Journal of the Korean
OR/ MS Society
Vol. 20, No. 2, August 1995

A Temporal Relational Database : Modeling and Language

Jae-Kyeong Kim*

Abstract

A temporal database systems provides timing information and maintains history of data compared to the conventional database systems. In this paper, we present a temporal relational database which use an interval-stamping method for instant-based events and for interval-based states. A set of temporal algebraic operators are developed on an instance of time and interval of time so that we can manipulate events and states at a same time. The set of operations is the basis for creating a relational algebra that is closed for temporal relations. And temporal SQL is also suggested as a temporal query relational language for our algebraic operations on temporal relations.

1. Introduction

Most conventional databases represent the state of the universe of discourse at a particular instant of time, a static snapshot of the real world. As the real world changes, new values are incorporated in the database by replacing the old values. Conventional database systems maintain no history of changes. The database thus represents only the current state of some domain rather a history of that domain [8, 10, 12, 17]. A temporal data model, on the contrary, preserves the complete history of objects (entities and relationships) by retaining their previous values. Many applications require storing and accessing of such historical information. Examples can be found in scientific studies (forecasting, medical care), and administrative and operational control (project management, personal management) [4, 10, 11].

Many of the temporal relational data models use an object's valid time in the real world to timestamp data in a relational model. Sarda [14] has summarized the following four different methods for including timestamps at the representational level: (1) instant stamping of tuples: each tuple includes a time value at which the data in the tuple became current;

(2) interval stamping of tuples: each tuple includes two time values that define a time interval during which the data were current; (3) instant stamping of attributes: a time value is associated with each attribute value, and (4) interval stamping of attributes: two time values defining an interval are associated with each attribute value in a tuple. In addition, some hybrid representations have been created that provide both instant and interval stamping [7, 19]. The hybrid models require two distinct types of relations, which prevents the development of a sound relational algebra that is closed for operations on relations. This paper uses interval stamping method, because instances can be stored as intervals of zero length and time-invariant data as intervals of infinite length. Time stamping of attributes have been reported that the results in relations not in first normal form provides difficulties to develop algebraic operators. So, interval stamping of tuples is preferred because it retains the simple structure of the relational data model. [2, 6, 11].

In this paper, we have examined two aspects of temporal relational databases: (1) an interval-stamping method for instant-based events and for interval-based states that can incorporate temporal uncertainty, and (2) a set of algebraic operators for interval-stamped relations that is closed.

Our model of time uses the timepoint as the basis for representing both events and states. We formalize an interval-stamping method that uses two timepoint attributes. Our data model stores events and states as intervals using two timepoint attributes. The representation of Navathe and Ahmed [12] assumes that the pair of attributes for interval timestamps, denoted as TIME-START and TIME-STOP, are timepoints at a single granularity. If the value of an attribute occurs only at a single point in time, the values of TIME-START and TIME-STOP are equivalent. These researchers do not provide modeling techniques for handling incomplete temporal data, such as interval ranges for events. They do not provide a direct way for relating concurrent data from two or more relations. This is regarded as a very basic requirement for users of temporal databases. The question of different time granularities within the same database is not yet addressed adequately. Sarda [14] models the temporal uncertainty of instant-based data by timestamped events with timepoints of various granularities, such as YEAR, YEAR:MONTH, and YEAR:MONTH:DAY. Each temporal relation in his data model contains a pair of timepoint attributes that are FROM and TO, and that have a single suitable granularity for that table. To manipulate timestamped data at various granularities, Sarda defines a set of operations on granularities. Instant-based data timestamped at a coarser granularity are converted to an interval at finer database-defined granularity. Compared to Sarda's approach, our model of events and of states explicitly represents the temporal uncertainty associated with timestamped data. For each interval-based datum, we represented *Un-*

certain Interval (*Certain Interval*) during which we are uncertain (certain) of the state's value. Sarda does not make clear whether such an interval exists in his temporal model. If the granularity of start and stop timepoints can be converted to any finer level, then no single-defined set of timepoints exists to bound the interval. Our representation for handling the imprecision of timestamped data, thus, differs significantly from Sarda's granularity-based representation, and overcomes his model's limitations in representing temporal uncertainty.

The standard set of relational operators does not form a proper algebra for temporal relations. Previous researchers have noticed the inadequacy of standard operators as an algebra for temporal relations, but few researchers have yet defined a closed temporal algebra for interval-stamped relations [11, 12, 14]. In this research, we created our own set of temporal algebraic operators; we defined a set that is closed for temporal relations and that is complete for supporting temporal operations. Our set of temporal algebraic operators differs from those proposed by other researchers [2, 6, 9, 19] for several reasons. First, we establish a set-theoretic notion for temporal relations that uses timestamping at the tuple level. Other researchers [2, 6] have argued for timestamping at the attribute level, but their data models do not maintain relations in first normal form. Second, we keep both events and states in a single type of interval-stamped temporal relation. Data models that define two types of temporal relations (one for events and one for states) do not have a unified set of algebraic operators for both types [9, 19]. Third, we develop a closed set of algebraic operators that incorporates the full range of temporal operations needed to query timestamped data in temporal relation.

In brief, our main contributions to research on temporal relational database are a general representation for events and states and a well-defined semantics for manipulating timestamped data.

The organization of this paper is as follows. In Section 2, we discussed the set of temporal operations that includes all basic comparisons and computations needed for interval-stamped data, regardless of associated temporal uncertainty. In section 3, we addressed our own set of temporal algebraic operators that is closed for temporal relations and that is complete for supporting temporal operations. In section 4, we presented the syntax of a temporal SQL, TSQL. And the data manipulation capabilities of our temporal operators are demonstrated with example queries. In section 5, we make concluding remarks and further research area is suggested.

2. Temporal Relational Model

The elementary set of data values used in data models are called value domains, denoted $U_D=\{D_1, D_2, \cdots, D_n\}$. Each D_i is a set of values which commonly include the set of integers and the set of character strings [18]. The temporal relational model requires a special type of domain, the time domain (TD).

2. 1. Representation and Operations of Time Domain

Time can be measured using a clock of suitable granularity. And time is usually discretized as a linearly ordered set of temporal elements, *timepoints or time instances*, on a timeline. The time domain consists of a set of timepoints. The timepoints have a granularity of the smallest time unit that is of interest in the database application. In this paper, minute is used as a granularity level but other level also can be used according to the application domains. The following notation is used to denote the values in the time domain: year/ month/ day hour:min. Two special timepoints, $-\infty$ and NOW, are also included in the time domain.

In the database design, the designer identifies entities, relationships, their identifiers, and attributes. Alternatively, the designer identifies objects (or object type) for modeling them as relations. Each object has a unique identifier and a set of functionally-dependent attributes. Our data model stores three different types of timestamped data: instance-based, interval-based, and time-invariant.

Definition 1 Instance-based data, events, are objects which prevail for only one time unit. An event occurs instantaneously and has no temporal duration.

When the temporal information associated with an event is collected, the user may be uncertain of the precise instant at which the event occurred. Our model allow the user give an approximate interval of time during which the event took place. Our model uses 1 day, 1440 minutes as a default value. Two timestamps are used to represent the lower and upper bounds of the closed ***uncertain interval(UI)*** during which the event occurred.

Example 1. Suppose that a user timestamped the event of an english-test result as April 2, 1994, and he is certain that the event has occurred between 2 PM and 4 PM. Then the ***UI*** representation of that event would have a start time of April 2, 1994 14:00 and an end time of April 2, 1994 15:59. If the user has no information about the occurrence of event, ***UI*** would be April 2, 1994 00:00 and an end time of April 2, 1994 23:59□

The lower and upper bounds of an *UI* can be equal if the user certainly knows the instant associated with an event in the minute level. Since this data model represents events as *UIs*, interval comparison methods are used to compare events with different intervals of uncertainty (interval comparison methods are explained in next section). The storage of **interval-based** and **time-invariant data** is based on combination of events.

Definition 2 Interval-based data, states, prevail over an interval of time, during which none of the attributes change their values. States are bounded by start and end events.

Consequently, both the start and end events of a state can be represented as ***uncertain intervals***. The closed interval between the upper bound of the start event and lower bound of the end event, the ***main*** state, represents a ***certain interval(CI)***. The value of the state holds true at each timestamp in the *CI*. Thus, the storage of a state requires three intervals in our model. Figure 1 shows events (a), and states (b, c).

Definition 3 Time-invariant data are defined to be true for the current and all past timestamps. They are stored as CIs, where the value for the start point is "$-\infty$", and that for the end point is "now".

Our modeling of uncertainty is comparable to the temporal representations used in two artificial intelligence systems, both of which utilize constraint satisfaction approaches to temporal reasoning: Console, et al. [3] and Dechter, et al. [5]. They model a variable interval as an interval whose start and end points are not perfectly known. The variable interval is represented by a pair of intervals: The first interval captures the uncertainty associated with the start point of the variable interval, whereas the second captures the uncertainty of the end point. Or the variables in the constraint-network are modeled to correspond to timepoints, which normally represent the start and end times of interval-based data. The values of the start and end times are constrained to a feasible interval of time with six different pieces of information: the earliest and latest possible start time, the earliest and latest possible end time, and the minimum and maximum possible duration. These representation is similar to our definition of start and end events as *UIs*, and hence the variable-interval representation is similar to our state representation. Although our database model does not have the temporal-reasoning capabilities present in either Console's or Decther's constraint-satisfaction approaches, we have rectified the inability of previous database models to support temporal uncertainty.

2. 2 Temporal Relation

Instant-based, interval-based, and time-invariant data are modeled as *UIs*, *CIs*, or a combination of both. An interval-stamping method is defined for storing the three types of temporal data within a single type of relational table. A **relation** in the standard model is any subset of the cartesian product of one or more domains: $D_1 \times D_2 \times \cdots \times D_n$ [18]. Each relation that is a subset of this product has a degree of n.

Definition 4 The temporal relation is defined to the subset of the Cartesian product that includes time domain T_S(start time) and T_E(end time): $D_1 \times D_2 \times \cdots \times D_n \times T_S \times T_E$.

The temporal relation is represented as a traditional table form, like a standard relation. Two columns TS and TE correspond to the T_S and T_E, are called timestamp attributes. The degree of the temporal relation is k+2. Each row or tuple, of the table thus contains an interval that represents the time period during which the values of the nontemporal attributes, the columns (A_1, A_2, $\cdots$, A_n), are valid. With this interval stamping method, we can store the *UI* of an event within a single tuple, whereas three tuples are needed to store the state: two for the pair of *UIs* of the start and end events and one for the single *CI* of the main. To distinguish these types of interval, two nontemporal attributes are defined: Type and Interval. Type is used to maintain the type of interval being stored, the values of which can be event, start, main, or end. Interval stores the length of an *UI* for maintenance of temporal uncertainty.

In the following definitions, R_1 and R_2 are relations having the following relation schemes: R_1(TS, TE, A_{M1}, A_{M2}, $\cdots$, A_{MQ}), R_2(TS, TE, A_{M1}, A_{M2}, $\cdots$, A_{NR}), where A_{M1}, A_{M2}, $\cdots$ are non-temporal attributes. The degree of R_1 is Q+2 and that of R_2 is R+2. The variable T(R_i) corresponds to the set of temporal intervals in relation R_i.

Example 2. Consider the following 3 relation schemes;
EMPLOYEE_TEST(EmpNo, Test, Score, TS, TE, Interval, Type),
EMPLOYEE(EmpNo, Position, Salary, TS, TE, Interval, Type),

PROJECT(EmpNo, Project, TS, TE, Interval, Type).

In the EMPLOYEE_TEST relation, EmpNo is the number of employee, and Score is the result of Test. TS and TE represent start time and end time of an event, respectively. Interval is the length of *UI* and Type is used to denote whether the tuple is *event* in the case of event, or *start*, *main*, or *end* in the case of interval-based state. TS, TE are temporal attributes and the others are non-temporal relations. And the degree of relation EMPLOYEE_TEST is 7.□

Figure 1 shows tables, temporal relations of example 2. They store instant- and interval-based data. Figure 1(a) represents Employee_Test table, temporal relation of events, (b) represents Employee table, temporal relation of states, and (c) represents Project table, temporal relation of states.

EmpNo	Test	Score	TS	TE	Interval	Type
33	English1	890	94/ 04/ 02 00:00	94/ 04/ 02 23:59	1440	event
33	English1	780	94/ 08/ 17 00:00	94/ 08/ 17 23:59	1440	event
33	English1	940	94/ 12/ 02 00:00	94/ 12/ 02 23:59	1440	event
33	computing	460	94/ 08/ 17 10:00	94/ 08/ 17 11:59	120	event
33	computing	440	94/ 08/ 17 14:00	94/ 08/ 17 15:59	120	event

(a)

EmpNo	Position	Salary	TS	TE	Interval	Type
33	Jr. Engineer	1200	94/ 04/ 02 00:00	94/ 04/ 02 23:59	1440	start
33	Jr. Engineer	1200	94/ 04/ 03 00:00	94/ 08/ 15 23:59	null	main
33	Jr. Engineer	1200	94/ 08/ 16 00:00	94/ 08/ 16 23:59	1440	end
33	Jr. Engineer	1400	94/ 08/ 17 00:00	94/ 08/ 17 23:59	1440	start
33	Jr. Engineer	1400	94/ 08/ 18 00:00	94/ 12/ 01 23:59	null	main
33	Jr. Engineer	1400	94/ 12/ 02 00:00	94/ 12/ 02 23:59	1440	end

(b)

EmpNo	Project	TS	TE	Interval	Type
33	KPC94_12	94/ 05/ 01 00:00	94/ 05/ 01 23:59	1440	start
33	KPC94_12	94/ 05/ 02 00:00	NOW	null	main

(c)

Figure 1. Temporal relation for events and states.

The following methods are used to enforce ***temporal normalization*** of the temporal relation: for any particular timepoint, each attribute of an object in the database have a single or null value, which will result in the standard first normal form. And the pair of timepoint attributes are always included as part of the key attributes for the relation. To make the values of the timepoints a meaningful interval for an event or for a state, the value of TS either precedes or is equal to the value of the TE. Please refer Navathe and Ahmed [12] for the definition temporal normalization.

2. 3 Temporal Operations

All timestamps in our data model have the same granularity, the finest level of interest in the database application. Consequently, discrete timestamps can be comparable to determine a linear order using the standard comparison operations. Temporal comparisons of intervals play an important role in manipulating both instant-based data and interval-based data. Since intervals are pairs of timepoints, comparisons between intervals are based on timepoint comparisons of the lower and upper bounds; a complex set of comparisons can be made between two pairs of timepoints. The terms AT, BEFORE, and AFTER are used to indicate the comparison operators $=$, $<$, and $>$, respectively. The pair of timepoints, $[t_1, t_2]$ are used to represent a closed interval T. With this set, we can also make the mixed timepoint-interval comparisons of a timepoint BEFORE an interval, AFTER an interval or DURING an interval by representing the timepoint as a zero-length interval. Please refer Navathe and Ahmed [12] about interval comparison operators.

We define many operators and built-in functions on instants and intervals:

Definition 5 Instant Operators

a. INTERVAL(t_1, t_2) returns the number of units between the timestamps t_1 and t_2.

b. ADDTIME(t, x) adds x units to timestamp t to create a new timestamp. When $x=1$, the function returns the next timestamp: when $x=-1$, the function returns the previous timestamps.

c. MIN(t_1, t_2) returns the value t_1 if $t_1 <= t_2$, and returns the value t_2 if $t_1 > t_2$. MAX(t_1, t_2) is defined vise verse.

Given any two intervals, T[t_1, t_2] and U[u_1, u_2], we define three different binary operations to compute the union, difference, or intersection of the intervals.

Definition 6 Interval Operators

a. CONCATENATE ($\oplus$) takes two overlapping (or adjacent) intervals and returns a new interval that consists of the set of timepoints in either of the original intervals:

   $T \oplus U = [\mathrm{MIN}(t_1, u_1), \mathrm{MAX}(t_2, u_2)]$.

   If the operand intervals in CONCATENATE operation do not overlap or are not adjacent, no new interval is created.

b. SUBTRACT ($\ominus$) takes two intervals and removes from the first interval the points common to both to create a new interval:

   $T \ominus U = [t_1, \mathrm{ADDTIME}(u_1, -1)]$, when $t_1 < u_1$,

   $= [\mathrm{ADDTIME}(u_2, 1), t_2]$, when $t_2 > u_2$.

   $= [t_1, t_2]$, if the intervals do not overlap.

c. EXTRACT($\otimes$) takes two intervals, but returns a new interval containing only those points that are contained in both intervals:

   $T \otimes U = [\mathrm{MAX}(t_1, u_1), \mathrm{MIN}(t_2, u_2)]$, if the intervals overlap,

   $=$ NULL, if the intervals do not overlap.

We can apply these interval computations to any interval in the temporal relational data model. The pair of intervals used in these three binary operations, therefore, can be either both of type UIs, both of type CIs, or a pair of each type.

3. Temporal Relational Algebra

The algebraic operations form a convenient basis for defining, understanding, translation, optimization and execution of query language. The standard relational model defines the following as primitive operators: Union($\cup$), Set difference($-$), Cartesian product(X), Projection(Π), and Selection(σ). Additional, and more useful, operations, such as join and intersection, can be derived from these [18]. Although the five primitive operators are considered to be closed for a relational query language, two of them (Cartesian product and Projection) can not be applied directly to temporal relations. The Cartesian product of two temporal relations does not result in another temporal relation, as each tuple in the resultant relation contains two sets of intervals. The projection of a temporal relation is a temporal relation only when temporal attributes, TS and TE are part of the attributes specified in the projection list.

We define a new set of operators that form a proper algebra for temporal relations. Using the methods to handle overlapping or adjacent intervals, we modified the definition of the standard operations to handle temporal operations.

Definition 7 Temporal projection

Temporal projection returns the temporal attributes as well as the list of nontemporal attributes to be projected:

$R_2 = \Pi_{t(A_{N1},\cdots,A_{Ni})} R_i$, where Π_t represents temporal projection, $(A_{N1},\cdots,A_{Ni})$ is the list of nontemporal attributes to be projected. As a result, R_2 has the scheme $R_2(TS, TE, A_{N1}, A_{N2}, \cdots, A_{Ni})$.

Standard selection produces a horizontal subset of a given relation. We use the standard selection operator with its set of nontemporal predicate formulas, F. We add to this set the set of temporal comparisons Γ as conditions to restrict tuples.

Definition 8 Temporal selection

Temporal selection is defined as:

$R_2 = \sigma_F R_1$, where R_2 has the scheme $R_2(TS, TE, A_{N1}, A_{N2}, \cdots, A_{NK})$, Q=R and the set T ($R_2$) is equal to a subset of T(R_1) that satisfy the specified temporal predicate.

We use the standard union operator and difference operator, since the result remains a temporal relation.

Definition 9 Concatenate

Concatenate (K) is a unary operator that combines those tuples that have the same nontemporal attribute values, but overlapping or adjoining intervals, into a single tuple with a resultant interval created from concatenation of the operand intervals.

The prerequisite conditions of value-equivalent nontemporal attributes with adjacent interval timestamps can occur from the application of temporal projection. Concatenate operator does not have a nontemporal equivalent in the standard relational algebra. The concatenate operation has been defined previously by Navathe and Ahmed [12] as the *compress* operation and by Sarda [14] as the *coalesce* operation.

The Join operation is used to combine related tuples from two relations into single tuples. This operation is very important for any relational database with more than a single relation, because it allows us to process relationships among relations. Join is defined as a combination of the Selection and the Cartesian product. Cartesian product can not be applied directly to temporal relations, so standard join should not be included in a sound temporal relational algebra. Instead, we define three elementary temporal joins and two composite joins that can combine timestamped data from two different relations and maintain the temporal attributes as part of the keys for the resulting tuples. The variable $T(R_i)$ represents the set of temporal intervals in relation R_i.

Definition 10 Temporal semijoin

Temporal semijoin returns the attributes of only the first operand. The set of temporal comparisons is used in the Boolean comparison, denoted symbolically as Γ, to restrict the results of the temporal semijoin. The operator is defined as:

$$R_3 = R_1 \rtimes_{t(\Gamma)} R_2,$$

where R_3 has the scheme $R_3(TS, TE, A_{N1}, A_{N2}, \cdots, A_{NQ})$, and the set $T(R_3)$ is equal to the results of restricting the set $T(R_1)$ with the Boolean comparisons condition Γ.

Definition 11 Extract join

The extract join results the extraction of overlapping intervals, and take the nontemporal attributes from both operand relations. The definition of extract join is

$$R_3 = R_1 \bowtie_{t(\otimes)} R_2,$$

where R_3 has the scheme $R_3(TS, TE, A_{N1}, A_{N2}, \cdots, A_{NS})$, $S = Q+R$, and the set $T(R_3)$ is equal to the resulting set of $T(R_1) \otimes T(R_2)$

This type of temporal join has also been proposed by Sarda [14] as the *concurrent product* and by Navathe and Ahmed [12] as the *temporal join.*

Definition 12 Subtract join

Subtract join is a counterpart to the extract join. The intervals in the resulting relation are the intervals in the first relation subtracted from any overlapping intervals in the second relation. As with the extract join, the nontemporal attributes are taken from both operand relations. The values of the nontemporal attributes from the second relation are, as a consequence, null in the resulting relation. The subtract join is defined as;

$$R_3 = R_1 \bowtie_{t(\ominus)} R_2,$$

where R_3 has the scheme $R_3(TS, TE, A_{N1}, A_{N2}, \cdots, A_{NS})$, $S = Q+R$, and the set $T(R_3)$ is equal to the resulting set of $T(R_1) \ominus T(R_2)$.

With these three elementary joins, we can also define two additional joins: temporal outer join and temporal theta join. The relationship between the temporal outer join and extract join is the same as that between the standard outer join and equijoin.

Definition 13 Temporal outer join

The temporal outer join returns the same tuples as does the extract join, along with tuples representing the time periods when values from only one of the operand relations exist. These ***dangling*** time periods are exactly those that we obtain by taking the subtract join of the operand relations twice with the order of the operands alternated. Temporal outer join is defined as;

$$R_3 = R_1 ⟗_{t(\bigcirc)} R_2,$$

where R_2 has the scheme $R_3(TS, TE, A_{M1}, A_{N}, \cdots, A_{NS})$, $S = Q+R$, and the set $T(R_3)$ is equal to the union of the resulting set of $T(R_2 \ominus T(R_1)$, and the resulting set of $T(R_1) \otimes T(R_2)$

Definition 14 Temporal theta join

Temporal theta join is the temporal equivalent of the theta join in standard relational algebra, where theta (θ) is a set of relational operators(⟨,⟩,=,⟨=,⟩=,⟨⟩). Instead of restriction to nontemporal θ conditions, the resulting tuples in the operand relation satisfy the temporal comparison Γ. The attributes in the temporal theta join are from both operands. Temporal theta join is defined as:

$$R_3 = R_1 \bowtie_{t(\Gamma)} R_2,$$

where R_3 has the scheme $R_3(TS, TE, A_{M1}, A_{N}, \cdots, A_{NS})$, $S = Q+R$, and the set $T(R_3)$ is equal to subset of the set of intervals in the temporal outer join of the same operands.

The elementary set of operations for the temporal relational data model thus consists of temporal semijoin, extract join, subtract join, temporal projection, selection, union, difference, and concatenate. A requirement for a ***proper algebra*** is a set of operations on objects the results of which are themselves the same type of object. Since the application of each temporal operation on histories returns a history as a result, we have, consequently, defined a closed set of operators for a temporal relational algebra.

Proposition 1 Temporal semijoin can be defined using only the set of standard operators.

(proof)

In terms of the standard operators, it is the projection onto the attributes of the first operand of a join that is based on the temporal comparison Γ by definition 8:

$R_1 \ltimes_{t(\Gamma)} R_2 = \Pi_{(R_1.TS,\ R_1.TE,\ R_1.A_{N1}, \cdots, R_1.A_{NQ})}\ \sigma_{\Gamma}(R_1 \times R_2)$. □

The standard semijoin is defined as the projection of the attributes of the first operand from the equijoin on attributes that the two operand relations share in common [18]. The temporal semijoin, on the other hand, is one of the basic set of operators in the temporal algebra. Note that the operation $R_1 \ltimes_{t(\Gamma)} R_2$ is generally not equivalent to the operation $R_2 \ltimes_{t(\Gamma)} R_1$.

Unlike the temporal semijoin, the extract join returns both sets of nontemporal attributes with a single pair of timepoint attributes that are derived from the EXTRACT operation.

Proposition 2 Extract join can be defined using only the set of standard operators.
(proof)

According to definition 9, four possible pairs of timepoints can be returned based on which is the maximum of the first timepoint values and which is the minimum of the second timepoint values. We undertake a separate join for each of these possibilities; we then take the union of the join results to obtain the final result, as follows:

$$
\begin{aligned}
R_1 \bowtie_{t(\otimes)} R_2 = {} & (\Pi_{R_2.TS,\ R_1.TE,\ R_1.A_1, \cdots, R_1.A_{NQ},\ R_2.A_1, \cdots, R_2.A_{NR})} \\
& \quad \sigma_{(R_1.TS < R_2.TS,\ R_1.TE \geq R_2.TS,\ R_1.TS \leq R_2.TE)}\ (R_1 \times R_2)) \\
& \cup (\Pi(_{R_1.TS,\ R_1.TE,\ R_1.A_1, \cdots, R1.A_{NQ},\ R_2.A_1, \cdots, R_2.A_{NR}} \\
& \quad \sigma_{(R_1.TS \geq R_2.TS,\ R_1.TE \leq R_2.TE,\ R_1.TE\ R_2.TE)}\ (R_1 \times R_2)) \\
& \cup (\Pi(_{R_1.TS,\ R_1.TE,\ R_1.A_1, \cdots, R1.A_{NQ},\ R_2.A_1, \cdots, R_2.A_{NR}} \\
& \quad \sigma_{(R_2.TS \geq R_1.TS,\ R_2.TE \leq R_1.TE)}\ (R_1 \times R_2)) \\
& \cup (\Pi(_{R_2.TS,\ R_2.TE,\ R_1.A_{N1}, \cdots, R1.A_{NQ},\ R_2.A_{N1}, \cdots, R_2.A_{NR})} \\
& \quad \sigma_{(R_1.TS < R_2.TS,\ R_1.TE > R_2.TE)}\ (R_1 \times R_2)) \ \square
\end{aligned}
$$

The subtract join returns both sets of nontemporal attributes from the operands with a single pair of timestamp attributes. The values of the attributes from the second operand relation may be null. However, the necessity for such a result is apparent in the incorporation of the subtract join in the definition of the temporal outer join.

Proposition 3 Subtract join can be defined using only the set of standard operators.
(proof)

The interval timestamps in the result of subtract join can be derived by applying the SUB

TRACT operation to the join of the operands. According to definition 6(b), three possibilities exist for which timepoints to return: (1) the second interval's starting point is during the first interval, (2) the second interval's ending point is during the first interval, or (3) the first interval does not overlap with the second. With the standard operators, we first undertake a separate join for each of these possibilities; we next project the appropriate timepoint and the attributes of the first relation; and we then take the union of the three join results. To obtain the final result, we take the Cartesian product of the resulting union and of a tuple of null values that corresponds to the attributes of the second operand relation:

$$\begin{aligned} R_1 \bowtie_{t(\ominus)} R_2 = & ((\Pi_{(R_1.TS,\ \mathrm{ADDTIME}(R_2.TS,\ -1),\ R_1.A_{N1},\ \cdots,\ R_1.A_{NQ})} \\ & \quad \sigma_{(R_2.TS > R_1.TS,\ R_2.TS \le R_1.TE)} (R_1 \times R_2)) \\ & \cup (\Pi_{(\mathrm{ADDTIME}(R_2.TE,\ 1),\ R_1.TS,\ R_1.A_{N1},\ \cdots,\ R_1.A_{NQ})} \\ & \quad \sigma_{(R_2.TE \ge R_1.TS,\ R_2.TE < R_1.TE)} (R_1 \times R_2)) \\ & \cup (\Pi_{(R_1.TS,\ R_1.TE,\ R_1.A_{N1},\ \cdots,\ R_1.A_{NQ})} \\ & \quad \sigma_{(R_1.TE < R_2.TS\ OR\ R_1.TS > R_2.TE)} (R_1 \times R_2))) \times \langle A_1, \cdots, A_{NR} \rangle, \text{ where} \end{aligned}$$

$\langle A_1, \cdots, A_{NR} \rangle$ is a null values of R_2.□

Like the temporal semijoin, the subtract join is not symmetric.

Proposition 4 The temporal outer join can be defined using only the set of standard operators.

(proof)

The temporal outer join of two relations is defined as the union of the extract join of the relations with the pair of subtract joins of the relations:

$$R_1 \bowtie_{t(\bigcirc)} R_2 = (R_1 \bowtie_{t(\otimes)} R_2) \cup (R_1 \bowtie_{t(\ominus)} R_2) \cup (R_2 \bowtie_{t(\ominus)} R_1).$$

The union of these three temporal joins is feasible, since the degrees of the extract and subtract joins are equivalent.□

Proposition 5 The temporal theta join can be defined using only the set of standard operators.

(proof)

The temporal theta join can be defined from the set of temporal operators only, as follows:

$$R_1 \bowtie_{t(\Gamma)} R_2 = (R_1 \ltimes_{t(\Gamma)} R_2) \bowtie_{t(\bigcirc)} (R_2 \ltimes_{t(\Gamma)} R_1).\square$$

Proposition 6 Temporal projection can be defined using only the set of standard operators.

(proof)

Temporal projection is a simple modification of standard projection. We add the timestamp attributes of the list of projected attributes to obtain the result:

$$\Pi_{t(A_{N1}, \cdots, A_{Nj})} R_1 = \Pi_{(R_1.TS,\ R_1.TE,\ A_{N1}, \cdots, A_{Nq})} R_1. \Box$$

The selection, the union, and the difference operators in our temporal relational algebra are unmodified from the standard set. The concatenate operator cannot be derived from the set of standard operators; thus, it is a primitive operator for the temporal relational data model.

4. A Temporal SQL and Example Queries

Defining a set of algebraic operations for a temporal relational data model is useful for understanding and executing a temporal extension to SQL. Since the temporal relational data model is defined to maintain the underlying view of temporal data as timepoints on a timeline, we call the query language **Temporal SQL (TSQL)**. SQL meets the criteria for completeness as a relational query language, since it supports all operations in standard relational algebra [18]. TSQL is also complete as a temporal query language for algebraic operations on temporal relations. In this section, we defined syntax of TSQL only. A sequence of operations to execute a query is beyond the scope of this research and refer other researchers [4, 12, 14]. Next, we illustrate example temporal queries on the set of temporal relations in Figure 1 with temporal relational algebra and TSQL.

4. 1 Temporal Query Syntax

TSQL is based on the simple structural framework of SQL, with syntactic extensions to support the set of temporal relational algebra. TSQL adds the following new constructs to standard SQL.

1. Selections based on temporal comparisons, using a WHEN clause.
2. Selections based on ordinal ordering, using the terms FIRST, SECOND, THIRD, or LAST in the SELECT clause.
3. Methods to perform concatenation of overlapping or adjoining intervals with the same

nontimestamp values, using the term CONCATENATED in the SELECT clause.

4. Methods to perform the extract or subtract operation on intervals in two different relations, using the terms CONCURRENT WITH (for extract) or NOT CONCURRENT WITH (for subtract) in the WHEN clause.

4. 2 Queries with TSQL and temporal relational algebra

The set of temporal relational algebra that we have outlined in Section 3 provides the capabilities for temporal querying of timestamped data. We illustrate the expressiveness of the temporal relational algebra with example temporal queries on the set of temporal relations in Figure 1. With the mapping between the syntax of a temporal SQL and the semantics of temporal relational algebra, we can write an equivalent TSQL query for each example query. These four example queries illustrate how the application of our operators to temporal relations by (a) temporal relational algebra, by (b) TSQL, and by (c) temporal relation as a table.

Q1. During what time periods was employee 33 a junior engineer receiving 1200 or 1400?

(a) In temporal relational algebra, this query translates to the following:

$$K(\Pi_{t(EmpNo,\ Position)}\ \sigma_{(EmpNo\ =\ 33\ \wedge\ Position\ =\ \text{“Jr. Engineer”}\ \wedge\ (Salary\ =\ \text{“1200”}\ \vee\ Salary\ =\ \text{“1400”}))}\ \text{EMPLOYEE}).$$

(b)
```
SELECT CONCATENATED EmpNo, Position
FROM EMPLOYEE
WHERE EmpNo = 33 AND Position = “Jr. Engineer” AND
        (Salary = “1200” OR Salary = “1400”)
```

(c) The temporal relation that results from application of these operators to the EMPLOYEE table follows:

EmpNo	Position	TS	TE
33	Jr. Engineer	94/0?/02 00:00	94/12/02 23:59

Q2. What are the values of any employee-tests performed during the day of August 17, 1994 for employee 33?

(a) $\sigma_{(EmpNo = 33 \wedge ([TS, TE] \ DURING \ [94/08/17\ 00:00, 94/08/17\ 23:59]) \vee ([TS, TE] \ EQUALS \ [94/08/17\ 00:00, 94/08/17\ 23:59]))}$ EMPLOYEE_TEST.

(b) SELECT EmpNo, Test, Score, Interval, TYPE

FROM EMPLOYEE_TEST

WHEN ([TS, TE] DURING [94/ 08/ 17 00:00, 94/ 08/ 17 23:59]) OR ([TS, TE] EQUALS [94/ 08/ 17 00:00, 94/ 08/ 17 23:59])

WHERE EmpNo = 33

(c)

EmpNo	Test	Score	TS	TE	Interval	Type
33	English1	780	94/ 08/ 17 00:00	94/ 08/ 17 23:59	1440	event
33	computing	460	94/ 08/ 17 10:00	94/ 08/ 17 12:00	120	event
33	computing	440	94/ 08/ 17 14:00	94/ 08/ 17 16:00	120	event

Q3: What are the scores of any english1 test performed before employee 33 was junior engineer with 1400 as a salary?

(a) $\sigma_{(Emp\_No = 33 \wedge Test = \text{"English1"})}$(EMPLOYEE_TEST $\ltimes_{t([EMPLOYEE\_TEST.TS, EMPLOYEE\_TEST.TE] \ BEFORE \ [EMPLOYEE.TS, EMPLOYEE.TE])}$ $\sigma_{(Emp\_No = 33 \wedge Position = \text{"Jr. Engineer"} \wedge (Salary = \text{"1400"}))}$ EMPLOYEE).

(b) SELECT EMPLOYEE_TEST. EmpNo, EMPLOYEE_TEST. Test, EMPLOYEE_TEST.Score,
EMPLOYEE_TEST. Interval, EMPLOYEE_TEST. TYPE

FROM EMPLOYEE_TEST, EMPLOYEE

WHEN [EMPLOYEE_TEST. TS, EMPLOYEE_TEST. TE] BEFORE
[EMPLOYEE. TS, EMPLOYEE. TE]

WHERE EMPLOYEE_TEST. EmpNo = 33 AND EMPLOYEE. EmpNo =33

(c)

EmpNo	Position	Score	TS	TE	Interval	Type
33	Jr. Engineer	890	94/ 04/ 02 00:00	94/ 04/ 02 23:59	1440	event

Q4: During what time periods was employee 33 either participating in project KPC94_12 or junior engineer?

(a) K($\Pi_{t(PROJECT.EmpNo, PROJECT.Project, EMPLOYEE.Position, EMPLOYEE.Salary)}$ $\sigma_{(PROJECT.EmpNo = 33 \wedge PROJECT.EmpNo = EMPLOYEE.EmpNo)}$ (PROJECT $\bowtie_{t(\bigcirc)}$ EMPLOYEE)).

(b)
```
SELECT CONCATENATED PROJECT. EmpNo, PROJECT. Project, EMPLOYEE. Position,
         EMPLOYEE. Salary
FROM PROJECT, EMPLOYEE
WHERE PROJECT. EmpNo = 33 AND PROJECT. EmpNo = EMPLOYEE. EmpNo
```

(c)

EmpNo	Project	Position	Salary	TS	TE
33	KPC94_12	null	null	94/ 02/ 01 00:00	94/ 04/ 01 23:59
33	KPC94_12	Jr. Engineer	1200	94/ 04/ 02 00:00	94/ 08/ 16 23:59
33	KPC94_12	Jr. Engineer	1400	94/ 08/ 17 00:00	94/ 12/ 02 23:59
33	KPC94_12	null	null	94/ 12/ 03 00:00	NOW

5. Conclusions

In this paper, we have examined two aspects of temporal relational databases that had not been addressed by other researches: (1) a time-stamping method for instant-based events and for interval-based states that can incorporate temporal uncertainty, and (2) a set of algebraic operators for interval-stamped relations that is closed.

The set of temporal operations is discussed that includes all basic comparisons and computations needed for interval-stamped data, regardless of associated temporal uncertainty. A set of temporal algebraic operators are defined that is closed for temporal relations and that is complete for supporting temporal operations. These operators give special consideration to the timestamp attributes. Our set of temporal algebraic operators differs from those proposed by other researchers for several aspects.

The main contributions of this research can be stated as a general representation for events and states and a well-defined semantics for manipulating timestamped data in the temporal relational database. The added complexity of the temporal representations and operations underlying TSQL will be necessary. But it is expected to be relatively easy compared to other previous researches.

Our temporal database provides a relational representation of temporal uncertainty that is suitable for long-term storage of event and states. So it is a promising further research area to develop a temporal reasoning methods based on our representation of temporal uncertainty. And it is also a further research area to integrate a temporal relational database with a knowledge-base for a specific area.

References

[1] Allen, J. F., "Maintaining knowledge about temporal intervals," *Communications of ACM*, Vol. 26 (1983), pp. 832-843.

[2] Clifford, J. and A. U. Tansel, "On an algebra for historical databases: Two views," In *Proceedings of the ACM SIGMOD Conference*, Austin, TX, May 1985, pp. 247-265.

[3] Console, L., A. Furno and P. Torasso, "Dealing with time in diagnostic reasoning based on causal models," In *Methodologies for Intelligent System*, Vol. 3 (Ras, Z. W., Saitta, L. Eds), Amsterdam: North Holland, 1988.

[4] Das, A. K., S. W. Tu, G. P. Purcell and M. A. Musen, "An extened SQL for temporal data management in clinical decision-support systems," In *Proceedings of the Sixteenth Annual SCAMC*, Baltimore, MD, November 1992, pp. 128-132.

[5] Dechter R., Meiri, I. and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, Vol. 49 (1991), pp. 61-95.

[6] Gadia, S. K. and C. S. Yeung, "Inadequacy of interval timestamps in temporal databases," *Information Sciences*, Vol. 54 (1991), pp. 1-22.

[7] Koubarakis, M., "Database Models for Infinite and Indefinite Temporal Information," *Information Systems*, Vol. 19 (1994), pp. 141-173.

[8] Lai, V. S., J. P. Kuiboer and J. L. Guynes, "Temporal Databases: Model Design and Commercialization Prospects," *DATABASE* Vol. 25 (1994), pp. 6-18.

[9] Lorentzos, N. and R. Johnson, "Extending relational algebra to manipulate temporal data," *Information Systems*, Vol. 13 (1988), pp. 289-296.

[10] Maiocchi, R. and B. Pernici, "Temporal data management systems: A comparative view," *IEEE Trans. on Knowledge and Data Eng.*, Vol. 3 (1991), pp. 504-523.

[11] McKenzie, L. E. and R. T. Snodgrass, "Evaluation of relational algebra incorporating the time dimension in databases," *ACM Computing Survey*, Vol. 23 (1991), pp. 501-543.

[12] Navathe, S. B. and R. Ahmed, "A temporal relational model and a query language," *Information Sciences*, Vol. 49 (1989), pp. 147-175

[13] Roddick, J. F. and J. D. Patrick, "Temporal Semantics in Information Systems-A Survey," *Information Systems*, Vol. 17 (1992), pp. 249-267.

[14] Sarda, N. L., "Extensions to SQL for historical databases," *IEEE Trans. on Knowledge and Data Eng.* Vol. 2 (1990), pp. 220-230.

[15] Segev, A. and A. Shoshani, "The representation of a temporal data model in the relational environment," In *Lecture Notes in Computer Science*, vol. 339, New york, Springer-Verlag

(1988), pp. 39-61,

[16] Shoham, Y., "Temporal logic in AI: Semantical and ontological considerations," *Artificial Intelligence*, Vol. 33 (1987), pp. 89-104.

[17] Snodgrass, R. and I. Ahn, "Temporal database," *IEEE Computer*, Vol. 19 (1986), pp. 35-42.

[18] Ullman, J. D., *Principles of Database and Knowledge-Base Systems*, vol. 1, Rockville, MD: Computer science Press, 1988.

[19] Wiederhold, G., S. Jajodia, and W. Litwin "Dealing with granularity of time in temporal databases," In *Lecture Notes in Computer Science*, vol. 398, New York, Springer-Verlag (1991), pp. 124-140.

[20] Wiederhold, G., *Semantic Database Design* McGraw-Hill, New York, 1994.