A GENERALIZED MODEL FOR A RELATIONAL TEMPORAL DATABASE

Shashi K. Gadia
Computer Science Department
Iowa State University
Ames, IA 50011

Chuen-Sing Yeung
Intellution Inc.
220 Norwood Park South
Norwood, MA 02062

Abstract: We propose a generalized relational model for a temporal database which allows time stamping with respect to a Boolean algebra of multidimensional time stamps. The interplay between the various temporal dimensions is symmetric. As an application, a two dimensional model which allows objects with real world and transaction oriented time stamps is discussed. The two dimensional model can be used to query the past states of the database. It can also be used to give a precise classification of the errors and updates in a database, and is a promising approach for querying these errors and updates.

1. INTRODUCTION.

In this paper we generalize several existing approaches to relational temporal databases. Our model allows multidimensional time stamps, treating all the individual temporal dimensions symmetrically. We also present an algebra which exploits this symmetry to allow a considerable interplay between the various temporal dimensions. The algebra allows temporal and Boolean expressions for navigation within a tuple. At run time, when a tuple τ is substituted into a temporal expression m [resp. a Boolean expression f], the result m(τ) [resp. $f(\tau)$] is a time domain [resp. true of false]. The set of relational expressions (expressions which evaluate to relations), temporal expressions and Boolean expressions are defined by mutual recursion and they are very powerful. We introduce selections and joins of the form σ(r,[m;X]), and r[m;X]s, respectively, which allow X part of every tuple in the resulting relation to be restricted to the time domain which is obtained by substituting tuples from r and s in m. A two dimensional case of our model has an interesting application to databases: it can be used to give a precise semantics of errors and updates in the evolution of a database, and the algebra allows a user to query these errors and updates.

The paper is organized as follows. In Section 2 we present a basic model allowing a single time dimension. In Section 3 we give an algebra for this model. In Section 4 we generalize our model and the algebra of sections 2 and 3 to allow multidimensional time stamps. In Section 5 we consider the two dimensional case of our model. In Section 6 we discuss other related approaches to temporal databases. In Section 7 we discuss weak identity operators which arise naturally in temporal databases. In fact our view is that these operators form an integral part of an algebra; we do not present them in sections 2 and 3 for the ease of reading. Our organization necessitates a change in notation which is stated in Section 4.4.

2. THE BASIC MODEL FOR LOCAL NAVIGATION.

Although a tuple in a temporal database can be a very large object, time imposes a considerable structure within it. A salient feature of our model is that it allows navigation within a tuple to exploit this structure, which also reduces number of joins. We term navigation within a tuple as *local navigation.* We view local navigation to be of critical importance in temporal databases.

2.1 A Structure for time stamps. We use the concept of time introduced in [Ga1]. We assume that a universe of time instants [0,now], where now denotes the current time, together with a linear order $\leq$ is given. Intervals are not closed under union, intersection and complementation, and thus are not adequate to model natural language queries

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1988 ACM 0-89791-268-3/88/0006/0251 $1.50

involving "or", "and" and "not". A *temporal element* is a finite union of intervals in [0,now]. An interval in [0,now] is obviously a temporal element. An instant t may be regarded as a temporal element by identifying it with the interval [t,t]. We use the variables μ, ν, μ_1, ν_1, ... to denote temporal elements. The union and intersection of μ and ν are denoted as $\mu+\nu$ and $\mu*\nu$, respectively, and complement of μ (with respect to the universe [0,now]) is denoted as $-\mu$. The set of all temporal elements, denoted *TE*, is a Boolean algebra under +, * and −, with $\emptyset$ and [0,now] as its minimum and maximum elements. In this paper we assume that [0,now] consists of equidistant instants 0, 1, 2, ..., now; this assumption has been made only for the sake of simplicity and in most situations it can be eliminated along the lines of [Gal].

2.2. Navigation in time. Let μ and ν be temporal elements. $\mu<\nu$ (resp. $\mu\leq\nu$) means for every $t_1\in\mu$ and $t_2\in\nu$, $t_1<t_2$ (resp. $t_1\leq t_2$). We allow set comparisons $\subseteq$ and = among the temporal elements. As in [GV] we define $fi(\mu)$ = the first time instant in μ, $\ell i(\mu)$ = the last time instant in μ and $\mu-\nu$ = {t: $t\in\mu$ and $t\notin\nu$}. (Note that − is overloaded: it is used as a unary, as well as a binary operator.) An interval $I\subseteq\mu$ is a *maximal interval in* μ, if for all intervals $J\subseteq\mu$, $I\cap J\neq\emptyset$ implies $J\subseteq I$. Now, $fI(\mu)$ is defined to be the first maximal interval in μ and , $\ell I(\mu)$ the last maximal interval in μ.

2.3. Assignments. To capture time variant properties of objects, we introduce the notion of a temporal assignment [Gal]. A *temporal assignment* (or simply an *assignment*) ξ to an attribute A with a temporal element $\nu\subseteq$ [0,now] as its *temporal domain* is a function from ν, such that for each $t\in\nu$, $\xi(t)$ is an element of dom(A). We denote temporal domain of an assignment ξ as $\llbracket\xi\rrbracket$. If ξ is a temporal assignment and ν is a temporal element, then $\xi\upharpoonright\nu$ denotes the restriction of ξ to $\nu*\llbracket\xi\rrbracket$ as a function; thus $\xi\upharpoonright\nu$ is also a temporal assignment.

Two temporal assignments ξ_1 and ξ_2 are said to *agree* if (i) they are assignments to the same attribute, and (ii) $\forall t\in\llbracket\xi_1\rrbracket*\llbracket\xi_2\rrbracket$, $\xi_1\upharpoonright t=\xi_2\upharpoonright t$. If ξ_1 and ξ_2 agree, then $\xi_1\cup\xi_2$ is the common extension of ξ_1 and ξ_2 on $\llbracket\xi_1\rrbracket+\llbracket\xi_2\rrbracket$; if ξ_1 and ξ_2 do not agree, $\xi_1\cup\xi_2$ is undefined. Suppose $\mu=\{t\in\llbracket\xi_1\rrbracket*\llbracket\xi_2\rrbracket: \xi_1(t)=\xi_2(t)\}$, then we define $\xi_1\cap\xi_2$ to be $\xi_1\upharpoonright\mu$ (or $\xi_2\upharpoonright\mu$), and $\xi_1-\xi_2$ to be $\xi_1\upharpoonright(\llbracket\xi_1\rrbracket-\mu)$. We define shift($\xi$) to be the assignment obtained by adding one to every instant in $\llbracket\xi\rrbracket$. We may denote a temporal assignment ξ as $\langle\nu_1\mapsto a_1, ..., \nu_\ell\mapsto a_\ell\rangle$, where ν_1, ..., ν_ℓ are temporal elements, $\xi(t)=a_i$ if $t\in\nu_i$, $1\leq i\leq\ell$, such that $a_i\neq a_j$ if $i\neq j$.

2.3.1. Example. Suppose A and B are attributes such that dom(A) = dom(B) = {a,b,c}. Further assume that, in addition to = and ≠, the two domains are equipped with <, a transitive relation satisfying a<b<c. Suppose $\xi_1=\langle[0,5]\cup[9,9]\mapsto c, [6,7]\mapsto a\rangle$ and $\xi_2=\langle[3,9]\mapsto a\rangle$ are temporal assignments to A, and $\xi_3=\langle[8,10]\mapsto c\rangle$ is a temporal assignment to B. Then $\xi_1\upharpoonright[7,10]=\langle[7,7]\mapsto a, [9,9]\mapsto c\rangle$. $\xi_1\cup\xi_2$ is undefined because ξ_1 and ξ_2 take conflicting values at some instants, $\xi_1\cup\xi_3$ is undefined because ξ_1 and ξ_3 are temporal assignments to different attributes. $\xi_1\cap\xi_2=\langle[6,7]\mapsto a\rangle$ and $\xi_1-\xi_2=\langle[0,5]\cup[9,9]\mapsto c\rangle$, shift$(\xi_1)=\langle[1,6]\cup[10,10]\mapsto c, [7,8]\mapsto a\rangle$.

2.4. The micro when operator. Suppose ξ_1 and ξ_2 are temporal assignments to θ-comparable attributes A and B, respectively. We define the ***micro when operator*** $\llbracket\xi_1\theta\xi_2\rrbracket=\{t\in\llbracket\xi_1\rrbracket*\llbracket\xi_2\rrbracket: \xi_1(t)\theta\xi_2(t) \text{ holds}\}$. For ξ_1, ξ_2 and ξ_3 of Example 2.3.1, $\llbracket\xi_1<\xi_2\rrbracket=\emptyset$, $\llbracket\xi_2<\xi_1\rrbracket=[3,5]\cup[9,9]$. The construct $\llbracket\xi_1\theta\xi_2\rrbracket$ is of fundamental importance in temporal databases. Its value is a time domain which lies between $\emptyset$ and $\llbracket\xi_1\rrbracket*\llbracket\xi_2\rrbracket$. If the value is $\emptyset$, it says that the two temporal assignments were never related. One may use the micro when operator to define $\xi_1\theta_S\xi_2\leftrightarrow\llbracket\xi_1\theta\xi_2\rrbracket\neq\emptyset$, and $\xi_1\theta_A\xi_2\leftrightarrow\llbracket\xi_1\theta\xi_2\rrbracket=\llbracket\xi_1\rrbracket*\llbracket\xi_2\rrbracket\wedge\llbracket\xi_1\rrbracket*\llbracket\xi_2\rrbracket\neq\emptyset$. We observe that $\xi_1\theta_S\xi_2$ and $\xi_1\theta_A\xi_2$ would allow a user to capture "sometimes" and "always", respectively. ("Always" is used ambiguously in natural languages; we have given one interpretation.) The use of "=" needs some caution: $\xi_1=\xi_2$ is not used as a set theoretic comparison of ξ_1 and ξ_2, thus $\llbracket\xi_1=\xi_2\rrbracket=\{t: \xi_1(t)=\xi_2(t)\}$. For set theoretic comparison we use $\xi_1\subseteq\xi_2\wedge\xi_2\subseteq\xi_1$.

2.5. Tuples, relations and databases. A *tuple* τ over a scheme R is a function from R, such that for each attribute A of R, $\tau(A)$ is a temporal assignment to A. If τ is a tuple over R, and μ a temporal element, then $\tau\upharpoonright\mu$ is the function from R, such that $(\tau\upharpoonright\mu)(A)=\tau(A)\upharpoonright\mu$, for every $A\in R$. (Note that $\tau(A)$ may be empty for some $A\in R$, and in this case we say that $\tau(A)$ is a ***null***.) A *relation* over a scheme R is a finite set of non-null tuples over R. In Section 2.6 we introduce our concept of key for this paper, and require every relation to be equipped with a key. If r is a relation and μ a temporal element, then $r\upharpoonright\mu=\{\tau\upharpoonright\mu: \tau\in r\}$. A *database* is a finite set of relations.

2.6 Key. Now we introduce the concept of a key to be used in this paper. If ξ is an assignment, $|\xi|$ denotes its range $\{\xi(t): t\in\llbracket\xi\rrbracket\}$. Suppose r is a relation over R. We say that $K\subseteq R$ is the key of R, if (i) $|\tau(A)|$ is a singleton for every $A\in K$, and (ii) If τ and τ' are tuples of R, then $\forall A\in K(|\tau(A)|$

NAME	SALARY	DEPT
[11,60] John	[11,49] 15K [50,54] 20K [55,60] 25K	[11,44] Toys [45,60] Shoes
[0,20]U[41,51] Tom	[0,20] 20K [41,51] 30K	[0,20] Hardware [41,51] Clothing
[71,now] Inga	[71,now] 25K	[71,now] Clothing
[31,now] Leu	[31,now] 23K	[31,now] Toys
[0,44]U[50,now] Mary	[0,44]U [50,now] 25K	[0,44]U [50,now] Credit

The emp relation

DEPT	MANAGER
[11,49] Toys	[11,44] John [45,49] Leu
[41,47]U[71,now] Clothing	[41,47] Tom [71,now] Inga

The management relation.

Figure 2.1. The personnel database.

= $|\tau'(A)|$) if and only if $\tau = \tau'$. We assume that every relation has a key. We discuss our concept of a key in further details in Section 7.

2.7. The macro when operator. If r is a relation over R and A∈R, then we define the *macro when operator* $[\![r(A)]\!] = \cup_{\tau \in r}[\![\tau(A)]\!]$. Intuitively, $[\![r(A)]\!]$ extracts the temporal domain of a relation r along the column A. We also define $[\![r]\!] = \cup_{A \in R}[\![r(A)]\!]$.

2.8. Example. A database consisting of an emp relation over NAME SALARY DEPT with NAME as its key, and a management relation over DEPT MANAGER with DEPT as its key is shown in Figure 2.1. Note that $[\![\text{emp}]\!] = [0,\text{now}]$ and $[\![\text{management}]\!] = [11,49] + [71,\text{now}]$.

3. THE BASIC RELATIONAL ALGEBRA.

The algebraic expressions can be divided into four mutually exclusive groups: terms, Boolean expressions, temporal expressions and relational expressions. Throughout this paper, we will use the variables $\mathfrak{a}$ and $\mathfrak{b}$ to denote terms, $\mathfrak{m}$ and $\mathfrak{n}$ to denote temporal expressions, $\mathfrak{f}$ and $\mathfrak{g}$ to denote Boolean expressions, $\mathfrak{r}$ and $\mathfrak{s}$ to denote relational expressions, and $\mathfrak{e}$ to denote any expression. (We use subscripts if more variables are needed.) We say that an expression $\mathfrak{e}$ is *non-relational* if it is not a relational expression. Every expression $\mathfrak{e}$ is defined over a set of attributes R (R may be empty). Suppose $\mathfrak{e}$ is a non-relational expression defined over R. It is convenient to think of R as a list of parameters. The expression $\mathfrak{e}$ yields a concrete object $\mathfrak{e}(\tau)$, when a tuple τ over a scheme $R' \supseteq R$ is substituted in it. The value of $\mathfrak{e}(\tau)$ only depends upon $\tau(R)$, i.e. $\tau(R'-R)$ part of τ has no effect on $\mathfrak{e}(\tau)$. Thus if R is empty, then $\mathfrak{e}$ yields a concrete object without any tuple substitution. An expression is said to be *consummate* if it is either a non-relational expression over the empty scheme or a relational expression. A consummate expression $\mathfrak{e}$ can be used as a query (i.e. a terminal expression) or part of a query; it stands for a concrete object $\mathcal{I}(\mathfrak{e})$ on its own without needing a tuple substitution. We introduce the algebraic expressions below; for each non-consummate expression $\mathfrak{e}$ over R, we define $\tau(R')$, the result of substituting a tuple τ over $R' \supseteq R$, and for each consummate expression $\mathfrak{e}$ we define $\mathcal{I}(\mathfrak{e})$.

3.1. Terms. A term $\mathfrak{a}$ over a scheme R is paired with an attribute A∈R, called the *post* of $\mathfrak{a}$; on substitution of a tuple over $R' \supseteq R$ it yields an assignment over R. Thus a term is the syntactic counterpart of a temporal assignment. As A∈R, $R \neq \emptyset$, $\mathfrak{a}$ can not be a consummate expression. The terms with an attribute A as their post are defined below. For each term $\mathfrak{a}$ over R with A∈R as its post, we define $\mathfrak{a}(\tau)$, the result of substituting a tuple τ in $\mathfrak{a}$.

A (constant) element a ∈ dom(A) is considered a term with A as its post; $a(\tau)$ = the temporal assignment $\langle [0,\text{now}] \mapsto a \rangle$. For each attribute A, $A^{\vee}$ is a term; $A^{\vee}(\tau)$ is defined to be $\tau(A)$. If $\mathfrak{a}$ is a term over R with post A and $\mathfrak{m}$ a temporal expression over S, then $\mathfrak{a} \upharpoonright \mathfrak{m}$ is a term over R∪S with A as its post; if τ is a tuple over $R' \supseteq R \cup S$, then $\mathfrak{a} \upharpoonright \mathfrak{m}(\tau) = \mathfrak{a}(\tau) \upharpoonright \mathfrak{m}(\tau)$. If $\mathfrak{a}$ is a term over R with A as its post, then shift($\mathfrak{a}$) is a term over R with A as its post; if τ is a tuple over $R' \supseteq R$, then shift($\mathfrak{a}$)(τ) = shift($\mathfrak{a}(\tau)$).

3.1.1. Example. Consider the database of Figure 2.1. $\text{SALARY}^{\vee}$ is a term. Suppose τ is the John's tuple in the emp relation. Then $\text{SALARY}^{\vee}(\tau)$ evaluates to $\langle [11,49] \mapsto 15K, [50,54] \mapsto 20K, [55,60] \mapsto 25K \rangle$. Note that $\text{SALARY}^{\vee} \upharpoonright ([0,20] \cup [60,\text{now}])$ is also a term, and when τ is substituted, it evaluates to $\langle [11,20] \mapsto 15K, [60,60] \mapsto 25K \rangle$.

3.1.2. Notational Convention. From now on we write a term $A^{\vee}$ simply as A.

3.2. Temporal expressions. A temporal expression is the syntactic counterpart of a temporal element. The temporal expressions, and $\mathfrak{m}(\tau)$, the result of substituting τ in $\mathfrak{m}$ are defined as follows.

Every temporal element μ is a consummate temporal expression; $\mu(\tau) = \mathcal{I}(\mu) = \mu$. If $\mathfrak{a}$ is a term over R, then $[\![\mathfrak{a}]\!]$

is a temporal expression over R; $[\![\mathfrak{a}]\!](\tau)$ is defined to be $[\![\mathfrak{a}(\tau)]\!]$. If $\mathfrak{r}$ is a relational expression over R and A $\in$ R, then $[\![\mathfrak{r}(A)]\!]$ and $[\![\mathfrak{r}]\!]$ are consummate temporal expressions. $[\![\mathfrak{r}(A)]\!](\tau) = \mathcal{I}([\![\mathfrak{r}(A)]\!]) = \cup_{\tau' \in \mathcal{I}(\mathfrak{r})}[\![\tau'(A)]\!]$, and $[\![\mathfrak{r}]\!](\tau) = \mathcal{I}([\![\mathfrak{r}]\!]) = \cup_{A \in R}[\![\mathfrak{r}(A)]\!]$. If $\mathfrak{a}$ and $\mathfrak{b}$ are terms over R and S, respectively, then then $[\![\mathfrak{a}\theta\mathfrak{b}]\!]$ is a temporal expression over R$\cup$S; $[\![\mathfrak{a}\theta\mathfrak{b}]\!](\tau)$ is defined to be $[\![\mathfrak{a}(\tau)\theta\mathfrak{b}(\tau)]\!]$. If m and n are temporal expressions, then m+n is a temporal expression over R$\cup$S; $(m+n)(\tau) = m(\tau)+n(\tau)$. If m and n are consummate then so is m+n and $\mathcal{I}(m+n) = \mathcal{I}(m) + \mathcal{I}(n)$. The temporal expressions m$*$n, m$-$n, $-$m, fi(m) and fl(m) are similar.

3.2.1. Example. Let us again consider the database of Figure 2.1. Suppose m is the temporal expression $[\![$SALARY $\neq$ 25K$]\!]$ + $[\![$DEPT = Toys$]\!]$. (According to our notational convention introduced in Section 3.1.2, SALARY$^{\vee}$ and DEPT$^{\vee}$ have been written as SALARY and DEPT, respectively.) Suppose τ is John's tuple in the emp relation. SALARY is a term, SALARY(τ) evaluates to $\langle$[11,49]$\mapsto$15K, [50,54]$\mapsto$20K, [55,60]$\mapsto$25K$\rangle$. Now, $[\![$SALARY $\neq$ 25K$]\!](\tau)$ evaluates to [11,49] + [50,54] = [11,54]. Similarly, $[\![$DEPT = Toys$]\!](\tau)$ = [11,44]. Thus m(τ) = [11,54] + [11,44] = [11,54]. If τ is Tom's tuple, then m(τ) = [0,20] + [41,51]. If τ is the tuple of Mary or Inga, m(τ) = $\emptyset$. If τ is Leu's tuple, then m(τ) = [31,now].

3.3. Boolean expressions. A Boolean expression over R yields *true* or *false* on a tuple substitution. We only introduce their definitions, as their interpretation is similar to temporal expressions.

If m and n are Boolean expressions over R and S [resp. consummate Boolean expressions], then m=n and m$\subseteq$n are Boolean expressions over R$\cup$S [resp. consummate Boolean expressions]. The Boolean expressions $\mathfrak{a}\subseteq\mathfrak{b}$ and $\mathfrak{a}=\mathfrak{b}$, where $\mathfrak{a}$ and $\mathfrak{b}$ are terms, and $\neg\mathfrak{f}$, $\mathfrak{f}\vee\mathfrak{g}$ and $\mathfrak{f}\wedge\mathfrak{g}$, where $\mathfrak{f}$ and $\mathfrak{g}$ are Boolean expressions, are defined easily. If $\mathfrak{r}$ is a relational expression, then ($\mathfrak{r}=\emptyset$) is a consummate Boolean expression with a natural interpretation.

3.4. Relational expressions. In temporal databases certain weak identity operators arise naturally; however, for the ease of reading, we discuss them in Section 7. A relational expression is consummate; it yield a relation without any tuple substitution. The relational expressions and their interpretation are given below. Note that we do not specify the key for $\Pi_X(\mathfrak{e})$, $\mathfrak{r}\times\mathfrak{s}$, $\mathfrak{r}[\mathfrak{f}]\mathfrak{s}$ and $\mathfrak{r}$[m,X]$\mathfrak{s}$ and thus our semantics of these operators is not completely precise. This point is also discussed in Section 7. Throughout this sub-section, $\mathfrak{r}$ denotes a relational expression over R with K as it key, and $\mathfrak{s}$ denotes a relational expression over S.

- A stored relation r is a relational expression; $\mathcal{I}$(r) = r.
- If R = S and K is also the key of $\mathfrak{s}$, then $\mathfrak{r}\cup\mathfrak{s}$ is a relational expression over R with K as its key. Suppose τ_1 and τ_2 are tuples over R. We say that τ_1 and τ_2 *overlap* in K, if $\forall A \in K(|\tau_1(A)| = |\tau_2(A)|)$. Now suppose τ_1 and τ_2 overlap in K. If τ_1(A) and τ_2(A) agree (i.e., they do not have conflicting values at any instant) for all A$\in$R, we say that $\tau_1\cup\tau_2$ is *defined*, otherwise $\tau_1\cup\tau_2$ is *undefined*. If $\tau_1\cup\tau_2$ is defined, for each A$\in$R, $\tau_1\cup\tau_2(A)$ is defined to be $\tau_1(A)\cup\tau_2(A)$. To define $\mathcal{I}(\mathfrak{r}\cup\mathfrak{s})$ we recursively assume that $\mathcal{I}(\mathfrak{r})$ = r, and $\mathcal{I}(\mathfrak{s})$ = s. If there exists a pair of tuples $\tau_1\in$r and $\tau_2\in$s such that they overlap in K, but $\tau_1\cup\tau_2$ is undefined, then $\mathcal{I}(\mathfrak{r}\cup\mathfrak{s})$ is *undefined*. Now we assume that such a pair of tuples does not exist. Then we define $\mathcal{I}(\mathfrak{r}\cup\mathfrak{s}) = \{\tau$: ($\exists\tau_1\in\mathfrak{r}$ and $\exists\tau_2\in\mathfrak{s}$, such that τ_1 and τ_2 overlap in K and $\tau = \tau_1\cup\tau_2$) $\vee$ ($\tau\in$r and τ does not overlap in K with any tuple of s) $\vee$ ($\tau\in$s and τ does not overlap in K with any tuple of r)$\}$. We also allow the relational expressions $\mathfrak{r}-\mathfrak{s}$ and $\mathfrak{r}\cap\mathfrak{s}$; their interpretations $\mathcal{I}(\mathfrak{r}-\mathfrak{s})$ and $\mathcal{I}(\mathfrak{r}\cap\mathfrak{s})$ are always defined.
- If R and S are disjoint, then $\mathfrak{r}\times\mathfrak{s}$ is a relational expression over R$\cup$S; $\mathcal{I}(\mathfrak{r}\times\mathfrak{s})$ is the usual cross product $\mathcal{I}(\mathfrak{r})\times\mathcal{I}(\mathfrak{s})$ of $\mathcal{I}(\mathfrak{r})$ and $\mathcal{I}(\mathfrak{s})$.
- If X$\subseteq$R, then $\Pi_X(\mathfrak{r})$ is a relational expression over X; $\mathcal{I}(\Pi_X(\mathfrak{r})) = \{\tau(X): \tau\in\mathcal{I}(\mathfrak{r})\}$.
- If $\mathfrak{f}$ is a Boolean expression over R, then $\sigma(\mathfrak{r};\mathfrak{f})$ is a relational expression over R; $\mathcal{I}(\sigma(\mathfrak{r};\mathfrak{f}))=\{\tau\in\mathcal{I}(\mathfrak{r}):\mathfrak{f}(\tau)\}$.
- If m is a temporal expression over R and X$\subseteq$R, then $\omega(\mathfrak{r};[$m;X$])$ is a relational expression over R; $\mathcal{I}(\omega(\mathfrak{r};[$m;X$]))$ is the relation $\{(\tau(X)\restriction m(\tau))\circ\tau(R-X): \tau\in\mathcal{I}(\mathfrak{r})\}$ minus the null tuples. ($\circ$ denotes concatenation.) ω is called the *when operator*; it allows X part of a tuple τ of a relation to be restricted to the value of m computed for τ. If X = R, $\omega(\mathfrak{r};[$m;X$])$ is simply written as $\omega(\mathfrak{r};$m$)$.
- If R and S are disjoint, and $\mathfrak{f}$ is a Boolean expression over R$\cup$S, then $\mathfrak{r}[\mathfrak{f}]\mathfrak{s}$ is a relational expression over R$\cup$S; $I(\mathfrak{r}[\mathfrak{f}]\mathfrak{s}) = \{\tau_1\circ\tau_2: \tau_1\in\mathcal{I}(\mathfrak{r}), \tau_2\in\mathcal{I}(\mathfrak{s})$ and $\mathfrak{f}(\tau_1\circ\tau_2)\}$, where $\circ$ denotes concatenation.
- If R and S are disjoint, m is a temporal expression over R$\cup$S, and X$\subseteq$R$\cup$S, then $\mathfrak{r}[$m;X$]\mathfrak{s}$ is a relational expression over R$\cup$S; $\mathcal{I}(\mathfrak{r}[$m;X$]\mathfrak{s}) = \{\tau$: $\exists\tau_1\in\mathcal{I}(\mathfrak{r})$ $\exists\tau_2\in\mathcal{I}(\mathfrak{s})$ such that $\tau = (\tau_1\circ\tau_2)(X)\restriction m(\tau_1\circ\tau_2) \circ (\tau_1\circ\tau_2)(R\cup S-X)$ and τ is not null$\}$. ($\circ$ denotes concatenation.)
- If A $\in$ R and B $\notin$ R, then $\delta_{A\leftarrow B}(\mathfrak{r})$ is a relational expression over (R$\cup$$\{B\}$)$-\{A\}$; $\mathcal{I}(\delta_{A\leftarrow B}(\mathfrak{r})) = \delta_{A\leftarrow B}(\mathcal{I}(\mathfrak{r}))$. (At-

tribute A is renamed to B.)

3.5. Query Examples. It is clear that the above definition gives rise to very powerful expressions. For example a temporal expression can involve relations and it may be used to define a Boolean expression or a relational expression. We end this section with a few illustrations.

3.5.1. Example. Give details about those employees who earned a salary greater than 24K while they were in the Clothing or the Shoes departments.

$\omega(\text{emp}; [\![\text{SALARY} > 24\text{K}]\!]$
$\quad *([\![\text{DEPT} = \text{Shoes}]\!]+[\![\text{DEPT} = \text{Clothing}]\!]))$

3.5.2. Example. List the NAME and SALARY of all employees in Toys Department during the time when John was a manager in some department.

$\Pi_{\text{NAME SALARY}}(\omega(\text{emp};$
$\quad [\![\omega(\text{management};[\![\text{MANAGER}=\text{John}]\!])]\!] * [\![\text{DEPT}=\text{Toys}]\!]))$

3.5.3. Example. List the starting salaries of the employees who are currently employed by the organization.

$\Pi_{\text{NAME SALARY}}(\omega(\sigma(\text{emp}; \text{now} \subseteq [\![\text{DEPT}]\!]); \text{fi}([\![\text{SALARY}]\!]))$

3.5.4. Example. When did John's salary increase? The query is expressed by using the shift operator over an assignment as follows.

$[\![\omega(\text{emp}; [\![\text{NAME}=\text{John}]\!] * [\![\text{SALARY}>\text{shift}(\text{SALARY})]\!])]\!]$.

4. THE GENERALIZED MODEL.

We have observed that the model presented in the previous section allows substantial local navigation. In that model, the time stamps were one dimensional, allowing us to attach a single concept of time to objects. For example, we could view it to be the *valid time* [Sn1], i.e., the time of existence of objects in the real world, or the *transaction time* [Sn1], i.e., the time information is entered in the database. Given a positive integer N, we now generalize our model to deal with N orthogonal temporal dimensions. We let N–[0,now] denote the cross product of N copies of [0,now].

4.1. PROPOSITION. The set consisting of finite unions of *rectangles* of the form $I_1 \times I_2 \times \ldots \times I_N$, where I_k is an interval in [0,now], $1 \leq k \leq N$, is a Boolean algebra with respect to set theoretic union (denoted +), intersection (denoted *) and complementation (denoted –) with respect to N–[0,now].

We will denote the Boolean algebra mentioned in the above proposition as N–*TE* and its elements are called N–elements. An an N–instant is of the form $\langle t_1, t_2, \ldots, t_N \rangle$ where $t_1, t_2, \ldots, t_N$ are instants. If μ is an N–element then for $1 \leq k \leq N$, $\pi_k(\mu) = \{t_k: \langle t_1, t_2, \ldots, t_N\rangle \in \mu\}$ is called the k–projection of μ. If μ is an N–element, ν is a 1–element, and $1 \leq k \leq N$, then $\mu\downarrow_k\nu$ denotes the N–element obtained from μ by restricting (intersecting) its k–th dimension to ν. If μ and ν are N–elements, then we say that $\mu \leq \nu$ if for every k, $1 \leq k \leq N$, $\pi_k\mu \leq \pi_k\nu$.

4.2. N–dimensional assignments. An N–assignment to A with an N–element μ as its temporal domain is a function from μ into dom(A). If ξ is an N–assignment, then $[\![\xi]\!]$ denotes the domain of ξ, which is an N–element, and for each k, $1 \leq k \leq N$, and 1–element ν, $\xi\downarrow_k\nu$ denotes the restriction of ξ to the domain $[\![\xi]\!]\downarrow_k\nu$. We define $[\![\xi_1 \theta \xi_2]\!] = \{t \in \text{N–}[0,\text{now}]: t \in [\![\xi_1]\!]*[\![\xi_2]\!]$ such that $(\xi_1\downarrow t)\theta(\xi_2\downarrow t)$ holds$\}$.

4.3. N–dimensional tuples, relations and databases. An N–tuple over R is a function from R such that for each $A \in R$, $\tau(A)$ is an N–assignment to A. An N–relation over R is a finite set of N–tuples over R. An N–database is a finite set of N–relations. Next, we turn our attention to N–algebra. But we first introduce a change in our notational terminology.

4.4. New Notational conventions. From now onwards we drop the prefix N–. For example, when we say ξ is an assignment, it is understood that it is a N–assignment. Similar remarks apply to the other constructs. When we want to refer to one dimensional objects, we will use 1– as a prefix.

With the above notational conventions, almost all the algebraic expressions introduced in Section 3 can now be considered as expressions for the N–dimensional case. However a few expressions need changes: the shift(α) in the N–dimensional case is replaced with shift$_k(\alpha)$, $1 \leq k \leq N$; it represents the shift of α in the k–th dimension. We replace fi and fl with the 1–temporal expressions fi$_k$(m) and fl$_k$(m), respectively, $1 \leq k \leq N$. We also add the following expressions to enable one to manipulate each individual temporal dimension.

- If m is a temporal expression, then $\pi_k(\text{m})$, is a 1–temporal expression, for $1 \leq k \leq N$.
- If m is a temporal expression and n a 1–temporal expression, then $\text{m}\downarrow_k\text{n}$ is a temporal expression, for $1<k<N$.
- If α is a term and n a 1–temporal expression, then $\alpha\downarrow_k\text{n}$ is a temporal expression.

5. AN APPLICATION TO DATABASES.

In this section we assume that N = 2. We fix the first and the second temporal dimensions to be the *transaction time* [Sn2] and the *valid time* [Sn2], respectively to model the evolution of the history of a database. We need to give an appropriate interpretation to the concept of the transaction time to model it in a powerful manner. Suppose α is an object. Suppose at transaction time t, α is either created with v as its value or α is given a new value v. Further suppose that $t' \geq t$ be the first transaction instant after t, when v is either deleted, or modified. Then we take the point of view that the value v of the object α holds during the entire interval $[t,t')$ in the transaction time dimension.

We assume that for every assignment ξ in the database, if $\langle t,t' \rangle \in [\![\xi]\!]$, then $t \geq t'$. This just says that an update for real world time t' does not go into effect before t'. We assume that the information about the real world behavior of an object appears correctly at the transaction time now. (Of course we may find in future that we were wrong!) We also assume that there is no error in any key attribute. Traditionally, after an update has been made we do not know whether it was a new attribute value assumed by an object or it was a correction. Let ξ be a 2–assignment. We now discuss how to give a precise classification of errors and updates in ξ. As stated above, we assume that for all t, $\xi\langle now,t \rangle$ is correct.

5.1. Classification of errors. We fix an arbitrary real world time instant t_0. By comparing it with $\xi \restriction \langle t,t_0 \rangle$, we may classify errors in ξ at transaction time t as follows:

- No error: $\xi \restriction \langle t,t_0 \rangle$ and $\xi \restriction \langle now,t_0 \rangle$ are equal
- Missing information: $\xi \restriction \langle t,t_0 \rangle = \emptyset$ and $\xi \restriction \langle now,t_0 \rangle \neq \emptyset$
- Extraneous information: $\xi \restriction \langle t,t_0 \rangle \neq \emptyset$ and $\xi \restriction \langle now,t_0 \rangle = \emptyset$
- Incorrect information: $\xi \restriction \langle t,t_0 \rangle$ and $\xi \restriction \langle now,t_0 \rangle$ are non empty and unequal

5.2. Classification of updates. We fix the transaction time to now. We classify updates, i.e., the changes in the real world at time t by comparing $\xi \restriction \langle now,t \rangle$ with $\xi \restriction \langle now,t-1 \rangle$ as follows:

- No update: $\xi \restriction \langle now,t \rangle = \xi \restriction \langle now,t-1 \rangle$
- Creation of new information: $\xi \restriction \langle now,t \rangle \neq \emptyset$ and $\xi \restriction \langle now,t-1 \rangle = \emptyset$
- End of existing information: $\xi \restriction \langle now,t \rangle = \emptyset$ and $\xi \restriction \langle now,t-1 \rangle \neq \emptyset$
- Change in information: $\xi \restriction \langle now,t \rangle$ and $\xi \restriction \langle now,t-1 \rangle$ are non empty and unequal

Our algebra holds a great promise for querying errors and updates in a database. We give an illustration.

5.3. Example. Suppose emp is now a 2–relation. Let us ask, what were all the instants when some update was made in the salary or department of an employee, such that at the time of the update we thought we were making a correction? For the sake of discussion, suppose that $\tau \in emp$. The key is to make the $\neq$ comparison between $\tau(A) \restriction \langle t+1,t_0 \rangle$ and $\tau(A) \restriction \langle t,t_0 \rangle$. But, the micro when operator only allows us to capture comparisons at the same 2–instant, and not at the different 2–instants like $\langle t+1,t_0 \rangle$ and $\langle t,t_0 \rangle$. This is where $shift_1$ comes handy in expressing the query as follows:

$$\pi_1([\![\omega(emp;\ [\![shift_1(SALARY) \neq SALARY]\!] + [\![shift_1(DEPT) \neq DEPT]\!])]\!])$$

6. COMPARISON WITH OTHER WORKS

In this section we discuss papers directly related to our work. Some additional references have been included. Due to limitations of space our list is not complete; [Bo,Mc,Sn3] are excellent sources for further references.

A tuple τ over R is *homogeneous* if for all attributes $A \in R$, $[\![\tau(A)]\!]$ is the same. In [Ga1–Ga3,Ga,GV], the tuples have been assumed to be homogeneous. [Ga1] gives a relational algebra and an equivalent tuple calculus for temporal databases consisting of only homogeneous relations; [GV] gives a QUEL like interface for this framework. [Ga2] generalizes [Ga1] to allow the scheme R of a relation r to be split as a disjoint union $R_1 \cup \ldots \cup R_k$, such that for each $\tau \in r$, $\tau(R_1), \ldots, \tau(R_k)$ are homogeneous; such a relation r is said to be multihomogeneous. The prefix *multi* in multihomogeneous is used in a very different sense than its use in multidimensional time stamp; in the multihomogeneous model, the time stamps are still one dimensional. In this paper we have not imposed the homogeneity requirement in any form.

Suppose Δ is the database $\{r_1, r_2, \ldots, r_k\}$. If μ is a temporal element, then $\Delta \restriction \mu$ denotes the database $\{r_1 \restriction \mu, r_2 \restriction \mu, \ldots, r_k \restriction \mu\}$. In the traditional snapshot style databases, one can only query $\Delta \restriction (\{now\} \times \{now\})$. [Sn2,MS1,MS2] support transaction time to query $\Delta_t = \Delta \restriction (\{t\} \times [0,now])$, where t is arbitrary but a fixed constant in a given query. Since Δ_t is isomorphic to a temporal database with a single temporal dimension, a query system supporting a single temporal dimension is easily adopted to query Δ_t. Our generalization treats the two (or more) temporal dimensions symmetrically, without an inherent limitation on the interaction among them. The time stamps in a relation used in a TQUEL [Sn1,Sn2] query are intervals, which do not form a Boolean algebra. This makes it difficult for one to capture the meaning of *or*, *and* and *not* of natural languages; an in–depth study of TQUEL has been undertaken in [GY]. [GY] also presents a tuple calculus for a temporal database, which is essentially equivalent to the relational algebra for the single temporal dimension presented in Section 3. TSQL [NA], a query language for a single dimension shares the philosophy of TQUEL, and similar remarks may also be made about it. However, TSQL and a newer version of TQUEL [SGM] support aggregates. The aggregates have not been covered in our paper.

[Ta] gives a model and an algebra for temporal databases. His basic philosophy is to start with a temporal assignment ξ to an attribute A, similar to ours. An *unpack*$_A$ operator breaks ξ into smaller temporal assignments ξ', such that each $[\![\xi']\!]$ is an interval. Each resulting assignment $\xi' = \langle [t_1,t_2],a \rangle$ can be further *decomposed*, by using t–dec$_A$ operator, into three ordinary assignments $\langle t_1 \rangle$, $\langle t_2 \rangle$ and $\langle a \rangle$ to the attributes A_L, A_U and A, respectively. (L and U stand for the lower and the upper end points t_1, and t_2 of $[t_1,t_2]$, respectively. The attributes A_L A_U are created automatically by the t–dec$_A$ operator.) Thus an assignment is completely broken into several first normal form assignments ξ''; now one may use the usual θ operators for navigation and finally glue the results back to form a temporal assignment. Thus the navigation is typically done in up to five steps unpack → t–dec → application of a θ operator → t–form → pack. The approach in [MS2] makes the pack, unpack, t–dec and the t–form operators implicit in the semantics, hiding it from a user. Navigation in [Ta] and [MS2] is through intervals. The navigation in [Ye,GY] is directly in terms of temporal assignments; we feel that this is the most natural form of navigation expected to arise in temporal databases. We end this section with the remark that the concept of the when operator ω is implicit in [Sn2] and [Ta]; however its full potential can only be realized through local navigation.

7. WEAK IDENTITY OPERATORS

In this section we only consider the one dimensional case. Our ideas are easily generalized to higher dimensions. Two relations r and s over R are said to be weakly equal, if for every instant t ∈ [0,now], $r \upharpoonright t = s \upharpoonright t$ [Ga1,Ga3]. Although weakly equal relations are not identical, in some sense they have the same information content. For example the emp relation in Figure 2.1 and the newEmp relation in Figure 7.1 are weakly equal. Weak equality is an equivalence relation; thus a relation r gives rise to an equivalence class denoted as [r], which is also called a weak relation. A binary operator ⊙ is said to be weakly invariant, if r ⊙ s is weakly equal to r' ⊙ s', whenever r and r' are weakly equal and s and s' are weakly equal. (A weakly invariant unary operator is defined in a similar manner.) Thus the result of a weakly invariant operator is determined only up to weak equality. The when operator ω is not a weakly invariant operation; in fact local navigation is important, but it is not a weakly *invariant concept*. Thus in temporal databases idea is perhaps to allow weak relations, but not weakly invariant operators. Weak relations enable us to view the information content of a relation r in a flexible manner through all those relations which are weakly equal to r. There is one essential class of weak relations which we believe are of immediate importance form our point of view: given a relation r, these relations are the ones which arise from changing the key of r. The rest of this section is devoted to the weak relations arising from a key.

NAME		SALARY		DEPT	
[11,44]	John	[11,44]	15K	[11,44]	Toys
[45,49]	John	[45,49]	15K	[45,49]	Shoes
[50,54]	John	[50,54]	20K	[50,54]	Shoes
[55,60]	John	[55,60]	25K	[55,60]	Shoes
[0,20]	Tom	[0,20]	20K	[0,20]	Hardware
[41,51]	Tom	[41,51]	30K	[41,51]	Clothing
[71,now]	Inga	[71,now]	25K	[71,now]	Clothing
[31,now]	Leu	[31,now]	23K	[31,now]	Toys
[0,44]∪[50,now]	Mary	[0,44]∪[50,now]	25K	[0,44]∪[50,now]	Credit

newEmp with the key NAME SALARY DEPT

Figure 7.1.

Let us first recall our definition of a key. Suppose r is a relation over R. A set K ⊆ R is said to be a *key* of r, if (i) $\forall \tau \in r(|\tau(A)|$ is a singleton) and (ii) whenever τ_1 and τ_2 are tuples of r, then $\forall A \in K(|\tau_1(A)| = |\tau_2(A)|)$ if and only if $\tau_1 = \tau_2$. Note that the emp relation in Figure 2.1 and the newEmp relation in Figure 7.1 are weakly equal. The difference between them is captured by their keys: NAME is the key of emp, and NAME SALARY DEPT is the key of the newEmp relation. We have assumed that every relation is equipped with a key. Thus we may say that changing the key of emp relation from NAME to NAME SALARY DEPT gives us the newEmp relation. In this case the key of emp relation is properly contained in the key of newEmp relation. Sometimes there may be a pair of keys such that neither of the two keys is properly contained in the other.

FLIGHT	FROM	TO	DEPARTS
[11,30] F261	[11,20] JFK [21,30] JFK	[11,20] CHI [21,30] CHI	[11,20] 4PM [21,30] 5PM
[11,30] F361	[11,20] JFK [21,30] JFK	[11,20] CHI [21,30] CHI	[11,20] 6PM [21,30] 4PM

(a) flights1 with FLIGHT as the key

FLIGHT	FROM	TO	DEPARTS
[11,20] F261 [21,30] F361	[11,30] JFK	[11,30] CHI	[11,30] 4PM
[21,30] F261	[21,30] JFK	[21,30] CHI	[21,30] 5PM
[11,20] F361	[11,20] JFK	[11,20] CHI	[11,20] 6PM

(b) flights2 with FROM TO DEPARTS as the key

Figure 7.2

The interesting behavior of such keys is illustrated in the following example.

7.1. Example. Let R = FLIGHT FROM TO DEPARTS. A tuple over R is a record about a flight operated by a certain airline; the attributes FLIGHT and DEPARTS stand for the flight number and its time of departure, respectively. FROM and TO have a natural meaning. (We like to point out that the domain of DEPARTS happens to be a set of time instants, but it helps to think of it as any other ordinary attributes.) Now suppose r is an arbitrary relation over R only determined up to weak equality. Assuming that two flights do not not depart from the same airport, going to the same airport, at the same time, the functional dependencies FLIGHT → R and FROM TO DEPARTS → R hold in $r\uparrow t$ for every t. We have two minimal choices for a key: K_1 = FLIGHT and K_2 = FROM TO DEPARTS. In this case neither $K_1 \subseteq K_2$ nor $K_2 \subseteq K_1$. Suppose flights1 and flights2 are the relations with K_1 and K_2 as their keys, respectively. An interesting thing happens: a tuple in flights1 consists of parts of several tuples of flight2, and vice versa. Figure 7.2 illustrates this. We feel that this situation is somewhat unique to temporal databases. It is clear that the weak relations play an important role.

In the classical static relational databases, aggregate functions provide a mechanism to group a set of tuples to form a partition. The corresponding situation carries on to

SALARY	DEPT
[11,44] 15K	[11,44] Toys
[45,49] 15K	[45,49] Shoes
[50,54] 20K	[50,54] Shoes
[55,60] 25K	[55,60] Shoes
[0,20] 20K	[0,20] Hardware
[41,51] 30K	[41,51] Clothing
[71,now] 25K	[71,now] Clothing
[31,now] 23K	[31,now] Toys
[0,44]U 25K [50,now]	[0,44]U Credit [50,now]

$\Pi_{SALARY\ DEPT}(emp)$ with the key SALARY DEPT

Figure 7.3.

the temporal case also. One should not equate a partition with an object. An object τ (a tuple) in a temporal relation has the property that its restriction to a single attribute is a single valued function of time. A user can implicitly or explicitly capture an object into a single tuple variable x and exercise considerable power in its manipulation. On the other hand, the manipulation of partitions is confined to rather simple operations.

Now we discuss another use of the weak relations. It turns out that our semantics of $\Pi_X(r)$, r×s, r[f]s and r[m,X]s is not completely precise. Although we do not take a specific point of view regarding this problem in this paper, we discuss how one could use weak relations arising from a key to make this semantics precise. Suppose r is a relation over R with K as its key. In case K is not contained in X, our semantics of $\Pi_X(r)$ in Section 3 is somewhat incomplete as we do not know its key. For example, $\Pi_{NAME\ DEPT}(emp)$ is well defined, but $\Pi_{SALARY\ DEPT}(emp)$ is not. In the case when K is not a subset of X, an alternative is to expect the user to specify the key (by making some appropriate provisions in the syntax of a projection operator); another alternative is to use a default, and let the whole scheme X in $\Pi_X(r)$ to be the key; in this case, $\Pi_{SALARY\ DEPT}(emp)$ would be as shown in the Figure 7.3. Now we introduce the weak identity operators.

7.2. Weak Identity Operators. Suppose R is a scheme and $K_1, K_2, \ldots, K_m$ are all possible keys $\subseteq$ R. Then we assume that we are given $I_{K_1}, I_{K_2}, \ldots, I_{K_m}$, called the weak identity operators, such that if r is a relation over R with any key, $I_{K_i}(r)$ is weakly equal to r with K_i as its key.

8. CONCLUSION.

In the traditional relational approach only the current information is maintained at the front end. In case we are required to maintain the history of all past transactions, our two dimensional model can be used to impose a structure on these transactions, with the advantage that we can query the past states as well as their correctness. We believe that this is a promising application of temporal databases to databases. Although our one dimensional model is not literally a generalization of all the other approaches to temporal databases, but we hope that they would be benefited by our generalized approach.

ACKNOWLEDGEMENTS. We like to thank the anonymous referees and Gautam Bhargava for their helpful suggestions.

REFERENCES

[Bo] Bolour, A., Anderson, T.L., Dekeyser, L.J. and Wong, H.K.T. *The Role of Time in Information Processing, A survey.* SIGART Newsletter, 80, April 1982, pp 28–48. 1987, pp 528–537.

[CC] Clifford, J. and Croker, A. *The Historical Data Model (HRDM) an Algebra Based on Lifespans.* Proceedings of the third IEEE International Conference on Data Engineering, 1987.

[Ga1] Gadia, Shashi K. *A Homogeneous Relational Model and Query Languages for Temporal Databases.* To appear in ACM–TODS.

[Ga2] Gadia, Shashi K. *Toward a Multihomogeneous Model for a Temporal Database.* Proceedings of the IEEE International Conference on Data Engineering, 1986, pp 390–397.

[Ga3] Gadia, Shashi K. *Weak Temporal Relations.* Proc. Fifth Annual ACM SIGACT– SIGMOD Symp. on Principles of Database Systems, 1986. pp 70–77.

[GV] Gadia, Shashi K. and Vaishnav, Jay. *A Query Language for a Homogeneous Temporal Database.* Proc. Fourth Annual ACM SIGACT–SIGMOD Symp. on Principles of Database Systems,1985. pp 51–56.

[GY] Gadia, Shashi K. and Yeung, Chuen–Sing. *A tuple calculus for temporal databases and its comparison with TQUEL.* To appear in Information Sciences.

[JM] Jones, S. and Mason, P. *Handling the Time Dimension in Databases.* Proc. of the International Conference on Databases, British Computer Society. University of Aberdeen, July 1980, pp 66–83.

[Mc] McKenzie, E. *Bibliography: Temporal Databases.* ACM–SIGMOD Record, December 1986, pp 40–52.

[MS1] McKenzie, E. and Snodgrass R. *Extending* the *Relational Algebra to Support Transaction Time.* Proc. ACM–SIGMOD International Conference on Management of Data, 1987, pp. 467–478.

[MS2] McKenzie, E. and Snodgrass R. *Supporting Valid Time. An Historical Algebra.* Technical Report TR87–008, Aug 1987. Computer Science Dept., University of North Carolina at Chapel Hill..

[MS3] McKenzie, E. and Snodgrass, R. *Scheme Evolution and the Relational Algebra.* Tech. Report, TR87–003, May 1987. Computer Science Dept. Univ. of North Carolina at Chapel Hill.

[NA] Navathe, S.B. and Ahmed, Rafi. *A Temporal Relational Model and Query Language.* Technical Report, 1986, Database System R & D Center, University of Florida.

[RS] Rotem, Doron and Segev, Arie. *Physical Organization of Temporal Data.* Proceedings of the third IEEE International Conference on Data Engineering, 1987, pp 547–553.

[SK] Shoshani, A. and Kawagoe, K. *Temporal Data Management.* VLDB, Japan, 1986.

[Sn1] Snodgrass, Richard. *The Temporal Query Language TQUEL.* Proc. Third ACM SIGACT–SIGMOD Symp. on Principles of Database Systems, Waterloo, April 1984, pp 204–212.

[Sn2] Snodgrass, Richard. *The Temporal Query Language TQuel.* ACM Transactions on Database Systems, Vol 12, 1987, pp 247–298.

[Sn3] Snodgrass, R. ED. *Research concerning time in databases: Project summaries.* ACM SIGMOD Rec. Vol 15, Dec 1986, pp 19–39.

[SA] Snodgrass, Richard and Ahn, I. *A Taxonomy of Time in Databases.* Proc ACM–SIGMOD Int. Conf. on Management of Data, May 1985, pp. 246.

[SGM] Snodgrass, R., Gomez, S. and McKenzie, Ed. *Aggregates in Temporal Query Language TQUEL.* TEMPIS Document 16. Computer Science Dept. University of North Carolina at Chapel Hill.

[Ta] Tansel, A. U. *Adding Time Dimension to Relational Model and Extending Relational Algebra.* Information Systems, 1986.

[TA1] Tansel, A. U. *A Statistical Interface for Historical Relational Databases.* Proc.third IEEE Int. Conf. on Data Engineering, 1987, pp 538–546

[TA2] Tansel, A. U. and Arkun M. E. *Historical Query Languages,* 1985. (Submitted for publication.)

[TA3] Tansel, A. U. and Arkun M. E. *A Query Language for Historical Relational Databases,* 1986. Tech. Report, Bernard M. Baruch College.

[TAO] Tansel, A.U., M.E. Arkun and G. Ozsoyoglu. *Time–By–Example Query Language For Historical Databases.* To appear in IEEE Transactions on Software Engineering.

[Ye] Yeung, Chuen–Sing. *Query Languages for a Heterogeneous Temporal Database.* M.S. Thesis, Texas Tech University, August 1986.