Modeling Temporal Dynamics for Business Systems[1]

Gove N. Allen, Tulane University, USA
Salvatore T. March, Vanderbilt University, USA

ABSTRACT

Research in temporal database management has viewed temporal dynamics from a structural perspective, posing extensions to the Entity-Relationship (E-R) model to represent the state history of time-dependent attributes and relationships. We argue that temporal dynamics are semantic rather than structural and that the existing constructs in the E-R model are sufficient to represent them. Practitioners have long used E-R models without temporal extensions to design systems with rich support for temporality by modeling both things and events as entities — a practice that is consistent with the original presentation of the E-R model. This approach supports methodologies that leverage narrative and human cognitive processing capabilities in the development and verification of data models. Furthermore it maintains modeling parsimony and facilitates the representation of causality — why a particular state exists.

Keywords: Conceptual modeling, Entity-Relationship models, temporal databases, temporal data models, time semantics, event representation

INTRODUCTION

Numerous authors recognize the importance of representing the temporal, or time-dependent, aspects of business data. These include Dey, Barron, and Storey (1995), Ozsoyoglu and Snodgrass (1995), Etzion, Jajodia, and Sripada (1998), Gregersen and Jensen (1999), Snodgrass (2000), and March and Allen (2003). Managers frequently must know not only the most current data but also historical data. They require facts such as:

- a specific customer's account balance on a specific date (e.g., on the date that customer was refused additional credit) and the transaction history that lead to that balance,
- the length of time an employee has been at his or her current salary level and the history of salary reviews and salary changes, and

- the last date on which a stock-out was experienced for a specific inventory item and the history of sales and purchases for that item.

Business systems increasingly require support for temporality. In a seminal book on temporal database research, theory, and implementation Tansel et al. (1993) state the motivating assumption of temporal database research as follows, "Conventional databases were designed to capture the most recent data, that is, current data. As new values become available through updates, the existing data values are removed from the database. Such databases capture a snapshot of reality. Although conventional databases serve some applications well, they are insufficient for those in which past and/or future data are also required. What is needed is a database that fully supports the storage and querying of information that varies over time" (preface). Depending on the domain of interest, simply representing time-varying information may not be sufficient to meaningfully reconstruct its history. It may be necessary to represent the causes of those variations.

Numerous extensions to the relational model and relational databases have been posed seeking to better manage the temporal nature of business information. These include the notion of *temporal functional dependency* (Wang, et al., 1997), techniques to facilitate temporal queries (Dean, 1989; Gadia and Yeung, 1988), extensions to the relational model and relational algebra (Gadia, 1988; McKenzie and Snodgrass, 1987), and indexing algorithms to support temporal queries (Kouramajian et al., 1994; Kumar,et al., 1998). The complexity added by these extensions immediately suggests a need for temporal support at the conceptual level. Over a dozen temporal Entity-Relationship (E-R) models have been proposed (Gregersen and Jensen, 1999; Dey, et al., 1995). These extend the E-R model by adding syntactic constructs and changing the fundamental definitions of existing constructs. They are based on the assumption that, "The E-R model can at best represent a 'snapshot' of the real world at any point in time; it does not contain specific constructs to model the dynamic aspects of the real world. As a result, the E-R model is an inadequate tool for temporal database design" (Dey, et al., 1995, p. 306).

Temporality has arisen as a major problem in data representations not because of limitations to the E-R or relational models but because database design approaches have *viewed* data models as representing only a snapshot of the world at a point in time and recommend ignoring the temporal aspects of the application during initial data modeling (Snodgrass, 2000). Temporality is addressed in a post hoc, structural manner by identifying "temporal attributes" in a snapshot model and using a temporal DBMS to maintain state history for those attributes.

We contend that a data model should not be so viewed nor should the temporal aspects of an application be left to post hoc analysis. A data model represents the semantics necessary to support a specific application's purpose. If that purpose includes temporality, then the temporal nature of the data must be represented in the data model. Time is an integral part of the semantics of most business systems and should not be treated as a simple structural addition. Its semantics should be modeled. Furthermore, merely representing the state history of temporal attributes is often semantically deficient.

It may be necessary to identify and explicitly describe the events that cause state changes rather than simply recording what past states have been. For example, in an accounting application it is not sufficient to represent the fact that a customer's account balance changed from $10,000 to $8,000 on September 1, 2002. The *cause* of that change must also be represented. Was the change due to a payment or a return or a credit or a negotiated settlement? Explicitly representing events such as receive payment, receive return, issue credit, and negotiate settlement provide that causal explanation. Such a representation is consistent with the E-R model (Chen, 1976), which defines an entity as a "thing or event" that is important to the domain being modeled. Such event entities enable the reconstruction of the history of the domain including changes and the events that caused them. Hence they are the foundation of an accurate and explicit representation of the causal aspects of temporality.

By representing events as entities the temporal dynamics of an application can be simply and parsimoniously represented without structurally extending the E-R model. Organizations often create formal documents (artifacts) to record data about specific events (e.g., sell, pick, pack, ship, invoice). Such artifacts are naturally represented as entities in a data model. However, documents are not always created for all important events. Hence, event analysis (McCarthy, 1982; Denna, et al, 1993; Geerts and McCarthy, 2002) provides a practical approach to modeling temporality. We argue that focusing on events aids an analyst in determining and validating those aspects of temporal dynamics that are important for the application domain.

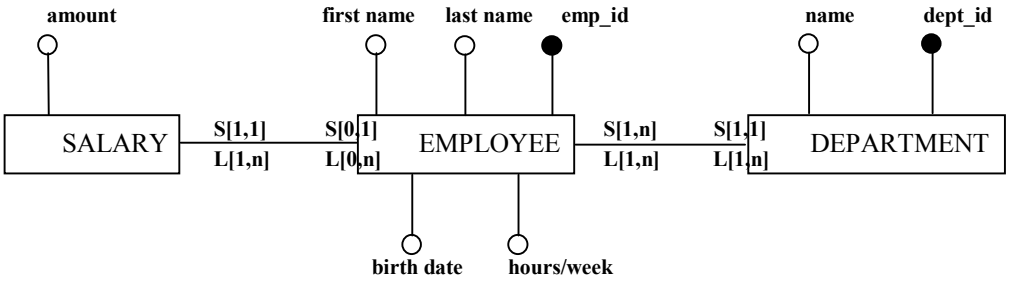
TEMPORALITY IN LOGICAL AND CONCEPTUAL DATABASE DESIGN

Two approaches have been proposed for representing temporality in relational databases, *tuple-versioning* and *attribute-versioning*. Tuple versioning retains first normal form relations (Snodgrass, 2000) while attribute versioning uses non-first normal form relations (Gadia and Vaishnav, 1985; Clifford and Tansel, 1985). Each begins with a snapshot schema, representing the current state of an organization, and provides a means to store past states of temporal attributes (using tuples or attributes, respectively).

Proposed modifications to the E-R model to support these *state-history* approaches to logical database design are varied. A recent survey by Gregersen and Jensen (1999) identified ten temporally extended E-R models. These all begin with the notion that the basic E-R model represents only current state. Hence they add various constructs to denote when state history must be maintained. Here we discuss two such approaches, the Temporal Entity-Relationship Model (TER) (Tauzovich, 1991) and Temporal Event-Entity-Relationship Model (TEERM) (Dey, et al., 1995). TER is an extension of the binary E-R model. It is composed of two models. The first is a temporally enhanced E-R model. The second is a conversion of that model into a traditional E-R model, which is then converted to a relational database schema. TER focuses on temporality of relationships. It does not explicitly address temporality of attributes. Figure 1 shows a TER diagram with employee, department and salary entities.

TER makes one major change to the basic E-R model—it replaces the notion

Figure 1: TER Diagram

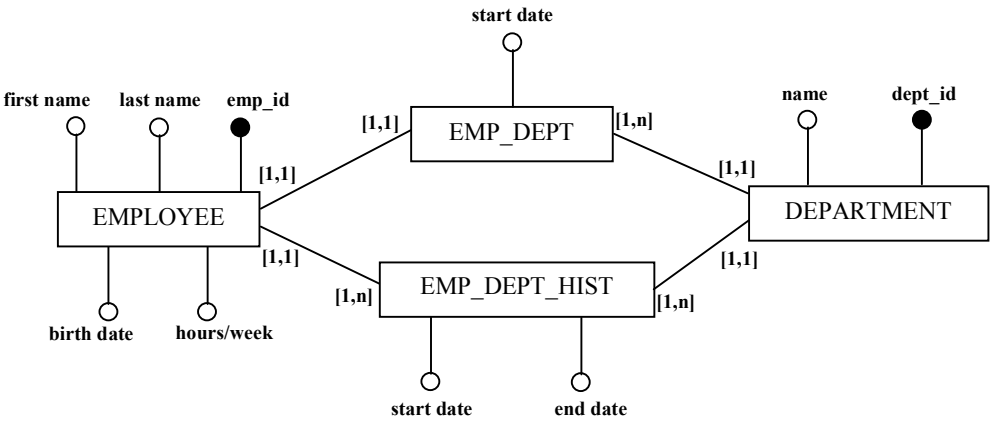


of minimum and maximum cardinality with *snapshot* and *lifetime* minimum and maximum cardinalities. Thus the S[1,1] above the relationship arc near the SALARY entity indicates that each employee has a (snapshot) minimum and a (snapshot) maximum of one salary at any point time. The L[1,n] below it indicates that each employee has a (lifetime) minimum of one and a (lifetime) maximum of many salaries over time.

The TER diagram is converted to an E-R diagram based on the need to access past states of the relationships. If there is no need to access past states, then the relationship is rendered with the snapshot cardinality, i.e., as a static representation. If history is accessed relatively infrequently compared to the current sate, then two entities are added, one for the current state (having a start_date attribute) and one for history (having both start_date and end_date attributes), each having the appropriate cardinalities. If history is accessed about as frequently as the current state, then only one entity is added (again having start_date and end_date attributes). Both current and past states are represented by this entity. It is not specified if the date attributes represent *valid time* (when the change occurred) or *transaction time* (when the change was recorded in the database). The temporal relationship between Employee and Department is similarly developed. The resulting E-R diagram is then mapped to relations in the standard way. The most complex case for the relationship between EMPLOYEE and DEPARTMENT is shown in Figure 2.

TER extends the E-R model by differentiating snapshot and lifetime

Figure 2: ER diagram from the TER diagram requiring occasional access to history

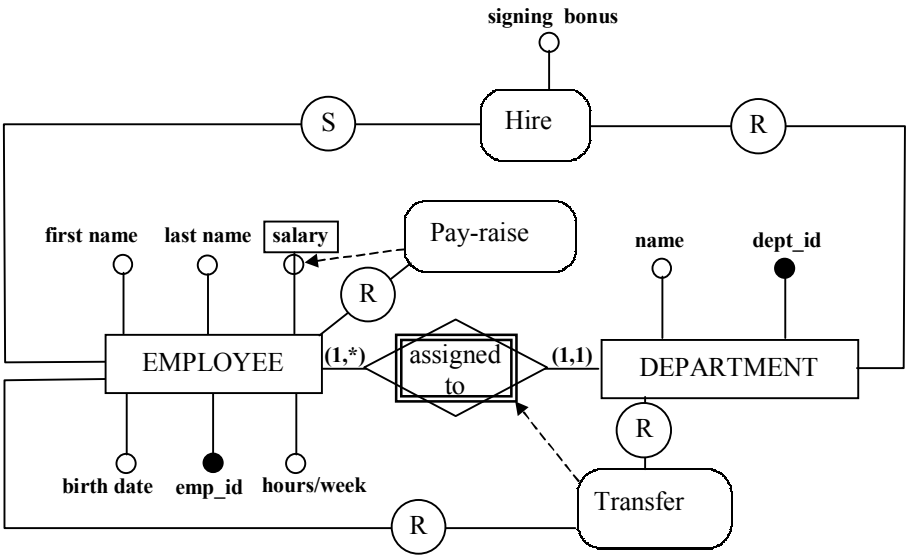


relationship cardinalities. In contrast, the Temporal Event Entity Relationship Model (TEERM) (Dey et al., 1995) adds semantic constructs for: *event, temporal relationship*, *quasi-static relationship*, *temporal attribute*, *quasi-temporal attribute*, and *causal arrows*. It redefines the standard symbols for relationships and attributes to be *static relationships* and *static attributes*, respectively. It provides rules designating how the new and redefined constructs are mapped to a first-normal-form, temporal-relational implementation. Figure 3 shows a TEERM diagram including the events Hire, Pay-raise and Transfer that represent the temporal dynamics of employees and departments.

A circled "R" on the line that connects an event to an entity indicates that the connected event can recur for the entity. The circled "S" indicates that an event is singular for a particular entity. Thus, Figure 3 expresses the business rules that a hire event relates an employee and a department, that a particular employee may only be hired once, and that a department may hire many employees. Temporal attributes (e.g., "salary") and temporal relationships (e.g., "assigned to") are boxed to indicate that their state histories must be maintained. The dashed arrows indicate causality and are used to specify business rules. In this model, events have identity, attributes, "relationships" with other entities, and a form of cardinality. In virtually every respect, they behave as entities; however, several new constructs are proposed to implement them.

We agree that it is important to identify events, but argue that representing them as entities instead of by new constructs clarifies the semantics of the application without sacrificing the simplicity of the E-R model. An examination of databases implemented without enlightenment from academic research on temporal databases reveals that this approach is frequently used in practice to address temporality. We will refer to this as the artifact approach to temporal design,

Figure 3: TEERM Diagram



and discuss its use and implications next.

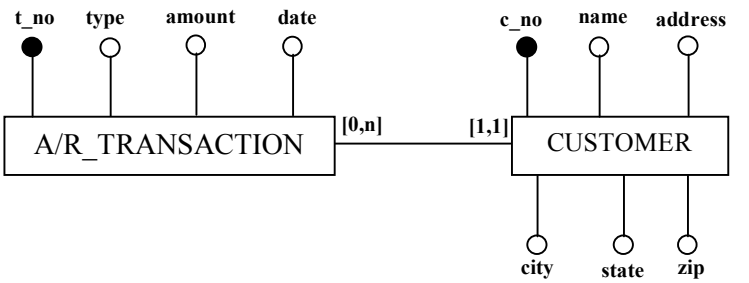
THE ARTIFACT APPROACH TO TEMPORALITY

Practitioners have long used E-R models to represent temporal semantics at the conceptual level. Consider, for example, a simple Accounts Receivable (A/R) application. From a business perspective, invoices are sent to customers when goods or services are delivered, customers make payments for those goods or services, and credits may be issued if goods are returned or damaged in delivery. These are *events* for which organizations typically create artifacts (standard business forms or documents) used in normal business processes. When a customer requires a "statement of account" the appropriate invoices, payments, and credits must be accessed. Hence it is critical to track the business transactions (events) over time. That is, the semantics of time must be explicitly represented. Figure 4 shows a simple E-R model for such an application. A customer participates in zero or more A/R Transactions (e.g., invoices, payments, credits). Each A/R Transaction must have exactly one participating customer. In this simple illustration, Customer is identified by customer number (c_no) and is described by the attributes name, address, city, state, and zip. A/R Transaction is identified by transaction number (t_no) and is described by the attributes type, date, and amount. From an accounting perspective there are at least two types of transactions, debits (when delivered goods or services are invoiced) and credits (when payments are made or credit is given for returned goods or services).

We argue that such a data model explicitly represents all of the necessary temporal dynamics of the application without requiring semantic constructs beyond those of the basic E-R model. The diagram alone, however, is not sufficient to specify all of the semantics of the application. As with all E-R diagrams the meaning of each entity and attribute must be explicitly defined as must the processing (rules) necessary to reconstruct the domain history. One important aspect of an entity definition is its membership criteria, particularly its temporal aspects. In this example, Customer is defined as "any person or organization to which goods or services have ever been provided or are intended to be provided, even if that entity has ceased to exist legally." A/R Transaction is defined as "any customer debit or credit exchange that has ever occurred."

Customer attributes are defined as "the most current values available for that

Figure 4: ER Model Based on Existing Documents (A/R Transaction)



customer." Hence the events affecting a customer's attributes and their state histories are deemed to be outside the scope of the system. A/R Transaction attributes are defined as follows. t_no is a system generated identification number. "type" has three legal values DI (debit invoice), CP (credit payment), and CM (credit memo). "date" is the date on which the transaction is recognized and "amount" is the amount of the transaction (positive for debits and negative for credits).

Note that there are numerous additional "transaction dates" that could be maintained, representing both actual time and transaction time. These include, for example, the date the transaction was postmarked, the date the corresponding business document was produced, and the date the transaction was recorded in the information system. Any or all of these may be important for the semantics of the application. The data modeler must determine which of these or other date related data are required. Each required date should be represented by an attribute in the data model.

Developers routinely develop applications that support temporality without extending the E-R model or the relational model. They do this quite naturally in situations where business processes generate artifacts that represent the causal events of the system. Thus an invoice, a remittance advice, and a credit memo are all documents that record events that change a customer's account balance. These are generalized to the well-understood accounting concept of transaction, which is represented at the conceptual level.

Clearly, this approach uses traditional E-R constructs (which can be mapped to traditional relational constructs) to model a system that supports temporality. Conversely a snapshot representation would view customer A/R balance as a "temporal" attribute of Customer, potentially ignoring the A/R transactions that cause it to change. Each time a transaction occurs, its value would be updated and its history maintained structurally. While this enables the production of customer balance and balance history, it misses the true semantics of the application required for the production of customer statements. It records only the fact that the balance changed. It does not differentiate types of transactions, which explain why the balance changed (e.g., credit versus payment). That is, a customer statement requires a transaction history not just a balance history.

Hence a snapshot perspective is inappropriate for this application. Customer A/R account balance is not just a "temporal attribute." It is the cumulative result of a set of transaction events and its value at any point in time is determined from those transactions. The representation of events is necessary and sufficient to provide the full range of its temporality. However, for efficiency reasons customer A/R balance may be maintained as a (calculated) data item (column) describing Customer and the transaction history may be truncated to a manageable length of time with removed transactions archived for the requisite legal period. There are a number of implementation alternatives. The A/R balance data item may, for example, contain the "balance forward" from archived transactions, it may contain the "current balance," or the balance forward from archived transactions may be implemented as a special transaction type. In any case transaction history can be reproduced for the period of time covered

by the remaining A/R transactions. These are design decisions made for efficiency reasons. They do not reflect the semantics of the application.

Temporal queries can easily be created to produce a customer's account balance at any point in time. For example if the data model in Figure 4 is implemented directly in relations then the following SQL query can be used to produce the account balance for customer number 1234 on June 1, 2002 (this query becomes slightly more complicated if for efficiency reasons the transaction history is truncated):

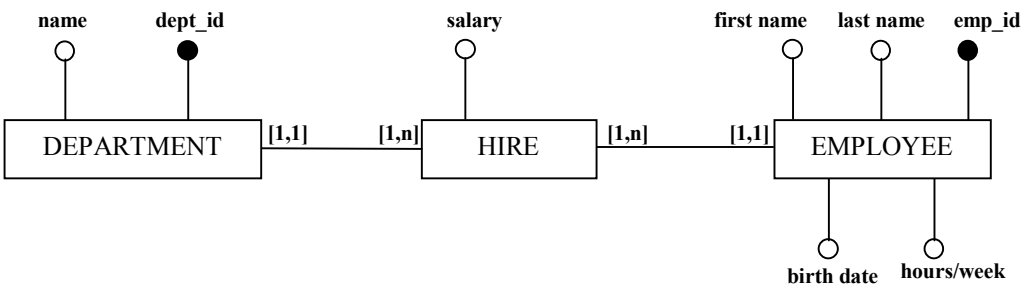
```
SELECT c_no, SUM(amount)
FROM Customer, A/R_Transaction
WHERE Customer.c_no = A/R_Transaction.c_no
and date <= '06/01/2002'
and Customer.c_no=1234
```

An event representation, facilitated by the existence of institutional artifacts, is a natural choice for an A/R application where the transactions affecting customer A/R balance are obvious and well documented. It is extremely unlikely that an analyst would choose to represent A/R balance as a temporal attribute without maintaining the underlying transaction (event) data. In situations where the set of causal events is not available to the system or where information about those events is not needed, then temporality can be represented as a set of state histories as discussed in the temporal data modeling approaches. However, when users need to understand "why" a particular state exists, then events must be recorded in some manner.

THE EVENT APPROACH TO TEMPORALITY

Unlike the artifact approach, the event approach does not rely on existing documents to represent temporal dynamics. Instead, this approach seeks to formally identify the events that cause state-changes and the "things" that are affected in the system being modeled. In this view things and events are inextricably intertwined. Things do not exist without events and events do not exist without things. Events cause things to change state. The event approach has its foundation in the work of McCarthy (1982), Geerts and McCarthy (2002), and Denna et al. (1993). It is fundamentally different from virtually all approaches in the literature on temporal databases because it does not seek to identify temporal attributes or relationships or to record past states of a system. Instead, it seeks to identify and represent all events that occur and the "things" (resources, agents) affecting and affected by them. When documents exist for all

Figure 5: ER Model Based on Events

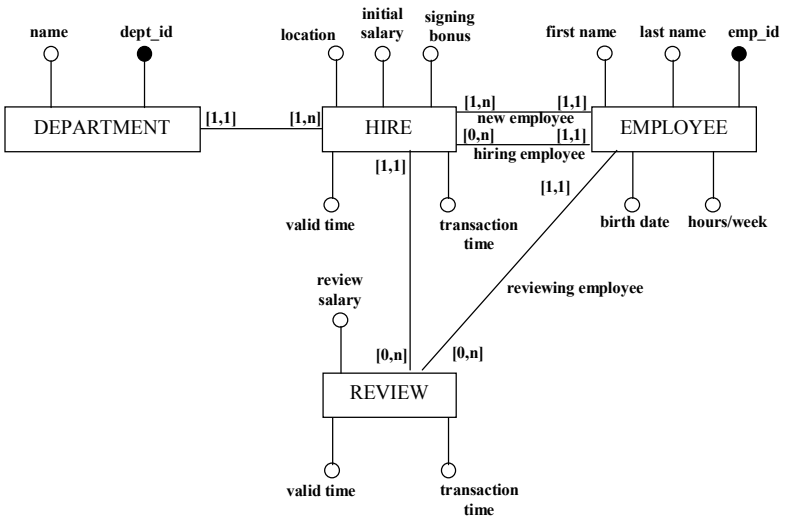


important events, the resultant data model may be indistinguishable from that produced using the artifact approach. However, it differs from the artifact approach in that it explicitly analyzes the (economic) resources, events, and agents that constitute business processes, not the artifacts that represent them.

Furthermore, McCarthy, as an accountant, recognized the inherent duality of economic events within an organization — for every debit there must be a balancing credit. Wand and Weber (1995) recognize a similar notion with the concept of stable and unstable states. When an event occurs that causes the system to move to an unstable state, at least one other event must occur to move the system back to a stable state. For example, the event "customer places order" causes the system to enter an unstable state. Although, strictly speaking, no economic (financial) event has occurred the organization has an obligation to fulfill that customer's order. In this way the existence of an unfilled customer order initiates the order fulfillment business processes. This process culminates in the delivery of and payment for the ordered goods, a stable state in which all obligations for the order have been fulfilled. Although it may not be important for the implemented information system to track all of the events that happen in such a business process an analyst must identify all of them in order to make that determination.

Consider the events that occur with respect to employment within an organization. Managers in the Human Resources area are familiar with events such as hire, promote, transfer, review performance, quit, terminate, etc. and the business rules associated with them. An employee is "created" (or more specifically the employee "role" is created for a person) and assigned to a department by a "hire" event. This event establishes the initial values for the employee's attributes and relationship as illustrated in Figure 5. It also creates an obligation or an unstable state in the sense that an employee is employed for some purpose - the employee is obligated to work; the organization is obligated to evaluate that work, etc. The analyst must determine which events are

Figure 6: A More Complete Event ER Model



within the scope of the information system under development. In particular, it may be important to recognize that an event occurred, even if the event did not result in any changes (e.g., an employee salary review with no salary change).

An event analysis facilitates the identification of additional facts related to the event. That is, events address the question of "why" a change occurred. Events not only happen at a point in time (when); they also happen at a location (where Denna et al., 1993); there are generally agents or actors who initiate or are involved in the event (who); there are often resources consumed by the event or required for the event to transpire (what and how). To the extent that these other items of information are of interest, they must be represented in the data model. For each event, then, an analyst must determine the associated resources, agents, and locations. This provides a checklist for assuring that the relevant information requirements are more completely determined.

If the users of the system need also to know when an event occurred (valid time) and when event data were recorded in the system (transaction time) both should be included in the data model. Figure 6 shows a more complete event representation as in Figure 5, including additional hire event information (location, signing bonus, valid and transaction time, and the employee responsible for managing the hire event) and the additional event, review, occurring when an employee is reviewed.

This model includes all of the temporal aspects modeled in the TER and TEERM diagrams in Figures 2 and 3, respectively. The one-to-many relationship between hire and employee represents the temporal nature of employees working in departments as in the lifetime many relationship between employee and department in the TER model and the hire and transfer events in the TEERM model. The review entity represents the lifetime many relationship between employee and salary in the TER model and the Pay-raise event in the TEERM model.

Once the creation or "coming to be" event (Bunge, 1977) is established for an employee it is appropriate to analyze additional events that cause changes in the employee's identified attributes and relationships (properties) relevant to the domain under investigation. This enables the analyst to derive the "narrative" of the business application, evoking human cognitive processes related to problem solving and sense making (Robinson and Hawpe, 1986; Pillemer, 1998). The representation "tells the story" of the employee through events such as hire, transfer, promotion, performance review, quit, and terminate. Each event is a separate entity, possibly interconnected in an abstraction hierarchy and interrelated through temporal constraints, e.g., an employee cannot participate in a performance review after a terminate event unless there is an intervening hire event. In any case, this approach models the events that cause state change rather than merely the sequence of past states and produces a more understandable representation for users and analysts.

A natural byproduct of this approach is that the resulting database has information about *why* a state exists. Perhaps more importantly, by recording data at the event level, the system has data in an elementary from which can be used later to query states that have become important but were not anticipated at the time the system was developed. For example, consider an inventory

management system that is initially modeled to maintain the state history of quantity on hand for each product. Since quantity on hand for any product is a value that changes because of many different events, the information available in the event approach is substantially richer than that available in any of the state-history approaches. In the state-history approach, when a product is received (or made), a history is retained to indicate the change to quantity on hand with corresponding valid-time and, perhaps transaction-time, likewise when a product is sold.

In the event approach, the events that change the quantity on hand are recorded. Thus, the fact that a product is received is recorded. The fact that a product is sold is recorded. The fact that a product is shipped is recorded. For convenience or efficiency, a quantity on hand attribute may be represented to indicate the current state, but its history would be recorded implicitly in the event data. This distinction becomes important when the "state of interest" changes. Suppose that users of the system redefine quantity on hand from "quantity available for purchase" to "physical amount of inventory on the shelf" (regardless of whether it is committed to an order) and require quantity committed defined as "sold but not yet shipped." In a state-history approach, a new attribute, quantity committed, would need to be created and the old attribute, quantity on hand, would take on different semantics. Furthermore, the system would be powerless to distinguish between the two values for any time before the new attribute was created. In the event approach, quantity committed can be calculated from the relevant events with no change to the temporal structure. This elucidates a critical distinction between business systems and other systems which may be better suited to the attribute versioning approach of most temporal database research.

In business systems, the large majority of the "states of interest" are artificially defined to be summaries of specific sequences of events. Unlike natural systems in which events occur to bring about observable states of things, these artificially defined states (such as profit, expenses, and quantity available for sale) cannot be observed and must be imputed from a set of observed events. In natural systems, the observed state is often the result of very complex (and ill-understood) transformation functions involving a very large number of often unobservable events. For systems that track the health of individuals over time, state-versioning systems are very appropriate because all of the events that bring about the state of an individual's health are not observable. Moreover, the relationship between those events and the observed health is not perfectly understood. In such a system, seeking to record events and then using those events to calculate past states for individuals' health is simply impossible. Instead, state is recorded as it is observed as kept as a state history. This approach is quite adequate for such a system because 1) state is observed, 2) events are not always observed, and 3) the relationship between state and events is complex and unclear. In business systems, just the opposite is true. State is most often not observable, but the events that cause state are. More importantly, the relationship between the events and the state is simple and very clearly understood because it is defined. In many cases, the definition of states is generally accepted as a matter of standard business terminology. For example, the state termed "revenue" is simply the sum of sales for a given period of time. The manner in which events

combine to yield other states needs to be documented in some fashion. Clearly, such documentation lies outside the scope of the E-R model.

In situations such as inventory control where sales, shipments, receipts, manufacturing and shrinkage all combine to yield a single value for the attribute "quantity on hand" recording state histories rather than events dramatically limits the information available from the database. Additionally, changes in the way users want to view data require changes to the database structure. Since the event approach seeks to record the events in their elemental form, changes to the way users want to view data do not often mandate changes to the database structure; rather, they require new queries or views to combine the event data to meet the new information requirements.

Analysts and designers choose how an information system will represent things, states, and events according to the domain of interest. Determining the current state of a thing from an event sequence requires the specification of the transformation (function) for each state and event in the sequence (Wand and Weber, 1995). When such transformation functions are invertible then the state-history and the event representations are equivalent. Often, however, the transformation function does not have an inverse; the same state history can be generated from different sets of events. In this case the designer must determine the appropriate representation. An information system can be designed to capture and maintain only the current state of the things of interest when state-history, events, and transformation functions are deemed to be outside the scope of interest. At the opposite extreme an information system can be designed to capture and maintain data about all of the events in which a thing is involved and the transformation functions relevant for a thing and calculate its current state and state-history when needed. Frequently some middle ground is most appropriate where some combination of state, event, and transformation function data is maintained.

Consider, for example, the General Ledger of a company. A General Ledger has some number of Accounts (things). Suppose each Account has three properties (attributes), "name" specifying the text to be used to describe the account, "type" designated as Asset, Liability, Capital, Revenue, or Expense, and "balance" specifying the current dollar value ascribed to that account. The account name is specified once when the account is created and may be changed when deemed necessary. It may or may not be important to keep track of the history of that attribute. It is very unlikely that an information system would record the business event or transformation function that lead to the change. Account type is also specified when an account is created, however, it is not allowed to change.

When business events (transactions) affecting the financial position of the company occur, accounting practice requires the effects of those events on appropriate account balances to be recorded. These transactions are recorded as debits and credits. The transformation rule is simple and invertible. For Asset and Expense accounts, account balance is the sum of debits minus the sum of credits. For Liability, Capital, and Revenue accounts, account balance is the sum of credits minus the sum of debits. A General Ledger system required to report account balances may record transactions (events) or it may maintain only the current balance (the state resulting from applying the

transactions). In both cases the transformation function must be applied. In the former design the transformation is applied on the output side, using recorded transactions to calculate the balance when needed. It requires maximal storage and processing but provides maximum flexibility. In the latter design the transformation is applied on the input side, updating the recorded data when a transaction occurs. It requires minimal storage space and minimal processing at report time. However, it does not allow for causal analysis of the General Ledger application and is clearly an inappropriate design for auditing purposes. A typical General Ledger design includes components of both. The value of "balance" is recorded for each account at a specified point in time (e.g., at year end) and transactions (events) are recorded from that point in time to the present.

In business information systems, we observe events but often define transformation rules to create artificial, unobservable states which are often of primary interest. For example, virtually all financial data regarding a company's performance are artificial states defined by transformation rules that aggregate events in various ways. A company's revenue for a period is defined as the sum of the price of all goods or services sold during that period. Revenue cannot be directly observed. It is only through the application of the transformation rule to a set of events that a value for revenue can be determined. The same is true for expenses, assets, shareholders' equity and many other financial attributes of an organization. Only the very few measures of a company's performance have states that can be directly observed, e.g., physical inventory, bindings, equipment. However, even in those cases the value of those observables is calculated based on events, e.g., when purchased, legal depreciation functions.

CONCLUSIONS AND FUTURE RESEARCH

The fundamental purpose of a data model is to faithfully represent the reality of the business system under investigation. If the business system requires only a single state (presumably the most recent state recorded in the database), then a static data model is adequate. If it requires state-history, then the data model must reflect that history. If it requires a causal history then events must be represented. We argue that event semantics are sufficient for representing the temporal dynamics of business systems and that adding additional constructs or treating temporal dynamics structurally as suggested by current temporal database research obscures rather than elucidates the semantics of the application. Temporality is semantic not structural. Representing events is a natural mechanism for representing the semantics of temporality.

Virtually all data are temporal in the sense that they reflect the effects of events that have occurred in the world. While it may not be necessary to represent all of these events in a data model, failure to recognize this fact can lead to significant errors of omission in the determination and representation of information system requirements. We have argued that an event analysis can provide an appropriate framework for addressing this issue.

Because the event approach represents events rather than states, implementing an events design can be processor and storage intensive. Because of the predominance of current state over other states in most database applications,

current state is often maintained in parallel with the event stream. This allows quick access to the most recent data; however, it does little for past states. One area of future research is to determine efficient algorithms for summarizing an event stream. Some of the work on the physical implementation of temporal databases may have direct implications in this area. Tsotras, et al. (1995), for example, identify indexing schemes that efficiently reconstruct state at a point in time by summarizing changes to state values. Additional research in this area is needed.

Whenever a database records two views of the same reality (as is the case when a system maintains both an event stream and current state) there is the possibility that the database is internally inconsistent. That is, the summarized event stream may show one state, while the maintained current state may show another. This "state anomaly" is obviously an undesirable condition. One way to avoid it is to disallow application programs from updating current state, instead through the use of triggers in an active database environment, enable the DBMS to maintain current state. However, this requires that transformation functions be specified in two places — first in the system used to summarize an event stream and second in the definition of the triggers that maintain current state. Research is needed to identify ways to specify the relationship between an event and state variables once and to allow that specification to be used to compute past states as well as to maintain current state.

Data models that represent events are very different from those that represent states and state history. It is unclear if one of these models is easier for analysts and users to understand or if there is no difference. Further, it is unclear if a business system is more easily modeled using either approach or if business managers are better able to validate one representation or the other. Although some research has been conducted in this area (Allen, 2001) published results are inconclusive—clearly more research is needed.

END NOTE

[1] This paper is an extension of research originally published in (March and Allen, 2003).

REFERENCES

Allen, Gove N. (2001). *The Effect of Event Organization in Database Representations on User Data Retrieval Performance,* Ph. D. Dissertation. University of Minnesota, Minneapolis.

Bunge, M. (1977). *Ontology I: the Furniture of the World, volume 3 of Treatise on Basic Philosophy*. Boston: D. Reidel Publishing Company.

Chen, P. P-S. (1976). The Entity-Relationship Model - Toward A Unified View Of Data. *ACM Transactions on Database Systems*, 1(1), 9-36.

Clifford, J., & Tansel, A. U. (1985). On An Algebra For Historical Relational Databases: Two Views. *Proceedings of ACM-SIGMOD 1985 International Conference on Management of Data*, Austin, Texas. New York: ACM Press, pp. 247-267.

Dean, T. (1989). Using Temporal Hierarchies to Efficiently Maintain Large Temporal Databases. *Journal of the ACM*, 36(4), 687-718.

Denna, E., Cherrington, J.O., Andros, D. & Hollander, A. (1993). *Event-Driven Database Solutions*. Homewood, IL:

Business One Irwin.

Dey, D., Barron, T., & Storey, V. (1995). A Conceptual Model for the Logical Design of Temporal Databases. *Decision Support Systems*, (15), 305-321.

Etzion, O., Jajodia, S. & Sripada, S. (Eds.) (1998). *Temporal Databases: Research and Practice*. Berlin: Springer-Verlag.

Gadia, S. K. & Vaishnav, J. (1985). A Query Language for a Homogenous Temporal Database. *Proceedings of the 4th Annual ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, Portland, Oregon. New York: ACM Press, pp. 51-56.

Gadia, S., & Yeung, C. (1988). A Generalized Model for a Relational Temporal Database. *Proceedings of the ACM SIGMOD Conference on Management of Data*, Chicago, Illinois. New York: ACM Press, pp. 251-259.

Gadia, S. (1988). A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, (13)4, 418-448.

Geerts, G. & McCarthy, W. E. (2002). An Ontological Analysis of the Primitives of the Extended-REA Enterprise Information Architecture. *The International Journal of Accounting Information Systems*, (3)1, 1-16.

Gregersen, H. & Jensen, C. S. (1999). Temporal Entity-Relationship Models - A Survey. *IEEE Transactions on Knowledge and Data Engineering*, (11)3, 464-497.

Kouramajian, V., Kamel, I., Elmasri, R., & Waheed, S. (1994). The Time Index+: An Incremental Access Structure for Temporal Databases. *Proceedings of the Third International Conference on Information and Knowledge Management*, Gaithersburg, Maryland. New York: ACM Press, pp. 296-303.

Kumar, A., Tsotras, V., & Faloutsos, C. (1998). Designing Access Methods for Bitemporal Databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(1), 1-20.

March, S. T. and Allen, G. N. (2003). Temporal On the Representation of Temporal Dynamics. In: Siau, K. (Ed.) *Advances in Database Management* (pp. 37-53). Hershey: Idea Group Publishing, 2003.

McCarthy, W. E. (1982). The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *The Accounting Review*, 58(3), 554-578.

McKenzie, E. & Snodgrass, R. (1987). Extending the relational algebra to support transaction time. *Proceedings of the ACM SIGMOD Conference on Management of Data*, San Francisco California, New York: ACM Press, pp. 467-478.

Ozsoyoglu, G. & Snodgrass, R. (1995). Temporal and Real-Time Databases: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4), 513-532.

Pillemer, D. B. (1998). *Momentous Events, Vivid Memories*, Cambridge, MA: Harvard University Press.

Robinson, J. A. and Hawpe, L. (1986). Narrative thinking as a heuristic process. In: T. R. Sarbin (Ed.), *Narrative Psychology: The Storied Nature of Human Conduct* (pp. 111-125). New York: Prager Publishers.

Snodgrass, R. (2000). *Developing Time-Oriented Database Applications in SQL*. San Francisco: Morgan Kaufmann.

Tansel, A., Clifford, J., Gadia, S., Jajodia, S., Segev, A., & Snodgrass, R. (1993). *Temporal Databases: Theory, Design, and Implementation*. Redwood City, CA: Benjamin/Cummings.

Tauzovich, B. (1991). Toward Temporal extensions of the Entity-Relationship Model. *Proceedings of the 10th International Conference on Entity Relationship Approach*, San Mateo, California: E-R Institute, pp. 136-179.

Tsotras, V., Gopinath, B., & Hart, G. (1995). Efficient Management of Time-Evolving Databases. *IEEE Transactions on Knowledge and Data Engineering*, 7(4), 591-607.

Wand, Y. & Weber, R. (1995). On the Deep Structure of Information Systems. *Information Systems Journal*, (5), 203-233.

Wang, X., Bettini, C., Brodsky, A., & Jajodia, S. (1997). Logical Design for Temporal Databases with Multiple Granularities. *ACM Transactions on Database Systems*, 22(2), 115-170.

Gove N. Allen holds bachelors and masters degrees in Accountancy from Brigham Young University. For three years, Dr. Allen operated a small consulting firm specializing in client/server database management and design, and the surrounding technologies, winning contracts from both state and federal government agencies as well as providing training and development services for major corporations such as Sony, AT&T, Sprint, Hewlett Packard, Micron, Intel, and American Express. More recently, he developed (and continues to support) WebSQL.org, a site for teaching database management that allows dynamic execution of Structured Query Language against Oracle and Microsoft SQL Server databases through a simple web interface. Dr. Allen received his Ph.D. from the University of Minnesota in 2001 and currently serves as an assistant professor of e-business and information systems at Tulane University's A. B. Freeman School of Business.

Salvatore T. March is the David K. Wilson Professor of Management at the Owen Graduate School of Management, Vanderbilt University. He received a BS in Industrial Engineering, MS and PhD degrees in Operations Research from Cornell University. His teaching responsibilities and research interests are in Information System Development, Conceptual Modeling, Distributed Database Design, Electronic Commerce, and Object-Oriented Development Tools and Methodologies. His research has appeared in journals such as *ACM Computing Surveys*, *ACM Transactions on Database Systems*, *Communications of the ACM*, *IEEE Transactions on Knowledge and Data Engineering*, *The Journal of MIS*, *Journal of Database Management*, and *Information Systems Research*. He served as the Editor-in-Chief of *ACM Computing Surveys* and as an Associate Editor for *MIS Quarterly*. He is currently a Senior Editor for *Information Systems Research* and an Associate Editor for *Decision Sciences Journal*, *Journal of Database Management*, *Information Systems and e-Business Management*, and *Information Systems Frontiers*.