Example

Assume there exists a temporal relation, with the explicit attributes **sname** and **course**, recording the courses each student has taken. One possible temporal query on this relation is, "What courses was Phil enrolled in at the start of the fall 1991 semester, as known on July 1, 1991?" This query, relevant if preregistration is allowed, specifies a rollback to the historical state current at a transaction time of July 1, 1991, then an historical selection on the valid time of fall 1991. □

In addition to the kinds of time supported, we must also consider the conceptual model of time employed. Two time models have been proposed: 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 and Tansel 1985]. In the continuous model, each real number corresponds to a "point" in time; in the discrete model, each natural number corresponds to a nondecomposable unit of time having an arbitrary duration. Although the two time models represent time differently, they share one important property: They both require that time be ordered linearly. Hence, for two

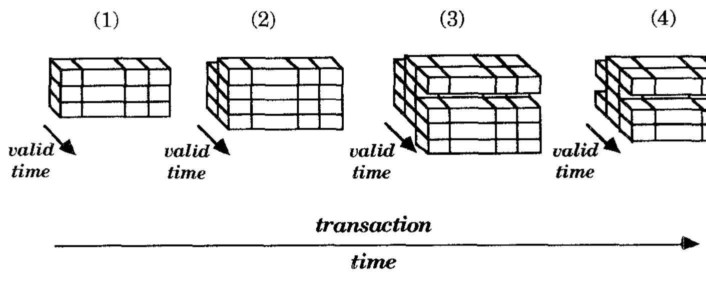

Figure 5. A temporal relation.

nonequal times, t_1 and t_2, either t_1 is before t_2 or t_2 is before t_1 [Anderson 1982; Clifford and Tansel 1985].

"Instant" [Gadia 1986], "moment" [Allen and Hayes 1985], "time quantum" [Anderson 1982], and "time unit" [Navathe and Ahmed 1987; Tansel 1986] are terms used in the literature to describe a nondecomposable unit of time in the discrete model. To avoid confusion between a point in the continuous model and a nondecomposable unit of time in the discrete model, we refer to a nondecomposable unit of time in the discrete model as a *chronon* [Ariav 1986; Clifford and Rao 1987] 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., semester, hour, second). A chronon typically is denoted by an integer, corresponding to a single granularity, but it may also be denoted by 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 1901 denotes March 15, 1985. If, we assume a nested granularity of ⟨year, month, day⟩, then the sequence ⟨6, 3, 15⟩ denotes March 15, 1985.

Although time itself is generally perceived to be 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 the preference of the discrete model over the continuous model. First, measures of time are inherently imprecise [Anderson 1982; Clifford and 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. Second, 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 do not mean that the event occurred at the "point" in time associated with 4:30 p.m., but at some time in the chronon (perhaps minute) associated with 4:30 p.m. [Anderson 1982; Clifford and Rao 1987]. Third, the concepts of chronon and interval allow us to naturally model 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 temporal algebras surveyed in this paper are compatible with the discrete time model.

Four basic design decisions characterize the types of objects a temporal algebra defines. These decisions arise because both valid and transaction time must be recorded in some way in relations. There is more latitude in representing valid time because it is independent of data storage.

- Is valid time associated with tuples (usually as additional implicit or explicit attributes) or with the attributes' values?

- How is valid time represented? Timestamps, which represent valid time, may be either chronons, intervals, or sets of intervals, where a set of intervals is denoted with a set of chronons, not all of which are consecutive.
- Are attribute values required to be atomic valued, or are they allowed to be set valued? Although first normal form dictates that attributes be atomic valued in a snapshot relation, attribute values that are atomic valued at all points in time may nonetheless be set valued when their behavior over time is recorded.
- Is transaction time associated with attribute values, tuples, or sets of tuples?

Two basic design decisions characterize the different kinds of operations each algebra provides. The first design decision represents a choice of one of only two possible alternatives; the second concerns the four basic operations concerning time that should be supported by a temporal algebra.

- Does the algebra retain the set-theoretic semantics of the five basic relational operators and introduce new operators to deal with time, or does the algebra extend the semantics of the existing relational operators to account for the temporal dimension directly?
- How does the algebra handle *temporal selection* (i.e., tuple selection based on valid time-stamps), *rollback* (i.e., tuple selection based on transaction time-stamps), *temporal projection* (i.e., computation of a new time-stamp for a tuple or attribute from its current time-stamp), and *temporal aggregation* (i.e., computation of a distribution of aggregate values over time)—operations that are unique to a temporal algebra?

1.2 Overview of Algebras

We now review briefly 12 algebras that extend the snapshot algebra to support valid time and in some cases also transaction time. These algebras differ in the types of objects they define and in the kinds of operations they provide. Because some of the algebras have similar names and others are unnamed, we identify each algebra with the name of the first author of the paper in which the algebra is presented, except for the heterogeneous algebra discussed in Gadia and Yeung [1988] and Yeung [1986], which we will identify as Yeung's algebra to avoid confusing it with Gadia's [1988] homogeneous algebra. We did not include TERM [Klopprogge 1981], PDM [Manola and Dayal 1986], or the accounting data model (ADM) algebra [Thompson 1991], all of which include support for time, in this evaluation since they are temporal extensions of other data models. TERM is an extension of the entity-relationship model [Chen 1976]; PDM is an extension of the entity-oriented Daplex functional data model [Shipman 1981]; and the ADM is an extension of the relational model incorporating sets of synchronized time-varying relations. Also, in passing we will mention three variants of the algebras we do study.

A detailed examination and justification of all of the decisions made with each algebra is outside the scope of the paper. Rather, this section introduces each algebra as a prelude to a detailed comparison, where certain aspects of each algebra that relate to the specific criteria are discussed in detail.

We first examine the models that time-stamp tuples, then discuss those that time-stamp attribute values, and finish with Tuzhilin's algebras, which are largely independent of data model. We proceed chronologically.

1.2.1 Jones

LEGOL 2.0 [Jones et al. 1979] is a language designed to be used in database applications, such as legislative rules writing and high-level system specification, in which the temporal ordering of events and the valid times for objects are important. It was the first time-oriented algebra defined; it introduced many of the features found in later algebras.

Objects in the LEGOL 2.0 data model are relations 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 (inclusive) end points of the interval of existence (i.e., valid time) of the real-world entity or relationship represented by a tuple; these values are specified during data entry by the user.

Example

R is a historical relation in LEGOL 2.0 over the explicit attributes {**sname, course**}, shown in Figure 3. Later examples in this section will show the semantically equivalent representation of *R* in the other algebras. Because the algebras all define relations differently and, in some cases, require implicit attributes, we show all relation examples in tabular form for both clarity and consistency of notation. This relation shows that Phil was a student in the English course for the fall 1980 semester and for the fall and spring 1981 semesters. □

LEGOL 2.0 retains the standard set-theoretic operations and introduces several time-related operations to handle the temporal dimension of data. Operations in LEGOL 2.0 are not defined formally; the more important operations are described using examples. 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 are 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).

1.2.2 Ben-Zvi

The Time Relational Model [Ben-Zvi 1982] supports both valid time and transaction time; it was the first to do so. Two types of objects are defined: snapshot relations, as defined in the snapshot algebra, and temporal relations. Temporal relations are sets of tuples, with each tuple having five implicit time attributes. The attributes **effective-time-start** and **effective-time-stop** are the end points of the interval of existence of the real-world phenomenon being modeled; **registration-time-start** is the transaction time of the transaction that stored the **effective-time-start**; **registration-time-stop** is the transaction time that stored the **effective-time-stop**; and **deletion-time** records the time when erroneously entered tuples are logically deleted. An erroneous attribute value may be corrected by deleting that tuple and inserting a corrected one.

Example

R is a temporal relation in the Time Relational Model over the explicit attributes **{sname, course}** containing four tuples.

$R =$

sname	course	effective time-start	effective time-stop	registration time-start	registration time-stop	deletion time
"Phil"	"English"	1	1	423	427	—
"Phil"	"English"	3	4	444	452	—
"Norman"	"English"	1	2	423	438	—
"Norman"	"Math"	5	6	469	487	—

Note that none of the tuples has been (logically) deleted. □

A new *Time-View* operator, TV = (t_e, t_s) that maps a temporal relation instance onto a snapshot relation instance is introduced. The Time-View operator can be thought of as a limited form of temporal selection that selects from the relation those tuples with a valid time containing t_e and a transaction time containing t_s. 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, 425), then

TV(R) =	sname	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 time and the 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 instance at valid time t_e having a transaction time of t_s; 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 union or intersection semantics. For example, the union operator is defined using union semantics, and the join operator is defined using intersection semantics. These two operators cannot be simulated by the conventional relational operators. What would be needed is a projection operator that allowed expressions using the two simple functions *first*(a, b) (= if $a > b$ then b else a) and *last*(a, b). The valid time of tuples resulting from the difference operator is left unspecified.

1.2.3 Navathe

The Temporal Relational Model [Navathe and Ahmed 1987] and its associated algebra were defined primarily to support TSQL [Navathe and Ahmed 1989], a time-oriented extension to SQL defined in the same paper. The model allows both non-time-varying and time-varying attributes, but all of a relation's attributes must be of the same type. Objects are classified as snapshot relations, whose attributes are all nontime 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**. (The name of this model is misleading, because only valid time is supported.) The structure of a historical relation in the Temporal Relational Model is the same as that of a historical relation in LEGOL 2.0 (Figure 3), with one additional restriction. *Value-equivalent* tuples (whose values on the nontime-stamp attributes are identical), although allowed, are required to be *coalesced* (i.e, adjacent intervals replaced with a single interval) so contiguous intervals are not represented by two value-equivalent tuples. 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 *TC-NJOIN* are both join operators defined using intersection semantics. Two other join operators, *TJOIN* and *TNJOIN*, are discussed. They retain the (multiple) time-stamps of underlying tuples in their resulting tuples and are, therefore, outside the algebra because they produce illegal tuples. These various selection and join operators produce differing valid times in the resulting relation.

1.2.4 Sadeghi

Sadeghi's [1987] algebra is similar in many ways to Navathe's. It was designed to support the calculus-based historical query language HQL [Sadeghi et al. 1987], which in turn is based on DEAL [Deen 1985]. In Sadeghi's algebra, all objects are historical relations. Two implicit attributes, **start** and **stop**, record the end points of each tuple's interval of validity. Hence, the structure of a historical relation in Sadeghi's algebra is also the same as that of the historical relation in LEGOL 2.0 (Figure 3). Sadeghi's algebra requires that value-equivalent tuples be coalesced. Temporal versions of the snapshot operators union, difference, Cartesian product, selection, projection, and join are defined. Both Cartesian product and join are defined using intersection semantics. A new operator, *WHEN*, is introduced to perform temporal selection. It maps a historical relation instance onto the intervals that are the time-stamps of tuples in the instance. Whether the result of this operation is another type of object of a historical relation without explicit attributes is unclear.

The next algebra, along with those of Tansel and Lorentzos, incorporates operators to switch between an interval representation, which is space efficient and more appropriate for presentation, and a single chronon-based representation, which is effectively a union of time slices that may be manipulated simultaneously with conventional relational operators but which are also highly space inefficient.

1.2.5 Sarda

Sarda's [1990] algebra associates valid time with tuples. Objects can be either snapshot or historical relations. Unlike the algebras mentioned previously, Sarda's algebra represents valid time in a historical relation as a single, nonatomic, implicit attribute named **period**. Also, unlike the other algebras, a tuple in Sarda's algebra is not considered valid at its right-most boundary point; that is, the interval is closed on the left and open on the right.

Example

R is a historical relation instance in Sarda's algebra containing four tuples:

R =

sname	course	period
"Phil"	"English"	$1 \cdots 2$
"Phil"	"English"	$3 \cdots 5$
"Norman"	"English"	$1 \cdots 3$
"Norman"	"Math"	$5 \cdots 7$

The first two tuples signify that Phil was enrolled in English during fall semester 1980 and fall semester 1981 but not during the spring semester 1981. □

Sarda's algebra retains the basic semantics of some of the set-theoretic operators, extends the definition of one operator to handle valid time directly, and introduces several new operators. Projection and Cartesian product are defined to treat the implicit attribute **period** the same as they would an explicit attribute rather than associating special time-oriented semantics with this attribute as do most of the other algebras. Projection maps a historical relation instance onto either a snapshot or a historical relation instance, depending on whether the implicit attribute **period** is a projection attribute. Similarly, Cartesian product simply combines tuples from two historical relations, without discarding or changing their time-stamps. Hence, the result of a Cartesian product is not a historical relation, which by definition contains one implicit **period** attribute, but rather a snapshot relation with two nonatomic attributes, one containing the period of validity of a tuple from the first argument relation and one containing the period of validity of a tuple from the second argument relation. The semantics of the selection operator is extended to allow for temporal as well as nontemporal predicates. Whether the algebra retains the set-theoretic semantics of union and difference is left unspecified.

EXPAND, *CONTRACT*, *PROJECT-AND-WIDEN*, and *CONCURRENT PRODUCT* are the new operators. *EXPAND* produces, for each chronon in the time-stamp of each tuple in a historical relation, a value-equivalent tuple with that chronon as its time-stamp. *CONTRACT*, the inverse of *EXPAND*, coalesces value-equivalent tuples. *PROJECT-AND-WIDEN* is a form of temporal projection that coalesces value-equivalent tuples, and *CONCURRENT PRODUCT* is Cartesian product defined using intersection semantics.

The remaining algebras use distinct non-first normal form (non-1NF) data models, with attribute value time-stamping and perhaps with multiple values per attribute. The nonatomicity of attribute values is due to their time-varying nature; any time-slice will usually be in first normal form. Hence, the data model of these algebras is an extension of the conventional (1NF) relational model. The representation, viewed as a normal relation, is certainly not in 1NF, but then the algebra does not operate on conventional relations; it operates on historical relations, which are extensions of conventional relations.

1.2.6 Clifford

Clifford [1982] was the first to suggest incorporating the temporal dimension at the attribute level. This idea appeared in the Historical Database Model [Clifford and Warren 1983] and its associated algebra [Clifford and Tansel 1985]. The Historical Relational Data Model [Clifford and Croker 1987], a refinement of that model, is unique in that it associates time-stamps with both the tuple and with each attribute value. The data model allows two types of objects—a set of chronons, termed a *lifespan*, and a historical relation, where each attribute in the relation schema and each tuple in the relation is assigned a lifespan. A relation schema 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). Relations have key attributes, and no two tuples in a relation are allowed to match on the values of the key attributes at the same chronon.

Example

R is a historical relation instance in the Historical Relational Data Model, where {**sname** → {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, **course** → {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}} is the function assigning lifespans to attributes.

R =

Tuple Value		*Tuple Lifespan*
sname	**course**	
1 →"Phil"	1 →"English"	{1, 3, 4}
3 →"Phil"	3 →"English"	
4 →"Phil"	4 →"English"	
1 →"Norman"	1 →"English"	{1, 2, 5, 6}
2 →"Norman"	2 →"English"	
5 →"Norman"	5 →"Math"	
6 →"Norman"	6 →"Math"	

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 someone 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 since 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 algebras, which we describe later, are functions, they have many physical representations. □

The semantics of the relational operators union, difference, intersection, projection, and Cartesian product is extended to handle lifespans directly. For example, the lifespan of each tuple output by Cartesian product is the union of the lifespans of the two tuples in the input relations that contribute to the output tuple. A null value is assigned to an attribute in the output tuple for each chronon that is in the lifespan of the output tuple but not in the lifespan of the input tuple associated with that attribute. Also, temporal versions of Θ-join, equi-join, and natural join are defined using intersection semantics, and several new time-oriented operations are introduced. *WHEN* maps a relation instance onto 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, a *TIME-JOIN* operator is defined that restricts the tuple lifespans of its resulting tuples to the value of a time-valued attribute.

1.2.7 Tansel

Tansel's historical algebra [Clifford and Tansel 1985; Tansel 1986] allows only one type of object—the historical relation. Four types of attributes are, however, supported: Attributes may be either non-time-varying or time-varying, and they may be either atomic-valued or set-valued. The attributes of a relation need not be the same type, and attribute values in a given tuple need not be homogeneous. 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, whereas the value of a time-varying, set-valued attribute is simply a set of such triplets. This data model makes this algebra a convenient target for interpreting queries expressed in the Time-by-Example language [Tansel et al. 1989] and in HQuel [Tansel and Arkun 1986].

Example

R is a historical relation instance in Tansel's algebra, where **sname** is a non-time-varying, atomic-valued attribute and **course** is a time-varying, set-valued-attribute.

R =

sname	course
"Phil"	{([1, 2), "English"), ([3, 5), "English")}
"Norman"	{([1, 3), "English"), ([5, 7), "Math")}

The enrollment history of a student can be recorded in 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 both the temporal dimension of data and presence of set-valued attributes, and several new time-related operations are introduced. The *PACK* operators combines tuples whose attribute values differ for one 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 a 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 nontime-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. Finally Tansel introduces a new operation, termed *enumeration*, to support aggregation [Tansel 1987]. The enumeration operator derives, for a set of chronons or intervals and a historical relation, a table of data to which aggregate operators (e.g., count, avg, min) can be applied.

Example

Let R_2 be the historical relation instance resulting from the unpacking of attribute **course** of R in the previous example containing four tuples.

$R_2 =$

sname	**course**
"Phil"	([1, 2), "English")
"Phil"	([3, 5), "English")
"Norman"	([1, 3), "English")
"Norman"	([5, 7), "Math")

Now, let R_3 be the historical relation instance resulting from the decomposition (*T-DEC*) of attribute **course** of relation R_2, where **sname, course, course$_L$**, and **course$_U$**, are all non-time-varying, atomic-valued attributes, the latter two computed by the *T-DEC* operator.

$R_3 =$

sname	**course**	**course$_L$**	**course$_U$**
"Phil"	"English"	1	2
"Phil"	"English"	3	5
"Norman"	"English"	1	3
"Norman"	"Math"	5	7

□

1.2.8 Gadia

Gadia's [1988] homogeneous model 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 domains [Gadia and Vaishnav 1985]. Temporal elements are closed under union, intersection, and difference, unlike interval time-stamps. 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 a historical relation at time t always produces a conventional snapshot relation without nulls.

Example

R is a historical relation instance in Gadia's homogeneous model.

$R =$

sname	**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" $[5, 7) \rightarrow$ "Math"

Here the interval $[t_1, t_2)$ is the set of chronons $\{t_1, \ldots, t_2 - 1\}$. 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 (otherwise multiple tuples are required). □

A 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 instance onto 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.

Bhargava's two-dimensional model [Bhargava and Gadia 1990, 1991] is an extension of Gadia's homogeneous model; it supports both valid and transaction time. Many of the criteria concerning transaction time that are satisfied by Yeung's algebra, discussed below, are also satisfied by Bhargava's algebra.

1.2.9 Yeung

Gadia's [1986] multihomogeneous model and Yeung's heterogeneous models [Gadia and Yeung 1988; Yeung 1986] are all extensions of the homogeneous model. They lift the restriction that all attribute values in a tuple be functions on the same temporal element, in part to be able to perform Cartesian product without loss of temporal information caused by merging two time-stamps into one. Here we consider only the latest [Gadia and Yeung 1988] of these extensions. In this (termed Yeung's) algebra, temporal elements may be multidimensional to model different aspects of time (e.g., valid time and transaction time). Attribute values are still functions from temporal elements onto attribute value domains, but attribute values need not be functions on the same temporal element. Expressed a different way, some time-slices may contain nulls. Relations are assumed to have key attributes, with the restriction that such attributes be single valued over their interval of validity. Also, no two tuples may match on the ranges of the functions assigned to the key attributes. Hence, in the previous example the attribute **sname** would qualify as a key attribute in the heterogeneous model. The semantics of union, Cartesian product, selection, projection, and join are extended to account for temporally heterogeneous attribute values. Also, temporal variants of selection and join are introduced. The semantics of difference and intersection, however, are left unspecified.

1.2.10 Lorentzos

The Temporal Relational Algebra [Lorentzos 1988; Lorentzos and Johnson 1988] was the first to support time-stamps with a nested granularity and to support periodic events. As with the algebras discussed above, this algebra associates time-stamps with individual attribute values rather than with tuples. Although a time-stamp is normally associated with each of the attribute values in a tuple, a time-stamp may be associated with any nonempty subset of attribute values in a tuple. Furthermore, no implicit or mandatory time-stamp attributes are assumed. Time-stamps are simply explicit, numeric-valued attributes to be viewed and updated directly by the user. 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. A time-stamp in the Temporal Relational Algebra, like one in Sarda's algebra, does not include its rightmost boundary point. Several time-stamp attributes may also be used together to represent a chronon of nested granularity. This algebra is also misnamed, as it only supports valid time.

Example

Let R be a historical relation instance in the Temporal Relational Algebra over the attributes {**sname, course, semester-start, semester-stop, week-start, week-stop**}, where all four time-stamp attributes are associated with both **sname** and **course**. Assume the granularity for the time-stamp attributes **week-start** and **week-stop** is a week relative to the first week of a semester.

$R =$	**sname**	**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"	"Math"	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 fall semester 1980 for only the first eight 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 remain unchanged in the Temporal Relational Algebra. 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*, whereas relations whose tuples contain only time-stamp attributes representing individual chronons of validity are considered to be *unfolded*. Relation R in the above example is folded.

Example

R_2 shown below, is an equivalent representation of R in which the two time-stamp attributes **semester-start** and **semester-stop** have been unfolded onto a single time-stamp attribute **semester**. Note that every value in the original interval defined by the values of the original time-stamp is found in a separate tuple, with a single time-stamp for the **semester** attribute.

R_2 =

sname	cource	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"	"Math"	5	9	17
"Norman"	"Math"	6	9	17

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

In order to do such things as temporal selection and temporal projection in this algebra, the user must first *UNFOLD* each underlying relation, then manipulate the resulting relations using conventional algebraic operators, then *FOLD* the result back into a relation representing intervals. The data model thus differs from the normal relational model only in that certain columns are given a specific interpretation as representing the period of validity of other column(s) in the relation. The operations are defined in terms of the conventional relational algebra but serve to manipulate these particular columns to convert between an interval-based representation and a chronon-based representation.

1.2.11 McKenzie

McKenzie's temporal algebra [McKenzie 1988; McKenzie and Snodgrass 1991] time-stamps attribute values but retains the requirement that attributes be single valued in an effort to achieve the benefits of attribute-value time-stamping (e.g., the ability to perform a Cartesian product) without the implementation complexities of set-valued attributes. The two types of objects in this algebra are the snapshot and historical relations. A *rollback relation* is a sequence of snapshot relations; a *temporal relation* is a sequence of historical relations, both indexed by transaction time. The value of an attribute in a historical relation is always an ordered pair whose components are a value from the attribute's domain and a set of chronons. There is no requirement that the time-stamps of any of the attribute values in a relation be homogeneous, but relations are not allowed to have two tuples with the same value component for all their attributes;

that is, value-equivalent tuples are disallowed.

Example

R is a historical relation instance in McKenzie's algebra containing three tuples.

R =

sname	course
⟨"Phil", {1, 3, 4}⟩	⟨"English", {1, 3, 4}⟩
⟨"Norman", {1, 2}⟩	⟨"English", {1, 2}⟩
⟨"Norman", {5, 6}⟩	⟨"Math", {5, 6}⟩

In McKenzie's 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* ($\hat{\delta}$), is introduced specifically to handle temporal selection and temporal projection. Two operators, snapshot and historical rollback (ρ and $\hat{\rho}$), are available to extract a snapshot or historical state from a rollback or temporal relation, respectively. Finally, two operators are available to perform nonunique and unique aggregation and two to convert between historical and snapshot relations.

1.2.12 Tuzhilin

Tuzhilin proposes two algebras, termed **TA** and **TA**′, as bases for temporal relational completeness [Tuzhilin and Clifford 1990]. When discussing properties satisfied by both, we will refer simply to Tuzhilin's algebra. The operators in these algebras were designed to be powerful yet simple and to be based on a well-accepted formalism of temporal logic [Rescher and Urquhart 1971]. Tuzhilin's algebras apply to historical relational objects supporting discrete, linear-bounded valid time and a time slice operator that produces first normal form relations without nulls. Hence, it could be applied to any of the objects associated with the other algebras discussed in this paper.

The conventional relational operators are given a time-slice consistent semantics, ensuring that the algebra reduces to the relational algebra for each time slice. In Tuzhilin's **TA** algebra, two additional binary operators are proposed, the *future linear recursive operator* $\mathbf{L_F}$ and the *past linear recursive operator* $\mathbf{L_P}$. The expression $C = \mathbf{L_F}\ (A, B)$ computes tuples valid at time $t + 1$ from tuples of C, A, and B valid at time t. C is valid at time $t + 1$ if C and A were valid at time t or if B was valid at time t. $\mathbf{L_P}$ analogously computes tuples at time $t - 1$. In Tuzhilin's **TA**′ algebra, future and past versions of three unary operators are proposed, *sequential union* ($\mathbf{SU_F}$ and $\mathbf{SU_P}$), which "compresses" future or past history into a single time (i.e., everything true at time t or at some time $t' > t$ of the argument relation is considered true at time t in the result relation), *sequential intersection* ($\mathbf{SI_F}$ and $\mathbf{SI_P}$), which gives those tuples that remained constant in the past or future to now, and *shift* ($\mathbf{SH_F}$ and $\mathbf{SH_P}$), which "shifts" tuples forward or backward one time unit.

1.3 Summary

The following summary oversimplifies the algebras in an effort to differentiate them.

- Jones was the first to define a time-oriented algebra.
- Ben-Zvi was the first to add transaction time.
- Navathe defined his algebra primarily to support his extension to SQL called TSQL.
- Sadeghi's algebra was defined primarily to support his extension to DEAL called HQL.
- Sarda, Lorentzos, and Tansel all incorporate operators to switch between an interval representation and a single chronon representation. Lorentzos'

algebra, closest to the conventional relational data model, supports nested granularity time-stamps and periodic time.

- Clifford, Gadia, Yeung, and Tansel all employ non-1NF data models. Clifford emphasizes associating time-stamps with both the attribute value and with the tuple; Gadia emphasizes the homogeneity property; Yeung emphasizes the multihomogeneous property; and Tansel includes four types of attribute values.
- McKenzie time-stamps attribute values but retains the requirement that attributes have only a single value within a tuple.
- Tuzhilin defined his algebras as a metric for temporal completeness.

Tables 1 and 2 summarize the features of the 12 algebras mentioned above. These tables show the range of solutions chosen by the developers of the algebras to 5 of the basic design decisions introduced in Section 1.1. The sixth design decision is not included since only Ben-Zvi's, Yeung's, and McKenzie's algebras support transaction time. Table 1 categories the algebras according to their representation of valid time. Clifford's algebra also associates time-stamps with attributes in a relation schema as well as with tuples in a relation (i.e., the tuple's lifespan). Yeung associates transaction time with attribute values; Ben-Zvi associates transaction time with tuples, and McKenzie associates transaction time with snapshot and historical states.

Table 2 describes other basic features of the types of objects defined and operations allowed in the algebras. The fourth column indicates whether each algebra retains the set-theoretic semantics of the five basic relational operators or extends the operators to deal with time directly. The final column lists new operators introduced specifically to handle the temporal dimension of the phenomena being modeled.

In the next section we discuss a collection of criteria for evaluating temporal extensions of the snapshot algebra. In Section 3 we evaluate these 12 algebras against the criteria.

2. CRITERIA

Although several historical and temporal algebras have been proposed, previous research has not focused on defining criteria for evaluating the relative merit of these algebras. Only Clifford presents a list of specific properties desirable of a temporal extension of the snapshot algebra [Clifford and 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 the proposed algebras. A more comprehensive collection of specific, objective criteria is needed. In this section, we identify 26 such criteria for evaluating temporal extensions of the snapshot algebra. First, we introduce the criteria. With each criterion, we indicate its source, if relevant. We identify the important aspects to consider when defining a temporal algebra. We argue that each is well defined, has an objective basis for being evaluated, and is arguably beneficial. Next, we discuss our reasons for not including as criteria several other properties of historical and temporal algebras. We use the terms *criteria* and *property* to delineate what is included in our evaluation of the algebras and what is not, respectively. Then, we examine in compatibilities among the criteria. For clarity, we represent a historical or temporal operator (either defined on an extension of the relational model or having a different semantics) as $\widehat{op}$ to distinguish it from its snapshot algebra counterpart *op*.

2.1 Adopted Criteria

The criteria appear in alphabetical order. Most are relevant to both valid and transaction time. We identify those few criteria relevant to only one kind of time. We also indicate where each criterion first appeared.

Criterion 1. *All attribute values in a tuple are defined for the same interval(s)* [Gadia 1986]. This requirement, termed

Table 1. Representation of Valid Time in the Algebras

	Single chronon	Interval (two chronons)	Set of chronons (temporal element)
Time-stamped tuples	Ben-Zvi, Jones, Navathe, Sadeghi	Sarda	
Time-stamped attribute values	Lorentzos	Tansel	Clifford, Gadia, McKenzie, Yeung

Table 2. Objects and Operations in the Algebras

Algebra	Objects	Attributes	Standard operations	New operations
Ben-Zvi	Snapshot relation, temporal relation	Atomic-valued	Extended	Time-view
Clifford	Lifespan, historical relation	Functional	Extended	When, Select-If, Select-When, Time-Slice, Time-Join
Gadia	Temporal element, historical relation	Functional	Snapshot semantics	tdom, Temporal selection
Jones	Historical relation	Atomic-valued	Retained	Time intersection, One-sided time intersection, Time union, Time difference, Time-set membership
Lorentzos	Snapshot relation	Atomic-valued	Retained	Extend, Fold, Unfold
McKenzie	Snapshot relation, historical relation	Ordered pairs	Extended	Historical derivation, Historical aggregation, Rollback, conversion
Navathe	Snapshot relation, historical relation	Atomic-valued	Retained	Time-Slice, Inner Time-View, Outer Time-View, TCJOIN, TCNJOIN
Sadeghi	Historical relation	Atomic-valued	Extended	Time Join, When
Sarda	Snapshot relation historical relation	Atomic-valued and non-atomic-valued	Some retained Others extended	Expand, Contract, Project-And-Widen, Concurrent Product
Tansel	Historical relation	Atomic-valued, Set-atomic valued, Triplet valued, Set-triplet valued	Extended	Pack, Unpack, T-Dec, T-Form, Drop-Time, Slice, Uslice, Dslice, Enumeration
Tuzhilin	Historical relation	Independent of representation	Extended	Seq. Union, Shift, Seq. Intersection, Linear Recursive
Yeung	Temporal relation, temporal element	Functional	Extended	Temporal Selection, Temporal Join

homogeneity by Gadia, applies when attributes are set valued, rather than atomic valued, and when valid time is associated with attribute values, rather than tuples; the criterion is trivially satisfied if tuples are time-stamped. Although attributes may change value at different times, this criterion requires that, for each chronon for which some attribute has an associated value in a tuple, every attribute in that tuple has a value for that chronon. If this is ensured, the algebra is simplified. In particular, operators need not be redefined to handle valid time 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 nonempty time-stamps for some, but not all, attributes) need not be considered.

Criterion 2. *Consistent extension of the snapshot algebra* [Clifford and Tansel 1985]. The algebra should be at least as powerful as the snapshot algebra so that queries possible when time was not modeled are not forbidden when time is added. Any relation or algebraic expression that can be represented in the snapshot model should have a counterpart in the temporal model. Thus the algebra should provide, as a minimum, a time-oriented counterpart for each of the five operators that serve to define the snapshot algebra: union, difference, Cartesian product, projection, and selection [Ullman 1988a]. If we assume the function *Transform* converts a snapshot relation instance into its temporal counterpart (with valid and transaction times of t_1 and t_2, respectively), then Figure 6 illustrates the structure of the proof of this criterion for the unary snapshot operators π and σ to show that T_2 (the transformed version of the result of applying the snapshot operator to S) is indeed equal to the result of applying the analogous temporal operator to the transformed version of S. The structure of the proof for the binary operators $\cup$, $-$, and $\times$ is similar.

Criterion 3. *Data periodicity is supported* [Anderson 1981, 1982; Lorentzos 1988; Lorentzos and Johnson, 1988]. 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, a temporal data model should be able to represent such periodic phenomena without having to specify the time of each of their occurrences, and temporal operators should be able to manipulate such periodic data directly.

Criterion 4. *Each collection of legal attribute values, drawn from the appropriate domain(s), is a legal 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 temporal model to retain the simplicity of the snapshot model by not imposing arbitrary extensions. If we extend the snapshot model so valid time is assigned to each attribute, we should extend the concept of attribute independence to include the valid-time component of the attribute as well as the value component of the attribute. Within a tuple, the value or valid-time component of one attribute should not restrict arbitrarily the value or valid-time component of another attribute. Limiting legal tuples to some subset of the possible tuples (such as restricting the time-stamp of one attribute to be identical to the time-stamp of another attribute, given attribute-value time-stamping) adds a degree of complexity to the temporal model not found in the snapshot model. The same considerations should hold for transaction time.

Criterion 5. *Each set of legal tuples is a legal relation*. In the snapshot model, every set of tuples composed of values from appropriate domains is a legal relation. The same should be true of the temporal model. Imposing additional intertuple constraints (such as not allow-

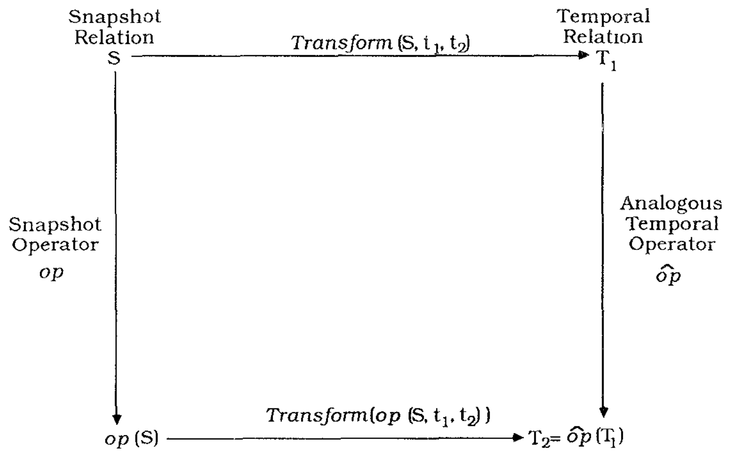
Snapshot Relation S
Transform (S, t_1, t_2)
Temporal Relation T_1
Snapshot Operator *op*
Analogous Temporal Operator $\hat{op}$
op (S)
Transform(*op* (S, t_1, t_2))
$T_2 = \hat{op}(T_1)$

Figure 6. Outline of an equivalence proof.

ing two tuples with identical attribute values but different attribute timestamps or requiring coalescing) adds another degree of complexity to the temporal model not found in the snapshot model.

Criterion 6. *Formal semantics are well defined.* Concise, mathematical definitions for all object types and operations are needed. Without such definitions, the meaning of algebraic operations is unclear. The snapshot algebra already has a formal semantics [Klug 1982].

Criterion 7. *It has the expressive power of a historical or temporal calculus* [Gadia 1986]. There should exist a historical or temporal declarative calculus-based query language to be used as the primary user interface to the temporal DBMS, whose expressive power is subsumed by that of the algebra, which can then serve as the underlying evaluation mechanism. Calculus-based query languages are generally easier to use by novices than algebraic-based languages [Reisner 1981; Reisner et al. 1975]. It should be possible to prove that every statement in the calculus-based query language can be converted into an expression in the temporal algebra having an equivalent semantics.

Criterion 8. *Includes aggregates.* The temporal algebra should provide formal semantics for versions of standard aggregate operations (e.g., sum, count, min), which already appear in several extensions of the relational algebra [Klug 1982].

Criterion 9. *Incremental semantics are defined.* Studies have shown that it may be more efficient to implement some recurring snapshot queries as incrementally maintained materialized views rather than recomputing the queries each time they are asked [Hanson 1987, 1988; Jensen et al. 1991; Roussopoulos 1991]. Because this strategy will likely be applicable to an even larger subclass of temporal queries [McKenzie 1988], an incremental version of the algebra is desirable.

Criterion 10. *Intersection, Θ-join, natural join, and quotient are defined.* In the snapshot algebra, intersection, Θ-join, natural join, and quotient are defined in terms of the difference, selection, projection, and Cartesian product operators [Ullman 1988a]. In a temporal algebra, analogous definitions may, but need not, hold. For example, if the historical versions of the basic operators do not retain the properties of their snapshot counterparts (e.g., satisfy algebraic equivalences), it may not be possible to define historical versions of these operators exactly as they are defined in the snapshot

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\substack{\text{Temporal}\\\text{Relation}\\T} \arrow[r, "\textit{Snapshot}\ t_1,t_2(\mathrm{T})"] \arrow[d, swap, "\substack{\text{Temporal}\\\text{Operator}\\\widehat{op}}"] & \substack{\text{Snapshot}\\\text{Relation}\\S_1} \arrow[d, "\substack{\text{Analogous}\\\text{Snapshot}\\\text{Operator}\\op}"] \\
\widehat{op}(\mathrm{T}) \arrow[r, "\textit{Snapshot}\ t_1,t_2(\mathrm{op(T)})"] & S_2 = op\,(S_1)
\end{tikzcd}$$

Figure 7. Outline of a reduction proof.

algebra. Hence, formal definitions should be given of these additional operators.

Criterion 11. *Is, in fact, an algebra* [Clifford and Tansel 1985]. This criterion is fundamental. Any algebra should define the types of objects supported and the allowable operations on objects of each defined type. This subsumes the property that the algebra be closed.

Criterion 12. *Model does not require null attribute values.* Restriction of attribute values to *nonnull* values is consistent with the snapshot model and simplifies the semantics of the algebra. If missing or unknown information needs to be recorded, then *nulls* are one such representation, but the algebra should not use *nulls* as a matter of course.

Criterion 13. *Multidimensional time-stamps are supported* [Gadia and Yeung 1988]. It may be desirable to associate more than one aspect of time with an object or relationship being modeled. Because valid time, in particular, is a multifaceted aspect of time, time-stamps of a single dimension may be inadequate for recording time in temporal databases. For example, it may be desirable to record both when a train was scheduled to depart and when it actually departed [Lindgreen 1982]. Hence, a temporal data model should support multidimensional time-stamps. Note that this criterion differs from the earlier one concerning periodicity. Satisfaction of the periodicity criterion only requires that the algebra support structured time-stamps that record a single aspect of time. This criterion is relevant only to valid time.

Criterion 14. *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 two-dimensional slice of a four-dimensional temporal relation at a valid time t_1 and transaction time t_2. Hence, for all temporal operators, the snapshot relation obtained by applying a temporal operator to a temporal relation then taking a snapshot should be equivalent to the relation obtained by taking a snapshot of the temporal relation and applying the analogous relational operator to the resulting snapshot relation. Figure 7 illustrates this reduction proof. Informally, this criterion differs from criterion 2 (consistent extension of the snapshot algebra) in that it starts from a temporal relation rather than a snapshot relation. The concern here is that operations on such relations, which may not be transformations of any snapshot relations, may be irreducible to the snapshot counterparts.

Criterion 15. *Restricts relations to first normal form.* The snapshot algebra owes much of its simplicity to the restriction of relations to first normal form, which requires that all attribute values be atomic. Any extension of the snapshot algebra should retain this property, unless doing so will compromise other, equally important properties.

Criterion 16. *Supports a three-dimensional conceptual visualization of historical relations and operations* [Ariav 1986; Ariav and Clifford 1986; Clifford and Tansel 1985]. Brooks [1956] was the first to observe that database relations recording changes to real-world objects over time can be visualized conceptually as three-dimensional objects. This spatial metaphor represents historical relations as three-dimensional objects, whose first two dimensions are tuple (row) and attribute (column) and whose third dimension is valid time. Although these spatial objects are not true cubes, they do possess geometric properties similar to those of cubes.

Example

Consider the historical relation instance R shown below with attribute-value time-stamping containing three tuples.

$R =$

sname	course
("Phil", {1,3,4})	("English", {1,3,4})
("Norman", {1,2})	("English", {1,2})
("Norman", {5,6})	("Math", {5,6})

Figure 8 is a graphical representation of this relation. Hence, this representation of R can be viewed as a three-dimensional object with geometric properties similar to that of a cube. □

If we accept this three-dimensional representation as a user 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 visualization that these operations manipulate spatial objects. For example, the difference operator should take two spatial objects (i.e., historical relations) and produce a third spatial object that represents the volume (i.e., historical information) present in the first spatial object but not present in the second spatial object. Likewise, the Cartesian product operator should take two spatial objects and produce a third spatial object such that each unit of volume (i.e., historical tuple) in the first spatial object is concatenated with a unit of volume in the second spatial object to form a unit of volume in a third spatial object. This description of operations on historical relations as "volume" operations on spatial objects is consistent with the semantics of the individual snapshot algebraic operations as "area" operations on two-dimensional tables extended to account for the additional dimension represented by valid time.

This criterion subsumes the property that the algebra supports *historical queries* concerning valid time [Snodgrass 1987]. An 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. This 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.

This criterion also subsumes the property that the algebra supports a valid-time snapshot operator (c.f., criterion 19).

This criterion has an important implication: If it is satisfied, then *historical data loss* [McKenzie 1988] 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.

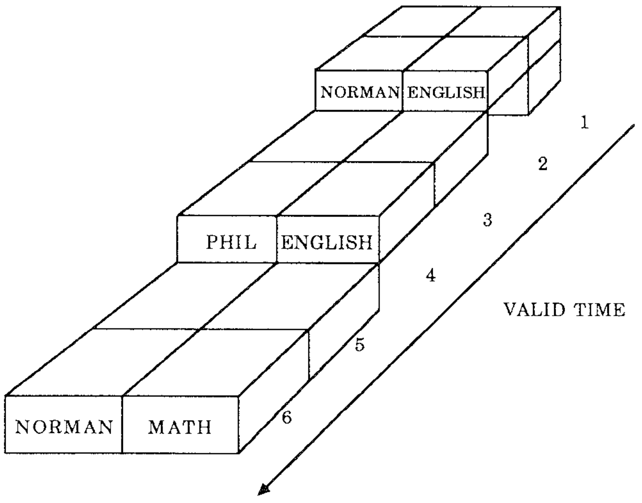

Figure 8. Graphical representation of a historical relation.

Example

Suppose a historical algebra allows attribute-value 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 (where the interval of the resulting tuple is defined to be the intersection of the intervals of the underlying tuples). Now consider the Cartesian product of two historical relations with attribute-value time-stamping: relation instance A, recording when students took courses, and relation instance B, recording when each student resided in a particular state.

$A =$

sname	course
("Phil", {1, 3, 4})	("English", {1, 3, 4})

$B =$

hname	state
("Phil", {1, 2, 3})	("Kansas", {1, 2, 3})

$A \hat{\times} B =$

sname	course	hname	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. □

Criterion 17. *Supports basic algebraic equivalences.* The following commutative, associative, and distributive equivalences, which hold for and in some sense define the snapshot operators, should also hold for their historical counterparts. As most optimization strategies for conventional relational databases are based on these equivalences, a temporal algebra that also satisfies them will be easier to optimize. Implementation efficiency is arguably the most important feature of an algebra. If an algebra cannot be implemented efficiently, it will have no practical application for the development of temporal DBMSs.

$$Q \mathbin{\hat{\cup}} R \equiv R \mathbin{\hat{\cup}} Q$$

$$Q \mathbin{\hat{\times}} R \equiv R \mathbin{\hat{\times}} Q$$

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

$$Q \mathbin{\hat{\cup}} (R \mathbin{\hat{\cup}} S) \equiv (Q \mathbin{\hat{\cup}} R) \mathbin{\hat{\cup}} S$$

$$Q \mathbin{\hat{\times}} (R \mathbin{\hat{\times}} S) \equiv (Q \mathbin{\hat{\times}} R) \mathbin{\hat{\times}} S$$

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

$$Q \mathbin{\hat{\times}} (R \mathbin{\hat{-}} S) \equiv (Q \mathbin{\hat{\times}} R) \mathbin{\hat{-}} (Q \mathbin{\hat{\times}} S)$$
$$\hat{\sigma}_F(Q \mathbin{\hat{\cup}} R) \equiv \hat{\sigma}_F(Q) \mathbin{\hat{\cup}} \hat{\sigma}_F(R)$$
$$\hat{\sigma}_F(Q \mathbin{\hat{-}} R) \equiv \hat{\sigma}_F(Q) - \hat{\sigma}_F(R)$$
$$\hat{\pi}_X(Q \mathbin{\hat{\cup}} R) \equiv \hat{\pi}_X(Q) \mathbin{\hat{\cup}} \hat{\pi}_X(R)$$
$$Q \mathbin{\hat{\cap}} R \equiv Q \mathbin{\hat{-}} (Q \mathbin{\hat{-}} R)$$

Included in this list are the commutative, associative, and distributive equivalences involving only union, difference, and Cartesian product in set theory [Enderton 1977]. Also included in this list are the nonconditional commutative laws involving selection and projection [Smith and Chang 1975; Ullman 1988b]. Finally, the definition of the intersection operator in terms of the difference operator, which holds for the snapshot algebra, should also apply.

Criterion 18. *Supports relations of all four classes* [Snodgrass and Ahn 1985, 1986]. Relations may be classified, depending on their support for valid time and transaction time, as either snapshot, rollback, historical, or temporal relations. Any temporal extension of the snapshot algebra that supports both valid time and transaction time should allow for relations of all four classes.

Criterion 19. *Supports rollback operations* [Ben-Zvi 1982; Snodgrass 1987]. In many database applications, there is sometimes a need to pose queries in the context of past database states. Hence, the algebra should allow relations to be rolled back to past states for query evaluation. The algebra should allow a query unrestricted access to tuples in past database states. Also, the algebra should allow a query access to multiple database states rather than access to a single database state. This criterion is relevant only to transaction time.

This criterion subsumes the property of supporting a three-dimensional visualization of rollback relations and operations. Incorporating transaction time should conceptually add a third dimension, just as incorporating valid time does. Given the special semantics of transaction time discussed above and the fact that support for transaction time requires that all modifications be treated physically as appends [Snodgrass and Ahn 1986], however, we must interpret this criterion with care. Specifically, supporting a three-dimensional visualization of rollback relations implies that modifications conceptually make a copy of the most recent snapshot state (a two-dimensional slice), modify it, then append this to the three-dimensional rollback relation. Additionally, rollback as an operation should involve extracting a two-dimensional (snapshot) slice from a three-dimensional rollback relation or a three-dimensional (historical) slice from a four-dimensional temporal relation.

Criterion 20. *Supports multiple stored schemas* [Ben-Zvi 1982; McKenzie and Snodgrass 1990]. Because a relation's structure, as well as its contents can change over time, a data model incorporating transaction time needs to support multiple stored schemas and to support retrieval via rollback of data consistent with the schema in effect when that data were originally stored.

Criterion 21. *Supports static attributes* [Clifford and Tansel 1985; Navathe and Ahmed 1987]. The algebra should allow for attributes whose role in a tuple is not restricted by time in concert with other time-varying attributes. This feature allows the temporal 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. This criterion is relevant only to valid time.

Criterion 22. *Treats valid and transaction time orthogonally* [Snodgrass and Ahn 1985, 1986]. Valid time and transaction time are orthogonal aspects of time. Valid time concerns the time when events occur and relationships exist in the real world. Transaction time, on the other hand, concerns the time when a record of these events and relationships is stored in a database. Because the two aspects of time are orthogonal, their treatment also should be orthogonal. The valid time assigned to an object in the database should not be restricted by or determined by the transaction time assigned that object. The algebra should allow both retroactive and postactive changes to be recorded. Also, operations involving one aspect of time should not arbitrarily affect the other aspect of time.

Criterion 23. *Tuples, not attribute values, are time-stamped.* Time-stamping tuples, rather than attribute values, 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.

Criterion 24. *Unique representation for each temporal 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 temporal 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.

Examples

To illustrate the potential problems of nonunique representations, consider the relation instances shown below.

A_1 =

sname	course
("Phil", {1,2})	("English", {1,2})
("Phil", {3,4})	("English", {3,4})

A_2 =

sname	course
("Phil", {1,2,3,4})	("English", {1,2,3,4})

B =

sname	course
("Phil", {5,6})	("English", {5,6})

C =

sname	course
("Phil", {2,3})	("English", {2,3})

Clearly, the information content of relation instances A_1 and A_2 is identical; the information content of relation instance B is a continuation of the information in both A_1 and A_2; and the information content of relation instance C is a subset of that contained in both A_1 and A_2. What, however, is the semantics of $A_1 \cup B$? Does the output relation contain three tuples, two tuples, or just one tuple? Is it identical to the result of $A_2 \cup B$? Similarly, what is the semantics of $A_1 \cup C$? Is the single tuple in C represented in the output relation, or is it absorbed by the two tuples in A_1? 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 relation instances A_1 and A_2? Retrieval of "Phil," which is the intuitively correct result when applying this query to A_1, 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. □

Our conclusion is that a selection operator with significantly more complicated semantics would be required to produce results that are correct intuitively. Although the above example assumes attribute-value time-stamping, the same problems arise with tuple time-stamping. 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.

This criterion subsumes the property that tuples with all duplicate attribute values be disallowed, as illustrated in the above example.

Criterion 25. *Unisorted (not multisorted).* In the snapshot algebra all operators take as input and provide as output a single sort of object, the snapshot relation. If possible, a temporal extension of the snapshot algebra should also be unisorted. A multisorted algebra, such as one with an operation that took a historical relation and a snapshot relation and produced a historical relation, would introduce a degree of complexity in the temporal model not found in the snapshot model.

Criterion 26. *Update semantics are specified* [Snodgrass 1987]. Concise, mathematical definitions for update operations, allowed on a relation's schema as well as its content, should be present and well defined. Without such definitions, the meaning of update operations such as tuple insertion and tuple deletion is unclear.

2.2 Properties Not Included

The following properties are either subsumed in concert by several criteria just presented or do not yet have an objective basis for being evaluated. Hence, they are not included as criteria.

The property that the algebra include the concept of key does not have an objective basis. There have been several definitions of *temporal key* proposed [Ariav 1986; Bhargava and Gadia 1991; Clifford and Croker 1987; Gadia and Yeung 1988; Navathe and Ahmed 1987]; no consensus has yet arisen as to which is *the* definition. Additionally, although keys are certainly essential during the design of a relational schema, whether snapshot or temporal, it is unclear whether keys should enter into the definition of a relational algebra. The conventional relational algebra does not include the concept of key (e.g., the algebra allows the union operator to be applied to any two relation instances with identical schemas, even if the result may violate externally defined key constraints). Hence, we do not include this property as a criterion.

The property that the algebra should serve as a standard for defining *temporal completeness* (i.e., an extension of Codd's notion of completeness in the snapshot model) [Clifford and Tansel 1985] currently has no objective basis for evaluating models since there is no consensus definition of temporal completeness, although several have been proposed [Croker and Clifford 1989; Tuzhilin and Clifford 1990].

If the algebra is closed and supports historical queries, it must support nonhomogeneous relations (i.e., relations having tuples whose attribute values are allowed to have different valid times). Therefore, this property that the algebra support nonhomogeneous relations [Gadia 1986] is subsumed by the criteria that the algebra *be an algebra*, that the algebra *be a consistent extension of the snapshot algebra*, and *support* a *three-dimensional view of historical relations and operators*.

The property that the algebra supports transaction time [Snodgrass and Ahn 1986] is subsumed by *supports relations in all four classes*, *supports schema evolution*, *supports rollback operations*, and *treats valid and transaction time orthogonally*.

Although some argue that the algebra should treat valid and transaction time uniformly [Gadia and Yeung 1988], we feel that transaction time, being a concern of the stored data rather than reality, is sufficiently different that it cannot be treated identically to valid time [McKenzie 1988; McKenzie and Snodgrass 1990]. In particular, uniform treatment of valid time and transaction time cannot be extended to include update operations. Transaction time has a specific semantics, very different from that of valid time, that requires special handling on update. Valid time is specified by the user and its value can be derived, via an algebraic expression, from values in underlying relations. Transaction time, however, is simply the time, as measured by a system clock, when update occurs. Its value cannot be specified by the user or derived from underlying relations. For update, therefore, it would seem impossible to treat valid time and transaction time uniformly and still retain a consistent semantics for transaction time.

2.3 Incompatibilities

Not all the criteria discussed above are compatible. There are certain subsets of criteria that no algebra can satisfy. In this section, we examine the incompatibilities among criteria.

The criterion that the algebra support a three-dimensional visualization (criterion 16) is incompatible with the criteria that

- Tuples, not attribute values, be timestamped (23)
- All attribute values in a tuple be defined for the same interval(s) (1)

- The equivalence $Q \hat{\times} (R \hat{-} S) \equiv (Q \hat{\times} R) \hat{-} (Q \times S)$ hold (17). (The remaining equivalences do not conflict.)

First, no algebra can support a three-dimensional model (16) and also time-stamp tuples (23). For the algebra to support such a model, its Cartesian product or join operator must concatenate tuples independent of their valid times and preserve, in the resulting tuple, the valid-time information for each of the underlying tuples. The alternative is historical data loss (c.f., the example in criterion 16). Yet, if the Cartesian product operator assigns different time-stamps to attribute values in its output tuples, the criterion that tuples, not attribute values, be time-stamped cannot be satisfied.

Example

Consider the following single-tuple historical relations with attribute-value time-stamping.

$A =$

sname	course
("Phil", {1, 2, 3})	("English", {1, 2, 3})

$B =$

hname	state
("Norman", {1, 2})	("Iowa", {1, 2})

$C =$

hname	state
("Norman", {2})	("Iowa", {2})

Figure 9 illustrates the representation of historical relations as spatial objects in calculating A $\hat{\times}$(B $\hat{-}$ C) and (A × B) $\hat{-}$(A × C), respectively. The results of these calculations are shown below.

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

sname	course	hname	state
("Phil", {1, 2, 3})	("Math", {1, 2, 3})	("Norman", {1})	("Iowa", {1})

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

sname	course	hnane	state
("Phil", ∅)	("Math", ∅)	("Norman", {1})	("Iowa", {1})

Note that the results are different, contrary to the standard relational model where the results are always identical. □

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 three-dimensional visualization (16).

Second, no algebra can support a three-dimensional model (16) and also require that all attribute values in a tuple be defined for the same interval(s) (1). If the Cartesian product operator required that all attribute values in a resulting tuple be defined over the same interval(s), arbitrary valid-time information associated with the attribute values of the underlying tuples could not be preserved, and the criterion that the algebra support a three-dimensional model could not be satisfied. Yet, if the Cartesian product operator preserved the valid-time information for the attribute values of the underlying tuples in the resulting tuple, attribute values in the resulting tuple would be defined for different intervals and the criterion that all attribute values in a tuple be defined for the same interval(s) could not be satisfied.

Third, no algebra can support a three-dimensional model (16) and also support the distributive property of Cartesian product over difference (17).

There are two other incompatibilities among the criteria. First, the criterion that each set of legal tuples be a legal relation (5) is incompatible with the criterion that there be a unique representation for each relation (24). If every set of legal tuples were allowed to be a legal relation, the algebra could not have a unique representation for each relation.

Example

The following are only two of several equivalent representations of a relation instance *A* over the attributes **{hname, state}** and attribute-value time-stamping.

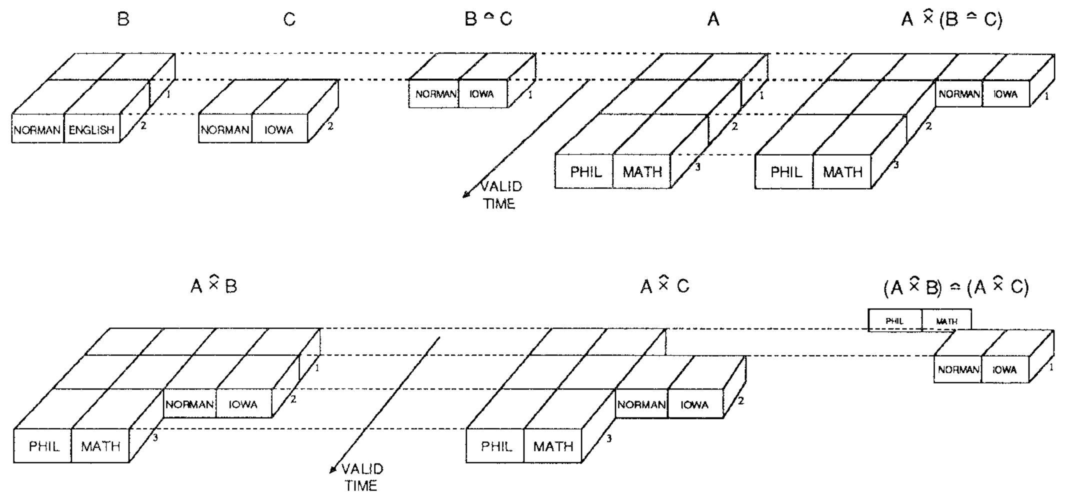

Figure 9. $A \hat{\times} (B \hat{-} C)$ and $(A \hat{\times} B) \hat{-} (A \hat{\times} C)$.

$A_1 =$

hname	state
("Norman", {1, 2, 3, 4})	("Utah", {1, 2, 3, 4})

$A_2 =$

hname	state
("Norman", {1, 2})	("Utah", {1, 2})
("Norman", {3, 4})	("Utah", {3, 4})

Yet, if the algebra allowed only one of these representations, say A_1, to ensure that there be a unique representation for each relation, there would be sets of legal tuples (e.g., A_2) that would not be legal relations. □

Finally, the criteria that the algebra restrict relations to first normal form (15), support a three-dimensional visualization (16), and have a unique representation for each historical relation (24) 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.

Example

Consider the following two single-tuple relation instances.

$A =$

hname	state
("Phil", {1, 2, 3})	("Kansas", {1, 2, 3})

$B =$

hname	state
("Phil", {2})	("Kansas", {2})

To define difference so that $A \hat{-} B$ can be calculated consistent with the user model of historical operators as "volume" operators on spatial objects, the algebra must allow tuples with duplicate attribute values in a relation

$A \hat{-} B =$

hname	state
("Phil", {1, 2})	("Kansas", {1, 2})
("Phil", {3, 4})	("Kansas", {3, 4})

(thereby violating the criterion that there be a unique representation) or allow the time-stamp associated with a tuple to be nonatomic (i.e., a set of intervals rather than a single interval).

$A \hat{-} B =$

hname	state
("Phil", {1, 3})	("Kansas", {1, 3})

□

Thus, to support a three-dimensional visualization 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 attribute values are time-stamped), the algebra must allow non-first normal form relations.

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

		Supports a three-dimensional view of historical states and operations?	
Unique representation for each historical state?		No	Yes
	No	No restrictions	All attributes in a tuple must not be defined over the same interval(s) The distributive property of Cartesian product over difference cannot hold Tuple time stamping cannot be used
	Yes	All sets of legal tuples cannot be legal relations	All attributes in a tuple must not be defined over the same interval(s) The distributive property of Cartesian product over difference cannot hold Tuple time stamping cannot be used. All sets of legal tuples cannot be legal relations Relation states cannot be restricted to first-normal-form

Figure 10. Incompatibilities among criteria

- Supports a three-dimensional visualization (16)
- Unique representation for each historical relation (24)

Figure 10 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, it can satisfy all the other criteria. If, however, the algebra satisfies both of these criteria, there are five criteria that it cannot satisfy. Because no algebra can satisfy all seven of these criteria, we term these criteria *conflicting criteria*.

3. EVALUATION

In this section we evaluate the 12 algebras reviewed in Section 1.2 against the criteria presented in the previous section. Table 3 summarizes the evaluation of these 12 proposals against the criteria. Note that Tuzhilin's algebras are not tied to a particular temporal data model. His algebras are meant *only* to provide an objective measure of the expressive power of algebras and calculi. Hence, many of the criteria can be satisfied or not satisfied depending on the data model chosen. For such criteria, we simply indicate "not applicable."

3.1 Conflicting Criteria

We first evaluate the algebras against the seven criteria introduced in the previous section that are not all compatible.

Criterion 1. *All attribute values in a tuple are defined for the same interval(s).* Gadia's homogeneous model and those that time-stamp tuples satisfy this criterion. In Tuzhilin's algebras, representation restrictions are not imposed on base relations. All-derived relations will, however, associate identical intervals with all attribute values. The other algebras allow attribute time-stamps in a tuple to be disjoint.

Criterion 5. *Each set of legal tuples is a legal relation.* The algebras proposed by Ben-Zvi, Gadia, Jones, Lorentzos, Sarda, and Tansel all satisfy this criterion. Clifford's algebra fails to satisfy this criterion because a relation may not contain two tuples that match on the values of the key attributes at the same chronon. Yeung's algebra, likewise, fails to satisfy this criterion: It does not allow a relation to contain two tuples that match values on the key attributes. In Tuzhilin's algebras, representation restrictions are not imposed. Finally, the algebras proposed by McKenzie, Navathe and Sadeghi also fail to satisfy this criterion. Their algebras require that tuples with identical values for the explicit attributes (termed *value-equivalent*) be *coalesced*; hence, tuples with identical values for the explicit attributes can neither overlap nor be adjacent in time.

Criterion 15. *Restricts relations to first-normal form.* The algebras proposed by Ben-Zvi, Jones, Lorentzos, Navathe, and Sadeghi restrict relations to first normal form. In Tuzhilin's algebras, representation restrictions are not imposed. The other algebras all fail to satisfy this criterion since they allow set-valued attributes, or set-valued time-stamps, or both.

Criterion 16. *Supports a three-dimensional visualization of historical relations and operations.* McKenzie's algebra supports the user-oriented visualization of a historical relation as a three-dimensional object in that it supports nonhomogeneous attribute-value time-stamping and avoids historical data loss as an operator side effect. Operators in Clifford's algebra, with the exception of the join operators, do not satisfy this criterion. Although lifespans are associated with tuples, Cartesian product is defined to prevent historical data loss as an operator side effect through the introduction of nulls into the Cartesian product's output tuples. It is unclear whether Yeung's algebra and Tansel's algebra satisfy this criterion since all operations are not defined formally.

Lorentzos' algebra fails to satisfy this criterion when relations have multiple attribute time-stamps.

Example

Consider the following single-tuple relation instances legal in Lorentzos' algebra.

$A =$

sname	n-start	n-stop	course	c-start	c-stop
"Marilyn"	2	4	"Math"	2	4

$B =$

sname	n-start	n-stop	course	c-start	c-stop
"Marilyn"	1	3	"Math"	1	3

In Lorentzos' algebra, historical difference is defined in terms of the Unfold, set difference, and Fold operators. If we unfold both A and B, first on the **n** attribute and then on the **c** attribute, we get the following relations:

$A' =$

sname	n-time	course	c-time
"Marilyn"	2	"Math"	2
"Marilyn"	2	"Math"	3
"Marilyn"	2	"Math"	4
"Marilyn"	3	"Math"	2
	.		
"Marilyn"	4	"Math"	4

$B' =$

sname	n-time	course	c-time
"Marilyn"	1	"Math"	1
	...		
"Marilyn"	3	"Math"	3

We then apply set difference to the unfolded relations, with the following results:

$A' - B' =$

sname	n-time	course	c-time
"Marilyn"	2	"Math"	4
"Marilyn"	3	"Math"	5
"Marilyn"	4	"Math"	2
"Marilyn"	4	"Math"	3
"Marilyn"	4	"Math"	4

If we then fold the result, again first on **n** and then on **c**, we get

$A \dot{-} B =$

sname	n-start	n-stop	course	c-start	c-stop
"Marilyn"	2	4	"Math"	4	4
"Marilyn"	4	4	"Math"	2	3

This result is inconsistent with the visualization of historical relations as three-dimensional objects and operations on historical relations as "volume" operations on spatial objects, as shown in Figure 11. □

Table 3. Evaluation of Algebras against Criteria[a]

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

The algebras proposed by Ben-Zvi, Gadia, Jones, Navathe, Sadeghi, and Tuzhilin 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. In Gadia's homogeneous model, attribute values are time-stamped, but the time-stamps of individual attribute values in each tuple are required to be identical. This requirement necessitates the definition of Cartesian product using intersection semantics. In Ben-Zvi's algebra, tuples rather than attribute values are time-stamped, and a *Time Join* operator is defined using intersection semantics. Likewise, in Navathe's algebra, tuples rather than attribute values 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 not closed; they produce tuples with two time-stamps. In Jones's algebra, tuples are time-stamped, and Cartesian product operators are defined using both intersection and union semantics. Finally, in Sadeghi's algebra, tuples are time-stamped, and the join and Cartesian product operators are both defined using intersection semantics.

Example

Consider the following single-tuple relation instances:

Table 3. (*Continued*)

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

^{a}√, Satisfies criterion; P, partial compliance (see text); △, criterion not satisfied; N.A , not applicable; ?, not specified in papers; $\surd^1$, see text.

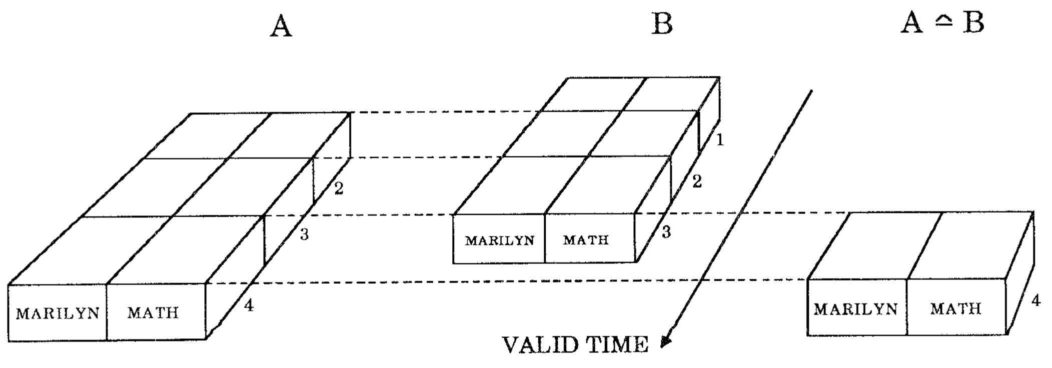

Figure 11. Conceptual view of the difference operator applied to historical relations.

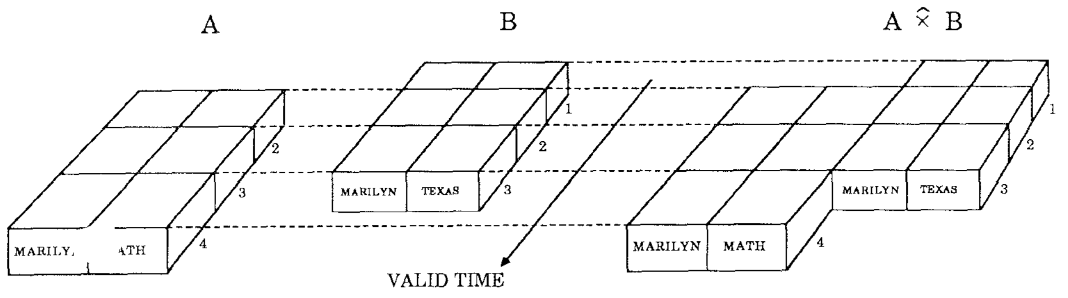

Figure 12. Cartesian product of historical relations.

$A =$

sname	course
("Marilyn", {2,3,4})	("Math", {2,3,4})

$B =$

hname	state
("Marilyn", {1,2,3})	(Texas, {1,2,3})

Using intersection semantics, the Cartesian operator produces the following relation:

$A \dot{\times} B =$

sname	course	hname	state
("Marilyn", {2,3})	("Math", {2,3})	("Marilyn", {2,3})	(Texas, {2,3})

If, however, Cartesian product is represented conceptually as a "volume" operation on spatial objects, we would expect

$A \dot{\times} B =$

sname	course	hname	state
("Marilyn", {2,3,4})	("Math", {2,3,4})	("Marilyn", {1,2,3})	(Texas, {1,2,3})

as illustrated in Figure 12. □

Sarda, in addition to defining a Cartesian product operator using intersection semantics, allows the relational Cartesian product operator to be applied to historical relations. Although tuples in the result retain the time-stamps of their underlying tuples, the result is not a historical relation. Its semantics are left unspecified.

Criterion 17. *Supports basic algebraic equivalences.* Ben-Zvi's, Gadia's, Lorentzos', Sadeghi's, and Tuzhilin's algebras satisfy this criterion. Jones' algebra supports the equivalences, with one exception. The Cartesian product operator defined using union semantics fails to support the distributive property of Cartesian product over difference. All the equivalences, except the distributive property of Cartesian product over difference, also hold for both Clifford's and McKenzie's algebras. Tuzhilin's algebras satisfy this criterion. 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 equivalence since union and difference are not defined formally. Similarly, it is unclear whether all the equivalences hold for Yeung's, Navathe's, and Sarda's algebras.

Criterion 23. *Tuples, not attribute values, are time-stamped.* Ben-Zvi, Jones, Navathe, Sadeghi, and Sarda 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 schema. Tuzhilin does not impose representation restrictions. The other algebras all time-stamp attribute values.

Criterion 24. *Unique representation for each historical relation.* Yeung's algebra supports a unique representation for each temporal relation because it does not allow two tuples to match values on the key attributes. Because McKenzie's algebra allows set-valued time-stamps and disallows value-equivalent tuples, it too supports a unique representation for each temporal relation. Because Navathe and Sadeghi require that value-equivalent tuples be coalesced, their algebras also satisfy this criterion. In Tuzhilin's algebras, representation restrictions are not imposed. None of the other algebras satisfy this criterion. They all allow multiple representations of identical temporal information within a relation, usually by not requiring value-equivalent tuples to be coalesced. Note that Clifford's algebra fails to satisfy this criterion because it only requires that no two tuples in a relation match values on the key attributes *at the same chronon*; a relation may contain value-equivalent tuples, even value-equivalent tuples adjacent in time, as long as they do not overlap in time. Hence, there are still multiple ways to represent a historical relation.

3.2 Compatible Criteria

We now evaluate the algebras against the remaining 19 criteria. Because these criteria are compatible, there is no a priori reason why an algebra cannot be defined that satisfies all these criteria.

Criterion 2. *Consistent extension of the snapshot algebra.* The algebras proposed by Ben-Zvi, Clifford, Gadia, Jones, Lorentzos, McKenzie, Sadeghi, and Tuzhilin satisfy this criterion. Although formal definitions for all operators are not provided for the other algebras, they too are likely to satisfy this criterion.

Criterion 3. *Data periodicity is supported.* Only Lorentzos' algebra satisfies this criterion. His algebra allows multiple time-stamps of nested granularity, which can be used to specify periodicity.

Criterion 4. *Each collection of legal attribute values is a legal tuple.* Tansel's and Yeung's algebras time-stamp attribute values without imposing any inter-attribute dependence constraints. Sarda's algebra encodes a tuple's time-stamp within a single attribute value without imposing any inter-attribute dependence constraints. In Tuzhilin's algebras, representation restrictions are not imposed.

The algebras proposed by Ben-Zvi, Jones, Navathe, and Sadeghi fail to satisfy this criterion because they use implicit attributes to specify the end points of a tuple's time-stamp, 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 schema. Hence, the interval(s) for which an attribute value is defined depends on both the tuple's time-stamp and the attribute's time-stamp in the relation schema. Gadia's homogeneous model fails to satisfy this criterion because all attribute values in a tuple are required to be functions on the same temporal element. Yeung's algebra also fails to satisfy this criterion because relations are restricted to nonnull tuples. Finally, McKenzie's algebra fails to satisfy this criterion because it does not allow the time-stamps of all attribute values in a tuple to be empty.

Criterion 6. *Formal semantics are well defined.* Clifford, Gadia, Lorentzos, McKenzie and Tuzhilin provide a formal semantics for their algebras. Jones, however, 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 and Tansel provide formal semantics for their algebras but 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, and Tansel does not provide formal definitions for his historical union and difference operators. Likewise, Yeung does not provide formal definitions for the historical difference and intersection 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. Sadeghi also requires that value-equivalent tuples be coalesced. He provides formal semantics for all operators, but the semantics of some operators (e.g., union) do not preserve this value-equivalence property of historical relations. Sarda provides formal semantics for five new historical operators and selection and projection when applied to historical relations. Although he allows the Cartesian product operator to be applied to historical relations, the result is not a historical relation. He also does not provide formal definitions for Cartesian product, historical union, and difference.

Criterion 7. *Has the expressive power of a historical or temporal calculus.* Gadia has defined an equivalent calculus for his homogeneous model. McKenzie's algebra has the expressive power of the TQuel [Snodgrass 1987] calculus [McKenzie 1988]. Likewise, Tansel has defined an equivalent calculus for his algebra [Tansel and Arkun 1985]. Ben-Zvi has augmented the SQL query language with a *Time-View* operator and has 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 way as any other snapshot relations. Yeung has defined an equivalent calculus for an earlier version of his algebra [Yeung 1986]. Navathe has defined the historical query language TSQL [Navathe and Ahmed 1987], 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. Sadeghi has defined a historical query language HQL as an extension of the query language DEAL [Sadeghi 1987] and has shown how to map queries in HQL onto expressions in his algebra. Sarda has extended SQL to handle historical queries and has shown how to map sample queries in this language onto expressions in his algebra [Sarda 1990]. Tuzhilin uses a temporal logic [Rescher and Urquhart 1971], **TC**, which he proves is equivalent in expressive power to Tuzhilin's **TA** temporal algebra, and a somewhat less expressive temporal logic, **TC**′, which he proves is equivalent to his **TA**′. The other algebras have not been proven to have equivalent expressive power of a historical or temporal calculus.

Criterion 8. *Includes aggregates.* Ben-Zvi and McKenzie define historical aggregate operators formally as part of their 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 "historical navigation" operators (e.g., First), which act similarly to other historically oriented aggregates. The other algebras do not include any aggregate operators.

Criterion 9. *Incremental semantics are defined.* An incremental version of all operators in McKenzie's algebra has been defined [McKenzie 1988]. An incremental version of none of the other algebras is provided.

Criterion 10. *Intersection, Θ-join, natural join, and quotient are defined.* Historical versions of these four operators are defined for McKenzie's algebra. Ben-Zvi defines a join operator, and Clifford defines intersection, Θ-join, and natural join operators. Gadia defines intersection, Θ-join, and natural join in his homogeneous model. Yeung defines all four operators in an earlier version of his algebra [Yeung 1986], but they are not defined in the later version [Gadia and Yeung 1988]. Finally, Navathe defines historical versions of join and natural join. None of the other algebras defines historical versions of these operators.

Criterion 11. *Is, in fact, an algebra.* Clifford's algebra fails to satisfy this criterion because it is not closed under union, difference, or intersection. The historical versions of these binary operators are defined for two relations only if they are *merge compatible* (i.e., tuples from the two relations that match on the values of the key attributes at some chronon must also match on all their attribute values at each chronon in the intersection of their lifespans). Likewise, Yeung's algebra does not satisfy this criterion because it is not closed under union. The union of two relation instances is undefined if there are tuples in the instances that match on the values of the key attributes but have different values at some time for some attribute. It is unclear whether Sarda's proposal satisfies the closure property since Cartesian product of historical relations, although allowed, produces a result that is not a historical relation. Its semantics, however, are left unspecified. Sadeghi does not specify the semantics of the *WHEN* operator or indicate whether this operator is closed. Two of Navathe's operators, *TJOIN* and *TNJOIN*, are not closed. Each of the other proposals satisfies this criterion.

Criterion 12. *Model does not require null attribute values.* Clifford's algebra fails to satisfy this criterion. The Cartesian product operator assigns null values to attributes in an output tuple for each chronon that is in the lifespan of the output tuple but not in the lifespan of the input tuple associated with that attribute value. The other algebras being evaluated all satisfy this criterion.

Criterion 13. *Multidimensional time-stamps are supported.* Only Yeung's algebra satisfies this criterion. In Tuzhilin's algebras, representation restrictions are not imposed. None of the other algebras supports multidimensional time-stamps. Extension of McKenzie's algebra to support multidimensional time-stamps, however, has been considered [McKenzie 1988].

Criterion 14. *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, Jones, and Tuzhilin imply that the operators are defined using snapshot semantics. Because Navathe, Sadeghi, and Sarda all assume tuple time-stamping, their algebras also satisfy this criterion. Although formal definitions have not been provided for all operators in Yeung's algebra, it can satisfy this criterion only through the introduction of distinguished *null's* when taking snapshots. Because the algebras of McKenzie, Tansel, and Lorentzos allow nonhomogeneous attribute time-stamps, they also satisfy this criterion only through the introduction of distinguished *null's* when taking snapshots. Likewise, because Clifford does not require that all attribute values in a tuple be defined for the same lifespan (i.e., an attribute's value in a tuple is specified only for chronons in the intersection of the tuple's lifespan and the attribute's lifespan in the relational schema), his algebra also satisfies this criterion only through the introduction of distinguished *null's* when taking snapshots.

Criterion 18. *Supports relations of all four classes.* Yeung's algebra, because it allows multidimensional time-stamps, can support relations of all four classes. McKenzie's algebra also satisfies this criterion. Ben-Zvi's model, although it supports both valid time and transaction time, can support rollback and historical relations only by embedding them in temporal relations. The other algebras, since they do not support transaction time, cannot support rollback or temporal relations.

Criterion 19. *Supports rollback operations.* McKenzie's algebra satisfies this criterion. The rollback operators allow queries to be posed on one or more arbitrary relations. Yeung's algebra also satisfies this criterion; transaction time is treated simply as another dimension in a multidimensional temporal element. Ben-Zvi's algebra, although it allows rollback, achieves only partial satisfaction of this criterion because it requires that the tuples participating in a query all have a specified valid time in common. All operations in Ben-Zvi's algebra are defined in terms of a transaction time t_s and a valid time t_e. During expression evaluation, rollback occurs to the relation at t_s, but only tuples valid at t_s are accessed. None of the other algebras supports rollback operations.

Criterion 20. *Supports multiple stored schemas.* McKenzie's algebra satisfies this criterion. Ben-Zvi, while describing an approach for representing an evolving schema as a temporal relation, does not include provisions for schema evolution in the formal semantics of his algebra. Hence, his algebra fails to satisfy this criterion. Yeung, although supporting transaction time, does not address the problem of schema evolution. An approach for handling schema changes in Navathe's formalization has been developed [Martin et al. 1987], but the algebra was not extended to support schema evolution. Because the other algebras do not support transaction time, they too fail to satisfy this criterion.

Criterion 21. *Supports static attributes.* Lorentzos', Navathe's, Sadeghi's, and Tansel's algebras satisfy this criterion by allowing both time-dependent and non-time-dependent attributes. McKenzie's and Yeung's algebras also support static attributes. In these two algebras, the time-stamp of an attribute value can be defined independently of the time-stamps of any of the other attribute values in a tuple. A static attribute would be represented in McKenzie's algebra as an attribute assigned the entire time domain. Clifford's algebra fails to satisfy this criterion because an attribute's value in a tuple cannot be specified for chronons that are not in the tuple's lifespan. In Tuzhilin's algebras, representation restrictions are not imposed. The other four 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.

Criterion 22. *Treats valid and transaction time orthogonally.* Ben-Zvi's and Yeung's algebras satisfy this criterion: They support retroactive and postactive changes and allow independent assignments of valid time and transaction time without restrictions. McKenzie's algebra imposes the restriction that rollback be performed first, followed by historical selection, on temporal relations. The other algebras all fail to satisfy this criterion because they do not support transaction time.

Criterion 25. *Unisorted (not multisorted).* The algebras proposed by Jones, Lorentzos, Sadeghi, Tansel, Yeung, and Tuzhilin are unisorted in that they define only one object sort. All the other algebras are multisorted. McKenzie defines algebraic operators on snapshot relations and historical relations. Gadia's homogeneous model is a multisorted algebra; its sorts are historical relations and temporal expressions. Clifford defines a multisorted algebra whose sorts are historical relations and lifespans. Ben-Zvi allows both snapshot and temporal relations, whereas Navathe allows both snapshot and historical relations. Finally, Sarda defines a projection operator that is allowed to map a historical relation instance onto a snapshot relation instance.

Criterion 26. *Update semantics are specified.* McKenzie'a proposal satisfies this criterion. Ben-Zvi defines the semantics of tuple insertion, deletion, and modification but does not extend the formalization to include schema evolution. The other proposals do not consider update semantics in their formalizations.

3.3 Criteria Metaproperties

In Section 2.2, we identified several criteria that were subsumed by other, more important criteria. It is desirable that the remaining criteria are independent in that no combination of criteria (logically) imply another criteria. Given the number of criteria, a complete analysis is infeasible. Instead, we have taken a more empirical approach, examining whether any (single) criterion logically implies any other criterion, as indicated by the satisfaction or unsatisfaction by the algebras surveyed here.

First, to detect criteria that were logically equivalent to other criteria, we attempted to identify, for each pair of criteria, an algebra in which the two criteria *differ*, that is, where one was satisfied while the other was not satisfied. The presence of such an algebra proves by example that the two criteria are not logically equivalent; the absence of such an algebra suggests a logical relationship between the pair. Seventeen out of a total of 325 pairs did not have such an algebra. Eleven of these pairs were eliminated because the criteria were obviously not related. The other six have a partial counterexample in that for each pair there exists an algebra in which one of the criteria was satisfied and the other was partially not satisfied.

Second, to detect criteria that were "negatively equivalent" (i.e., where the satisfaction of one criterion logically implies that the other criterion cannot possibly be satisfied), we attempted to identify, for each pair of criteria, an algebra in which the two criteria *agree*, that is, where both were either satisfied or were not satisfied. As before, the presence of such an algebra provides a proof by example that the two criteria are not negatively equivalent; the absence of such an algebra indicates the possibility of negative equivalence. Such an algebra was identified for all but four out of 325 possible pairs. Concerning *consistent extension of the snapshot algebra* (2) and *each collection of legal attribute values is a legal tuple* (4), it appears that Sarda's, Tansel's, and Yeung's algebras satisfy both criteria, although a definitive statement must await formal definitions for all operators. Since there is little experience with multidimensional time-stamps (13), its observed negative correlation with two other criteria is uncertain. Finally, one pair was rejected because the criteria were obviously not related.

In Section 2.3, we argued that seven of the criteria, labeled conflicting, were not mutually satisfiable. We conjecture that the remaining criteria are compatible in that it is theoretically possible for all to be satisfied by an algebra. As a partial test of this conjecture, we considered pairwise satisfiability. Specifically, we attempted to identify, for each pair of compatible criteria, an algebra that simultaneously satisfied both. The absence of such an algebra indicates that perhaps the two criteria cannot be simultaneously satisfied and hence are incompatible. Out of 171 pairs possible, 27 pairs suggested incompatibilities, of which 14 were rejected on first principles. *Data periodicity* (3) appears to be incompatible with five other criteria; insufficient experience with this aspect prevents a definite characterization. The same holds for *each collection of legal attribute values is a legal tuple* (4) (four criteria) and *multidimensional time-stamps* (13) (also four criteria).

Our conclusion from this informal analysis is that, with the exception of three criteria [*data periodicity* (3), *each collection of legal attribute values is a legal tuple* (4), and *multidimensional time-stamps* (13)], for which there is little experience (each is satisfied by at most two algebras), the criteria are not pairwise logically equivalent (in either direction), and the compatible criteria are not pairwise incompatible.

4. SUMMARY

In this paper, we examined 12 temporal algebras, considering the objects each defines and the operations on those objects each provides. We identified seven conflicting criteria that cannot simultaneously hold in any algebra and 19 compatible criteria. We then evaluated the algebras on these criteria. As was shown in Figure 10, the subset of conflicting criteria that an algebra can satisfy is necessarily dependent on whether

Unique representation for each historical state?	Supports a three-dimensional view of historical states and operations: No	Supports a three-dimensional view of historical states and operations: Yes
No	Ben-Zvi Gadia Jones Lorentzos Sarda Tuzhilin	Clifford Tansel
Yes	Navathe Sadeghi Tuzhilin	Yeung McKenzie

Figure 13. Classification of algebras according to criteria satisfied.

the algebra supports a three-dimensional visual model and whether each historical relation in the algebra has a unique representation. Based on these two criteria, we can identify the "approach" of each algebra, as shown in Figure 13. The quadrant an algebra occupies indicates those aspects the algebra's designer(s) chose to emphasize. For example, Yeung chose to satisfy both criteria, at the cost of not being able to satisfy the remaining five conflicting criteria listed in Figure 10. Ben-Zvi made exactly the opposite decision: His algebra satisfies the five criteria, but not the other two. In Tuzhilin's algebras, representation restrictions are not imposed. Hence, there may or may not be a unique representation for each historical state, and thus his algebras appear in two quadrants. There is obviously no consensus as to which approach is superior; each has several proponents. The design space has been explored in the sense that all combinations of basic design decisions have at least one representative algebra.

Concerning the compatible criteria, there is no a priori reason all cannot be simultaneously satisfied (Section 3.3 contains a partial argument to that effect). One measure of the quality of an algebra is the extent to which it satisfies these criteria. All algebras surveyed here are deficient, some more so than others.

We have attempted to identify as many criteria as possible (we uncovered a total of 36). We characterize criteria as conflicting (7), consistent (19), and rejected (10), based on logical and empirical arguments. We assign the algebras to four possible approaches based on the conflicting criteria to determine the quality of the algebras based on the consistent criteria.

We view the comparison methodology to be as important a contribution as the analysis of the individual algebras. Since no algebra was identified as being objectively superior, future research must first prove one of the four approaches to be the most appropriate, then define an algebra taking this approach that satisfies all of the consistent criteria.

ACKNOWLEDGMENTS

We 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 temporal algebras The reviewers, Gregory Andrews and Christian S. Jensen provided detailed suggestions that improved this paper. Salvatore March's very thorough subsequent critique inspired many further improvements.

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 grants DCR-8402339 and IRI-8902707 and ONR grant N00014-86-K-0680.

REFERENCES

ALLEN, J. F., AND HAYES, P. J. 1985. A commonsense theory of time, in *Proceedings of the International Joint Conference on Artificial Intelligence.* (Los Angeles, Calif. Aug. 1985), pp. 528-531.

ANDERSON, T. L. 1981. The database semantics of time. Ph. D. dissertation, University of Washington.

ANDERSON, T. L. 1982. Modeling time at the conceptual level, in *Proceedings of the International Conference on Databases: Improving Usabılıty and Responsiveness*. P. Scheuermann, Ed. (Jerusalem, Israel, June 1982). Academic Press, pp. 273-297.

ARIAV, G 1986. A temporally oriented data model. *ACM Trans. Database Syst. 11*, 4 (Dec), 499-527.

ARIAV, G., AND CLIFFORD, J. 1986. Temporal data management: Models and systems, in *New Directions for Database Systems*. Ablex Publishıng Corporation, Norwood, N.J., Chap 12, pp. 168-185.

BEN-ZVI, J. 1982. The time relational model. Ph.D. dissertation, Computer Science Department, Univ California, Los Angeles.

BHARGAVA, G , AND GADIA, S. K. 1990. The concept of error in a database: An application of temporal databases, in *Proceedings of 1990 COMAD International Conference on Management of Data.* N. Prakash, Ed. Tata McGraw-Hill, New Delhi, pp. 106-121.

BHARGAVA, G., AND GADIA, S. K. 1991. Relational database systems with zero information-loss. *IEEE Trans. Knowledge Data Eng.*. To be published.

BROOKS, F. P. 1956. The analytic desıgn of automatic data processing systems. Ph.D. dissertation. Harvard Univ.

BUBENKO, J. A., JR. 1977. The temporal dimension in information modeling, in *Architecture and Models in Data Base Management Systems.* North-Holland Publishing, The Netherlands, pp. 93-118.

CHEN, P. P-S. 1976. The entity-relationship model: Toward a unified view of Data *ACM Trans. Database Syst. 1*, 1 (Mar.), 9-36.

CLIFFORD, J. 1982. A model for historical databases, in *Proceedıngs of Workshop on Logıcal Bases for Data Bases.* (Toulouse, France).

CLIFFORD, J., AND CROKER, A 1987. The historical relational data model (HRDM) and algebra based on lifespans, in *Proceedings of the Internatıonal Conference on Data Engineering*. (Los Angeles, Calif.). IEEE Computer Society Press, pp. 528-537.

CLIFFORD, J , AND RAO, A. 1987 A simple, general structure for temporal domains, in *Proceedings of the Conference on Temporal Aspects in Information Systems.* (France). AFCET, pp. 23-30.

CLIFFORD, J., AND TANSEL, A. U. 1985. On an algebra for historical relational databases: Two views, in *Proceedings of ACM SIGMOD Internatıonal Conference on Management of Data.* S. Navathe, Ed. (Austin, Texas, May), ACM Press, pp. 247-265.

CLIFFORD, J., AND WARREN, D. S. 1983. Formal semantics for time in databases. *ACM Trans. Database Syst. 8*, 2 (June), 214-254.

CODD, E. F. 1970 A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (June), 377-387.

CODD, E. F. 1990. *The Relatıonal Model for Database Management: Versıon 2.* Addison-Wesley Publishing Company, Reading, Mass.

CROKER, A., AND CLIFFORD, J. 1989. On completeness of historical relational data models. Tech. Rep. New York University.

DADAM, P., LUM, V., AND WERNER, H.-D. 1984 Integration of time versions into a relational database system, ın *Proceedings of the Conference on Very Large Databases*. U. Dayal, G. Schlageter, and L.H. Seng, Eds. (Singapore, Aug.), pp. 509-522.

DATE, C J. 1986. An informal definition of the relational model, in *Relational Databasc: Selected Writings.* Addison-Wesley, Reading, Mass., Chap. 2. pp. 21-31.

DEEN, S. M. 1985. DEAL: A relatıonal language with deductions, functions and recursions. *Data and Knowledge Engıneerıng, 1.*

ELMASRI, R., AND NAVATHE, S. B. 1989. *Fundamentals of Database Systems.* Benjamin/Cummings Pub. Co., 1989.

ENDERTON, H. B. 1970. *Elements of Set Theory.* Academic Press, New York.

GADIA, S. K. 1986. Toward a multıhomogeneous model for a temporal database, in *Proceedings of the International Conference on Data Engineering.* (Los Angeles, Calif.). IEEE Computer Society Press, pp. 390-397.

GADIA, S. K. 1988. A homogeneous relational model and query languages for temporal databases. *ACM Trans. Database Syst. 13*, 4 (Dec), 418-448.

GADIA, S. K., AND VAISHNAV, J. H. 1985. A query language for a homogeneous temporal database, in *Proceedıngs of the ACM Symposium on Principles of Database Systems* (Mar.), ACM Press, pp. 51-56.

GADIA, S. K., AND YEUNG, C S. 1988 A generalized model for a relational temporal database, in *Proceedings of ACM SIGMOD Internatıonal Conference on Management of Data.* (Chicago, Ill., June 1988). ACM Press, pp 251-259.

HANSON, E. N. 1987. A performance analysis of view materialization strategies, in *Proceedings of the ACM SIGMOD Internatıonal Conference on Management of Data.* U. Dayal and I. Traiger, Eds. (San Francisco, Calif., May 1987). ACM Press, pp. 440-453.

Hanson, E. N. 1988. Processing queries against database procedures: A performance analysis, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* H. Borol and P.-A. Larson, Eds., (Chicago, Ill., June 1988) ACM Press, pp 295-302.

Jensen, C S , Mark, L , and Roussopoulos, N. 1991. Incremental implementation model for relational databases with transaction time. *IEEE Transactions on Knowledge and Data Engineering.* To be published

Jones, S., Mason, P , and Stamper, R. 1979. LEGOL 2.0: A relational specification language for complex rules. *Inf. Syst. 4*, 4, (Nov), 293-305.

Klopprogge, M. R. 1981. 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.* (Washington, D.C Oct). pp. 477-512.

Klug, A. 1982. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *J. ACM 29*, 3 (July), 699-717.

Lindgreen, P. 1982. The information graph, in *Proceedings of the 1st Scandinavian Research Seminar on Information Modelling and Database Management.* (University of Tampere, Jan), pp. 103-127.

Lorentzos, N. A 1988. A formal extension of the relational model for the representation and manipulation of generic intervals Ph.D. dissertation, Birkbeck College. University of London.

Lorentzos, N. and Johnson, R. 1988. Extending relational algebra to manipulate temporal data *Inf. Syst. 13*, 3, 289-296.

Lum, V., Dadam, P., Erbe, R., Guenauer, J , Pistor, P., Walch, G., Werner, H., and Woodfill, J 1984. Designing DBMS support for the temporal dimension, in *Proceedings of ACM SIGMOD International Conference on Management of Data.* B. Yormark, Ed. (Boston, Mass , June). ACM Press, pp. 115-130.

Maier, D. 1983. *The Theory of Relational Databases.* Computer Science Press, Rockville, Md.

Manola, F., and Dayal, U 1986 PDM: An object-oriented data model, in *Proceedings of the International Workshop on Object-Oriented Database Systems*

Martin, N. G., Navathe, S. B., and Ahmed, R 1987. Dealing with temporal schema anomalies in history databases, in *Proceedings of the Conference on Very Large Databases.* P. Hammersley, Ed (Brighton, England, Sept.) pp. 177-184.

McKenzie, E. 1986. Bibliography: Temporal databases. *ACM SIGMOD Record 15*, 4 (Dec), 40-52.

McKenzie, E. 1988 An algebraic language for query and update of temporal databases. Ph.D. dissertation. Computer Science Department, Univ. of North Carolina at Chapel Hill.

McKenzie, E., and Snodgrass, R. 1990. Schema evolution and the relational algebra. *Inf. Syst. 15*, 2 (June), 207-232

McKenzie, E , and Snodgrass, R. 1991. Supporting valid time in an historical relational algebra: Proofs and Extensions. Tech. Rep. TR 91-15 Dept. of Computer Science, Univ. of Arizona

Navathe, S. B., and Ahmed, R. 1987. TSQL—A language interface for history databases, in *Proceedings of the Conference on Temporal Aspects in Information Systems.* (France, May) AFCET, pp. 113-128.

Navathe, S. B., and Ahmed, R. 1989. A temporal relational model and a query language. *Inf. Sci. 49*, 147-175

Overmyer, R., and Stonebraker, M. 1982. Implementation of a time expert in a database system. *ACM SIGMOD Record 12*, 3 (Apr), 51-59

Reisner, P. 1981. Human factors studies of database query languages: A survey and assessment *ACM Comput. Surv. 13*, 1 (Mar.), 13-31.

Reisner, P., Boyce, R F., and Chamberlin, D. D. 1975. Human factors evaluation of two data base query languages: Square and sequel, in *Proceedings of the AFIPS National Computer Conference* (Arlington, Va) AFIPS Press, pp 447-452.

Rescher, N. C., and Urquhart, A. 1971. *Temporal Logic.* Springer-Verlag, New York.

Roussopoulos, N 1991. The incremental access method of view cache: Concept, algorithms, and cost analysis. *ACM Trans Database Syst.* To be published.

Sadeghi, R .1987 A database query language for operations on historical data Ph D. dissertation. Dundee College of Technology.

Sadeghi, R., Samson, W. B., and Deen, S M. 1987. HQL: A historical query language. Tech. Rep. Dundee College of Technology.

Sarda, N. 1990. Algebra and query language for a historical data model *Comput. J. 33*, 1 (Feb), 11-18.

Shipman, D W. 1981 The functional data model and the data language DAPLEX. *ACM Trans. Database Syst. 6*, 1 (Mar.), 140-173

Smith, J. M., and Chang, P. Y.-T 1975 Optimizing the performance of a relational algebra database interface. *Commun ACM 18*, 10 (Oct), 568-579.

Snodgrass, R. 1987. The temporal query language TQuel. *ACM Trans. Database Syst. 12*, 2 (June), 247-298

Snodgrass, R , and Ahn, I. 1985 A taxonomy of time in databases, in *Proceedings of ACM SIG-*

MOD International Conference on Management of Data. S. Navathe, Ed. (Austin, Tex, May). ACM Press, pp. 236-246.

SNODGRASS, R , AND AHN, I. 1986. Temporal databases. *IEEE Comput. 19*, 9 (Sept.), 35-42.

SOO, M. D. 1991. Bibliography on temporal databases. *ACM SIGMOD Record 20*, 1 (Mar.), 14-23

STAM, R., AND SNODGRASS, R. 1988. A bibliography on temporal databases. *Database Eng. 7*, 4 (Dec.), 231-239

TANSEL, A. U. 1986. Adding time dimension to relatıonal model and extending relational algebra. *Inf. Syst. 11*, 4, 343-355.

TANSEL, A. U. 1987. A statistical interface for historical relational databases, in *Proceedings of the International Conference on Data Engineering* (Los Angeles, Calif., Feb). IEEE Computer Society Press, pp. 538-546.

TANSEL, A. U., AND ARKUN, M. E. 1985 Equivalence of historical relational calculus and historical relational algebra. Tech. Rep. Bernard M. Baruch College, City Univ. of New York.

TANSEL, A U., AND ARKUN, M. E. 1986. HQUEL: A query language for historical relational databases, in *Proceedings of the 3rd International Workshop on Statistical and Scientific Databases.*

TANSEL, A. U., ARKUN, M. E., AND OZSOYOGLU, G. 1989. Time-by-example query language for historical databases. *IEEE Trans. Softw. Eng. 15*, 4 (Apr.) 464-478.

THOMPSON, P. M. 1991. A temporal data model based on accounting principles. Ph D dissertation. Dept. of Computer Science, Univ. of Calgary.

TUZHILIN, A , AND CLIFFORD, J 1990 A temporal relational algebra as a basis for temporal relational completeness, in *Proceedıngs of the Conference on Very Large Databases*. (Brisbane, Australia).

ULLMAN, J. D. 1988b. *Database and Knowledge – Base Systems*. Vol. II. Computer Science Press, Rockville, Md

ULLMAN, J. D. 1988a. *Principles of Database and Knowledge-Base Systems* Vol I Computer Science Press, Rockville, Md , Vol 1.

VANDENBERG, S. L., AND DEWITT, D. J. 1991. Algebraic support for complex objects wıth arrays, identity, and inheritance, in *Proceedıngs of ACM SIGMOD Internatıonal Conference on Management of Data*. J. Clifford and R. King, Eds. (Denver, Col., May). ACM Press, pp 158-167.

YEUNG, C. S. 1986 Query languages for a heterogeneous temporal database. Master's Thesis, EE/CS Dept. Texas Tech. Univ.

Receıved October 1987, fınal revısıon accepted July 1991.