A Temporally Oriented Data Model

GAD ARIAV
New York University

The research into time and data models has so far focused on the identification of extensions to the classical relational model that would provide it with "adequate" semantic capacity to deal with time. The temporally oriented data model (TODM) presented in this paper is a result of a different approach, namely, it directly operationalizes the pervasive three-dimensional metaphor for time. One of the main results is thus the development of the notion of the *data cube*: a three-dimensional and inherently temporal data construct where time, objects, and attributes are the *primary* dimensions of stored data. TODM's cube adds historical depth to the tabular notions of data and provides a framework for storing and retrieving data within their temporal context. The basic operations in the model allow the formation of new cubic views from existing ones, or viewing data as one moves up and down in time within cubes.

This paper introduces TODM, a consistent set of temporally oriented data constructs, operations, and constraints, and then presents TOSQL, a corresponding end-user's SQL-like query syntax. The model is a restricted but consistent superset of the relational model, and the query syntax incorporates temporal notions in a manner that likewise avoids penalizing users who are interested solely in the current view of data (rather than in a temporal perspective). The naturalness of the spatial reference to time and the added semantic capacity of TODM come with a price—the definitions of the cubic constructs and basic operations are relatively cumbersome. As rudimentary as it is, TODM nonetheless provides a comprehensive basis for formulating an external data model for a temporally oriented database.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design—*data models*; H.2.3 [**Database Management**]: Languages—*data manipulation languages* (*DML*); *query languages*

General Terms: Design, Languages, Theory

Additional Key Words and Phrases: Information modeling, information systems, relational model, time dimension

1. INTRODUCTION

Time and memory are inseparable concepts, and indeed, the study of data models and DBMSs that capture and preserve the inherent dynamics of database content is one of the major thrusts in the research into time and information systems. The intensifying research activity around time and databases is fueled by the observation that all human activities are embedded in time, yet it is very difficult to explicitly formulate this temporal context, and it is therefore rarely reflected

This research was supported in part by the ONR under contract N00014-75-0462 and by DARPA under contract MDA93-80-C-0093, with the University of Pennsylvania.

Author's current address: Faculty of Management, Tel Aviv University, Ramat Aviv, Tel Aviv 69978, Israel.

in the design of computer-based information systems. Roughly 40 references to research into time and databases have been recently identified and annotated in [8], and about a dozen more have since been reported. The underlying premise of this expanding body of research is the recognition that time is not merely another dimension, or another data item tagged along with each tuple, but rather something human users treat in very special ways. The results of this research reinforce the perception that designing temporal features into information systems requires new and different conceptual and practical tools. Time is a universal dimension of data management applications, and thus the focal point of the various attempts in this domain has been the conceptualization and design of a *general-purpose* system that deals with data dynamics in an explicit fashion.

The need to augment our view of data management with the ubiquitous time dimension has been specifically reinforced by the concept and the growing usage of Decision Support Systems (DSS) [23]. The capacity to deal adequately with time and historical information is a core requirement in DSS when these support business planning, the investigation of causal relationships, and Retrospective Analysis [2]. This conjecture has been recently supported by a direct practitioners' reference to the subject [29].

Such an advanced support for managerial decision and control is not easily achieved with current typical systems. In general, none of the three major data models explicitly addresses temporal or historical aspects of the data. In the practical realm of information systems, time aspects are usually either neglected, treated only implicitly, or explicitly factored out [34], in spite of the abundance of temporal references in common data. As a result, most information systems and generalized data management tools do not treat "present" data and "past" data symmetrically, but typically differentiate between them in terms of accessibility—both logical and physical. The operational database in the core of many information systems is a "thin," tenseless, and temporally inconsistent snapshot of latest available data. Data items within an instance of such a database pertain to various points in time, and newly recorded data eventually *replace* previously recorded ones. If kept at all, "forgetfully" replaced values are usually retained on "log" files, meant mainly to allow recovery of damaged data. All these practices imply substantial limitations on the range and economic feasibility of historical inquiries and "what if" analyses that information systems can support.

Beyond the perplexing conceptual complexity involved, attempts to implement historically rich databases suffered from the absence of adequate means to handle the huge volumes of secondary storage needed for maintaining "complete histories." The practical relevance of temporal data models like TODM is therefore contingent upon recent hardware developments that loosen some of these traditional constraints. A particularly promising trend in this respect is the development of optical mass-storage devices, which offer both immense *and* directly accessible data storage facilities [20]. Storage capacity of optical devices presently under development ranges from 1.25 to 4.00 and 5.00 Gbytes (1 gigabyte is equal to 1024 megabytes), whereas in general read and write times for optical and magnetic disks are expected to be roughly the same [12, 17]. Further development efforts are concentrating on devices with multiple optical disks ("jukeboxes") with storage capacity from 1250 Gbytes to 2000 Gbytes, at maximum access time below 5 seconds [25, 31]. Prices for commercially available optical storage (e.g., Drexon II) are already below the price per bit on magnetic tape—traditionally the least expensive storage medium directly manipulable by computers.

The observations made so far highlight the need for, and the practicality of, reexamining the traditional distinction between operational ("on-line," interactive, short-term) and archival ("off-line," historical, "never-forgetting") databases and applications. The research documented in [4] has focused on this broad issue and examined the design of an information system that *preserves* the inherent dynamics of its content and makes the time dimension of data accessible to its users. This paper summarizes the results in the domain of data modeling, while the above-mentioned research treats also the complementary issues of DBMS implementation and the design of an integrated temporally oriented graphic user interface.

It is often overlooked, but—besides the formal robustness of the relational model—a strong argument is also made for the "naturalness" of its reference to data [15]. Similar motivation underlies the construction more recently of the so-called "semantic data models" [3]. The temporally oriented data model (TODM) is an attempt to relate to temporal data in the same fashion, namely, to describe the temporal aspects of data in the most "natural" and "comfortable" way through a three-dimensional construct. The use of this metaphor is supported by abundant anecdotal evidence (e.g., almost all the research on temporal data models makes a reference to such a mental image) and a more rigorous psychological evidence (e.g., [1]). The latter suggests that the cubic form is the favorite temporal metaphor for presenting temporal relationships, and that a vertical orientation of the time axis is the least ambiguous for identifying such relationships.

This paper elaborates on the conceptual underpinnings of TODM. Section 2 is an informal introduction to the model and the underlying research issues. The model is then presented more formally in the next three sections: the data constructs in TODM (Section 3) operationalize the pervasive spatial metaphor for temporal relationships and views; the preliminary set of operations on these constructs (Section 4) provides an immediate and intuitively appealing view of the data in temporally oriented (or historical) databases; and the accompanying constraints address the nature of time attributes and functional dependencies over time (Section 5).

The abstract data model presented in Sections 3, 4, and 5 is used in Section 6 as a guideline in designing the temporal database query language TOSQL, the temporally oriented SQL. The discussion focuses specifically on an SQL-like *specification* retrieval operation suggested by TODM. The area of temporal data models is relatively young, and there are no established standards of quality by which proposed models can be judged. The discussion in Section 7 attempts a preliminary evaluation by means of contrast: TODM is compared with its significant, mostly contemporary, "competitors." Section 7 serves therefore also as a brief survey of the research to date on the subject. The concluding Section 8 identifies some major areas where further research is urgently needed.

2. MOTIVATION: A TEMPORALLY ORIENTED VIEW OF DATA

This section introduces the key issues underlying the research of temporally oriented views of data. Consider, for example, a database serving the maintenance crews of some airline, containing a single relation, AIRPLANE, with the

Fig. 1. The AIRPLANE database at 8:30 A.M.

PLANE-ID	MAKE	STATUS
AA-003	DC10	5
EA-218	L1111	0
LY-340	B747	0
TW-101	B747	1

Fig. 2. The AIRPLANE database at 9:45 A.M.

PLANE-ID	MAKE	STATUS
AA-003	DC10	5
EA-218	L1111	0
LY-340	B747	0
TW-101	B747	0

Fig. 3. The AIRPLANE database at 10:30 A.M.

PLANE-ID	MAKE	STATUS
AA-003	DC10	3
EA-218	L1111	0
LY-340	B747	4
TW-101	B747	0

attributes PLANE-ID, MAKE, and STATUS. PLANE-ID identifies the individual aircraft, and STATUS indicates the flight-readiness of the plane (e.g., 0 may mean the plane is flight-ready, 1 may indicate a defect in a popcorn heater, while 6 may mean a cracked wing or some other grounding problem).

At some point in time, say at the beginning of the morning shift at 8:30 A.M., the extension of this relation (i.e., the actual recording in the database) includes four objects (i.e., airplanes), as reproduced in Figure 1. About 5 minutes later a member of the maintenance crew consults the database to find the status of TW-101 and finds out '1'. The supervisor comes yet another 10 minutes later, asks for his favorite problem statistics report and finds out the distribution of problems at the moment, specifically, one problem of type 5 and one of type 1.

An hour later the TW-101 is brought back to perfect flying condition, and a corresponding transaction is entered to the central database, resulting in the extension reproduced in Figure 2. About 15 minutes later another curious member of the maintenance crew inquires about the status of EA-218 and finds out that the aircraft is flight-ready.

At 10:30 A.M. two reports are received in the maintenance control room: first, that the AA-003 is now partially repaired, and therefore its status should be reset to 3; and, second, that some problems have been discovered during the routine preflight check of LY-340. Its status is thus set to reflect some uncertainty about its availability for the next scheduled flight. The extension of the database after all these reports had been properly entered is reproduced in Figure 3.

Even though we imagine that in real life somewhat more complex databases might possibly exist, this rather simple example is sufficient for demonstrating some pervasive principles that define the notion of time in information systems. Specifically, there are three—typically implicit—temporal conventions or assumptions about the relationship between a database and its "environment" that underlie both the design and the use of *all* databases. (The first two are

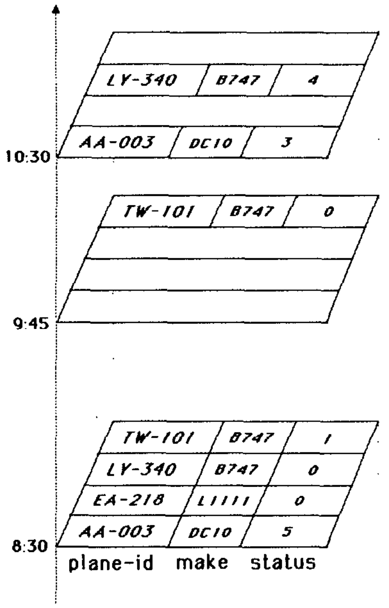

Fig. 4. The temporal path of the AIRPLANE database.

similar to the Comprehension Principle and the Continuity Assumption in [13].) The three, stated rather informally, are as follows:

Temporal completeness. The discrete and finite set of successive database instances (or equivalently the set of recorded events that bring about the transitions in the states of the database) *completely* describes the development of the enterprise—as it is captured by the corresponding data model—over the entire period of time covered by the database.

Temporal density. The state of the database, or of any object in it, at any point of time *during* the period between two successive events can be *inferred* from available database states. The most common rule for derivation of such intermittent states is that a recorded value prevails until changed by the recording of a subsequent event.

Temporal isomorphism. The order and pace in which the database evolves correspond to an order and a pace associated with events in the modeled environment. In particular, there is tight correspondence between the database and the temporally concurrent reality it is aimed to capture.

These assumptions or principles are so ingrained in our perceptions of information systems that we are rarely aware of the fact that they are being applied so abundantly. The example above most likely did not explicitly invoke any of the three principles, even though they underlie every step in it. The three complementary assumptions explicate the fundamental notion of time in databases, *regardless of whether these databases preserve the time dimension or not.* The temporal path through which the database has progressed can be represented as a three-dimensional object (a cube), utilizing the pervasive spatial metaphor for time. The scenario in the above examples is depicted in Figure 4.

The dimensions of this cube are objects and attributes (both in the standard relational meaning), and a dense and continuous representation of the temporal dimension of the relation. The common database provides at each point in time only a cut through its temporal path, while the temporally oriented (or historical) database captures its temporal path in its entirety. The latter type of database therefore contains data spaces or "cubes of data." TODM, the data model discussed in this paper, is of that nature. The state of the relation at any point in time is determined by slicing the cube horizontally in a depth corresponding to the specified time, and observing the values of data that prevail for each of the objects represented in the relation.

It should be noted that temporal isomorphism relates to *an* order and *a* pace. In particular, the same information that makes up a cube along one type of time most likely makes up alternative cubes—adhering to the three principles, but along different types of time. For instance, the cube of events along their database recording time may differ from the cube of the same set of events along their time of observation. The last two events in the above example were recorded during the same update pass and bear identical recording time stamps, but they most likely were observed and noted at different times. Even though we usually tend to perceive only a single time, and therefore equate the database recording time with the time of event occurrence, this may not necessarily hold in general.

What can a user do with the whole additional dimension bestowed upon him or her with the temporally oriented database? At the elementary level, a user should be able to move *upwhen* and *downwhen* in time[1] and view the database at various points of time, along the time dimension that defines the cube. In particular, users can view the uppermost surface of the cube, namely, a view that resembles the "most current" content of the common time-varying database. More elaborate operations should allow the user to derive and form new cubic views (i.e., extract and rearrange cubes by operating on any combination of dimensions—one could select attributes, and/or objects, and/or periods of time). More advanced operations should allow the user to join views, of possibly different points in time, and thereby create totally new cubes.

The temporally sensitive management of data could be, and indeed has been, examined from a number of complementary perspectives. Three of these are directly relevant to this paper, namely, (1) the treatment of time in conceptual data models, relating to the fundamental role of time in data modeling; (2) temporal data models that are semantically capable of dealing with the temporal aspects of stored data; and (3) query language constructs for the manipulation of temporal databases by their users.

The major efforts in category (1) are those conceptual data models that treat time as a primary conceptual component of data. Prominent among them have been the infological data model [27, 28] (interestingly, the earliest of the conceptual data models) and its refinement in [9]. A much broader perspective of information modeling and time as a modeling construct was developed in [10], and more recently the temporal extension of the Entity-Relationship model (TERM) [26].

[1] Our directed temporal expressions have been explicitly inspired by Asimov's *The End of Eternity*, Fawcett Crest Books, 1955.

The research that falls under category (2) has addressed the high-level organization of storage of temporal data, and the operations needed for their meaningful manipulation. The departing point for these works is exclusively the relational model; specifically they examine modifications and enhancements to this model necessary for allowing it to deal "adequately" with time information and related views. Major efforts in this category are the Historical Database (HDB) model [13, 14] and the Temporal Data Model (TDM) [5].

Under category (3) we include research efforts in the area of time and databases that have focused on query language constructs useful in manipulating data that contain temporal references. The emphasis in these works is typically on user access to data, and less on the data model. The major efforts among them are Legol 2.0 [21], the Time Relational Model (TRM) [7], and TQuel, a temporal extension of the Quel query language [33].

The classification of any research along the above "categories" is rather arbitrary, and based on subjective perception of the major contributions of the respective works. A brief summary of these efforts and a critical comparison with TODM follow the more complete presentation of the model in the sections ahead. TODM was not conceived as a response to the above-mentioned research (most of it was carried out during roughly the same period of time), but rather as an independent attempt to study the design of temporally oriented databases. The critical comparison and assessment of relative advantages and disadvantages of the various approaches are the subjects of Section 7.

3. THE DATA CONSTRUCTS IN TODM

TODM augments the traditional constructs of the relational model (i.e., attributes and relational schemes) with the temporal schemes. These schemes introduce the time dimension and determine the nature of the extension in the model—the cube. In this section we present these elements.

3.1 Attributes

Each attribute A has a domain dom(A) of values it can assume, allowing null values under some conditions. A subset of the attributes, the temporal attributes and the natural key, is designated in that they carry some fixed and inherent partial semantics.

Temporal attributes are attributes that contain some time data. We denote attributes of that type as TRA (*Time-Related Attribute*), and an example might be the expected arrival time of a flight. A subset of the TRAs adheres to a narrower semantic and is denoted by TSA (*Time-Stamp Attribute*). The latter includes attributes like the time of storing a record in the database, the time an equipment malfunction happened, was reported, and so on. It is the application of a strict time-stamping procedure that qualifies TRAs as TSAs. The distinction between the various types of temporal attributes is part of the discussion of TODM constraints in Section 5.

For any TRA, T, dom(T) is an interpreted domain whose underlying domain is the Integers. We have denoted the time unit as Chronon, expressing the time granularity in the given application (i.e., the shortest meaningful unit of time in the specific application). Thus the Chronon of an air traffic control database is probably a fraction of a second, while the Chronon of a payroll application may be an hour, week, or month.

Formally, the domain of the natural key is the set of the relevant objects. The natural key provides the means to associate data captured in the database with real objects in the environment of the database. The natural key constitutes—*at any point in time*—an unambiguous reference to an object (i.e., object identification). Further discussion of the concept is deferred to Section 5, immediately following the discussion of the temporal attributes to which the natural key is intimately related.

3.2 Relation Schemes

At any point in time we identify a relation scheme with a finite set of attributes, since a database schema cannot contain, simultaneously, two different relation schemes with the same set of attributes. Thus if R is a finite set of attributes, say, $R = \{A, B, C\}$, R also denotes a relation scheme. The relation scheme directly models an object type, which corresponds to an entity of interest. This entity of interest could in turn be of type entity or relationship—we maintain no distinction between these two subtypes [24]. At any point in time the database schema is a finite set of relation schemes, $S = \{R_1, R_2, \dots, R_k\}$.

3.3 Temporal Schemes

Temporal schemes define dense times, within which objects exist. These schemes should not be, and are not independent of, the associated relation schemes. Specifically, for each TSA T in relation scheme R the temporal scheme defines a time line: a linear order $<_T$ that awards (a rather straightforward) meaning to temporal expressions like before or after. The complete temporal metrics (e.g., as in [7, 13, 21, 33]) provides the basis for comparing periods and calculating the length of time intervals. Such an interval calculus is not really needed in our current discussion and is therefore omitted. The notion of time in the database is further developed in the discussion of TSAs in Section 5.

Following the common assumption of temporal density, event-data persist and prevail (i.e., they determine the subsequent value of the relevant variables) until the occurrence of a subsequent event upon the same object. This is a simplifying assumption that can be easily relaxed, but is retained for the purpose of clarity and brevity.

Temporal schemes bear some functional resemblance to the relation schemes: Specifically, a relation scheme determines the structure of an object, and a temporal scheme defines the structure of time, within which events concerning these objects happen. Together these schemes define a two-dimensional intension of a relation. As much as the relation scheme becomes an inherent characteristic of the actual tuples of the relation, the temporal scheme analogously becomes an inherent characteristic of the *TAT*—the *Temporally Anchored Tuple*, the temporally oriented relation's extension.

3.4 TATs: Temporally Anchored Tuples

Let R be a relation scheme, T a TSA, $T \in R$, and denote $B = \{R - T\}$. An R-TAT with respect to the TSA T is a function,

$$b_T\colon \{R\} \rightarrow \cup_{A \in B} \operatorname{dom}(A) \cup \operatorname{dom}(T)$$

s.t. if $b_T(A)$ denotes the attribute A of TAT b_T, then for every attribute $A \in B$, $b_T(A) \in \text{dom}(A)$ and the temporal attribute $b_T(T) \in \text{dom}(T)$. The latter constitutes the temporal determinant for the R-TATs, and designates the inherent temporal scheme (i.e., temporal context) of the TATs. Given the above formulation, we use the notation DATA-TYPE$(b_T) = R$ and TIME-TYPE$(b_T) = T$.

A TAT directly corresponds to an event in the database's environment. An event is an instantaneous change in one or more attributes of a modeled object. The database record that captures the state of a given object following such change is realized as a TAT of a relation extension.

The basic premise of TODM is that TATs are never deleted, and every time an object requires modification, a new related TAT is created. As part of their insertion into the database, TATs are automatically time-stamped, and this time stamp is saved in a special TSA denoted RT (for recording time). The intuitive semantics of RT is "when the *database* 'became aware' of the occurrence of the event." Relation schemes may include other TSAs, but their values have to be determined appropriately by the user.

3.5 Data Cubes

An R-cube with respect to a TSA T is a finite set of TATs of DATA-TYPE R and TIME-TYPE T, arranged in sequence by time for corresponding objects. We denote this construct by $E^c(R, T)$ where E^c is a mnemonic for cubic extension, R represents the scheme or intension, and T is the temporal determinant. The canonical extension of a schema $S = \{R_1, R_2, \ldots, R_k\}$ is the collection of corresponding extensions, $E^c(S, RT) = \{E^c(R_1, RT), E^c(R_2, RT), \ldots, E^c(R_k, RT)\}$, the database. This is the primary view of the database, while secondary views, with possibly restricted relation schemes and different temporal schemes, are discussed in Section 4.

In general, relational views of databases are unordered sets of tuples. Our cubic views, however, have an explicit and inherent order that preserves both the temporal context of the data and the identity of the objects captured in the database. In browsing through a cube, in order to follow an object uniquely through time, we require that $E^c(R, T)$ be arranged in a way such that TATs that correspond to a given object are positioned exactly below (above) a later (earlier) TAT of the same object. Moreover, for each object, "earlier" TATs are placed downwhen (i.e., lower) along their TIME-TYPE (i.e., the temporal dimension), in a distance that corresponds to the time difference between the TATs (see Figure 4). The order among the objects themselves is still immaterial, though. This particular arrangement provides the basis for locating all the TATs of any given object, which amounts to the object's history. This inherent order is similar in its function to the use of surrogate keys [16] or temporal invariant keys [5], and the mechanism that maintains this arrangement is likewise transparent to the user and is handled internally.

This spatial arrangement is very similar to the one underlying other historical data models (e.g., [5, 7, 13]). The difference, nevertheless, is that in TODM this arrangement is an inherent property of the data construct, while the others derive it or force it upon a completely unordered collection of tuples. The other attempts use this arrangement as a "mental model," while in TODM it is the primary data structure.

3.6 Views

For a given database schema S, a View V is defined by a set of operations on the schemes in S in an appropriate language [18]. The definition of a View V determines a functional mapping from extensions of the database schema, $E^c(S, T_1)$, to extensions of the View, $E^c(V, T_2)$, where T_1 and T_2 are not necessarily the same TSA.

In general, the extension of a View inherits its internal order from its cubic ancestor. However, if V is a View whose DATA-TYPE contains an orderable temporal attribute T (i.e., a TSA), then $\langle V, T\rangle$ is an ordered View whose extension $E^c(V, T)$ is a finite sequence of V-TATs ordered according to values of T. We refer to that View as the *temporal interpretation* with respect to T of the View V.

In the current state of development of TODM, Views are expected to be specified through a single expression, constructed through a single uninterrupted "database procedure," displayed and then discarded. This restriction is set to avoid the issues of updateability, View manipulations, and some potential semantic problems in relating Views to each other.

4. THE PRIMARY OPERATIONS IN TODM

The second aspect in the definition of a data model is the specification of valid operations on the model's data constructs. These operations facilitate the access to data and the manipulation of the database content. In this paper we only deal with the primary operations, that is, selections and projections, as they are applied to cubes. The perplexing nature of temporal JOIN operations is discussed in [5].

4.1 Object Selection

Let $E^c(R, T)$ be a relation extension and F be a formula concerning the attributes in R. The selection from $E^c(R, T)$ by F is denoted by $E^c(R, T)/F$, corresponding to the View $\langle(\langle R, T\rangle/F), T\rangle$. This selection operator selects object histories, and therefore when operating on a flat relation (i.e., without the temporal depth), it directly corresponds to the classical relational selection operator. In Figure 5 an object selection is applied to the example in Figure 4.

$E^c(R, T)/\text{Object-Id} = o$ is object o's history (i.e., the sequence of states of the object) as determined by the TATs pertaining to the object o, over the time-range covered by $E^c(R, T)$.

Object selections preserve the form of the model's constructs. $E^c(R, T)/F$ is a proper, though new, cubic extension with possibly a restricted set of objects. This means that for every possible F the resulting construct is always a cube. Moreover, the resulting View retains the type and identities of objects in the original relation.

4.2 Temporal Selection

The temporal selection, denoted $/_t$, operates on the temporal dimension of the data and selects the period of time to be covered by the View. Specifically, $/_t$ sets the lower and upper time bounds of the extracted cube, along the associated time

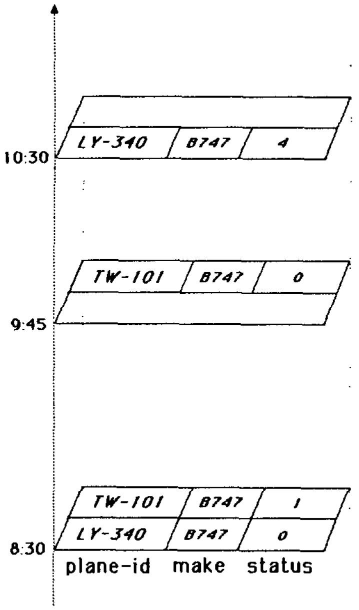

Fig. 5. Object selection: A graphical illustration. Operation: E^c(AIRPLANE, RT)/make = B747. Meaning: The histories of B747s on file.

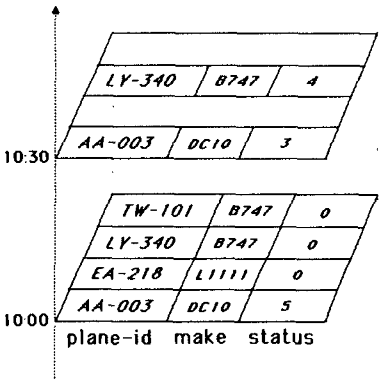

Fig. 6. Temporal selection: A graphical illustration. Operation: E^c(AIRPLANE, RT)$/_t$RT ≥ 10:00. Meaning: The sequence of events after 10:00.

dimension, that is, the temporal scheme. In Figure 6 the temporal selection is applied to the example in Figure 4.

The temporal selection, like the object selection before, preserves the model's basic forms: $E^c(R, T)/_tF$ is a proper, though new, cubic extension over possibly a restricted range of time, or times. This means that for every possible F the resulting construct is always a cube, whether F extracts one continuous time range out of $E^c(R, T)$ (e.g., "during the year of 1982"), or it specifies a set of disjoint time intervals (e.g., "during the months of January in each of the years 1972–1982"). The latter results in a cube which covers the range of time from the beginning of the earliest interval to the end of the latest one. The parts of the cube that correspond to time periods *not* specified by F raise some interesting questions of semantics and interpretation. In TODM these spaces are "filled" in the usual way, along the principle of temporal density, applied to $E^c(R, T)/_tF$.

Another unique, related, problem in any temporal data model is the nature of the lower surface of any disjoint portion of the cube selected by F, and in particular the initial state of the View. Let T, a TSA, be the basis of the temporal scheme of the View, and p_1 the lowest time point, along T, of such a time interval. The temporal selection operation infers the state of the view at p_1 by virtue of the principle of temporal density, that is, for each object in the View, the TAT that prevailed at time p_1 with respect to T is included in the View's extension, with its time stamp modified to equal p_1. The finality property of TSAs (as discussed in Section 5) ensures that there will be no ambiguity in the selection process. Since Views are never subjected to further manipulations, such copying and time-stamp modifications can be tolerated.

If no TAT for a specific object can be found prior (or equal) to p_1, a null object is created. A null object is a TAT with an object identifier (i.e., a key), the attribute T is set to equal p_1, and all other attributes are set to null. This reflects the fact that the corresponding object did not exist at time $T = p_1$.

An inferred time-slice View is a dense cross section of a cube. In particular, $E^c(R, T)/_tT = p$ corresponds to a regular flat relation version of the data as it looked at a time p along the temporal scheme determined by T. For $T = \mathrm{RT}$, the temporal selection re-creates the state of the stored objects as it existed at a specified point in time. In particular, the View $E^c(S, \mathrm{RT})/_t\mathrm{RT} = \mathrm{NOW}$ is the current version of the database, which corresponds to the classical, relational time-varying database with the same schema. This View still carries with it the temporal scheme, and is therefore placed explicitly on the designated time dimension, even though it is a cube with no depth.

4.3 Projection

Let b be a TAT of DATA-TYPE R, and let $X \subseteq R$. The projection of b onto X, denoted by $b[X]$, is simply the restriction of b to Y as a function, where $Y = X \cup \text{TIME-TYPE}(b)$. The projection operation retains the original TIME-TYPE of b, and $b[X]$ is obviously of DATA-TYPE X. Let R be a relation scheme and $X \subseteq R$, $R[X]$ is a View called the projection of R onto X. The extension of the projection view is $E^c(R[X], T) = \{b_T[X] \mid b_T \in E^c(R, T)\}$. In Figure 7 the projection operation is applied to the example in Figure 4.

Note that cubic projections, like the selection operations above, preserve the cubic form and maintain the property that operation on a cubic extension results in another cubic extension. The projection is defined such that the TSA T that orders $E^c(R, T)$ is automatically included in each projection.

In order to maintain compatability with fundamental conventions of the relational model, if two consecutive projected TATs of the same object turned out to be identical (except for T), the latter of the two is eliminated. Similarly, identical *sequences* of TATs (including time stamps) in $E^c(R[X], T)$ are consolidated and no complete duplicates are included. Figure 8 demonstrates these conventions, using the result of the temporal-selection operation example (Figure 6) as the source data (i.e., a slightly modified version of the running example in Figure 4). Specifically, in Figure 8 the middle object corresponds to two different objects in Figure 4.

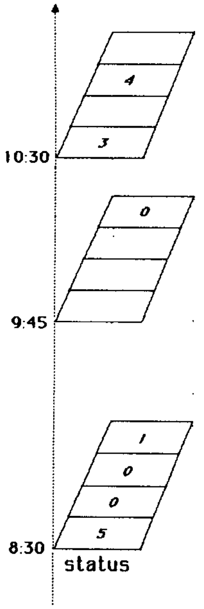

Fig. 7. Projection: A graphical illustration. Operation: E^c(AIRPLANE[status], RT). Meaning: All the sequences of "status" during the period of the view.

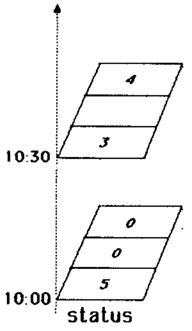

Fig. 8. Projection: Another graphical illustration. Operation: E^c(AIRPLANE[status], RT)/$_t$RT ≥ 10:00. Meaning: All the unique sequences of "status" after 10:00.

5. TODM's CONSTRAINTS: TIME, TEMPORAL ATTRIBUTES, AND KEYS

TODM reflects the uniqueness assumption of the relational model through two conceptual constraints. The first refines the type of time along which Views could be temporally interpreted, and the second extends the notion of object identifier.

5.1 TSAs and TRAs: Examples

There are fundamental differences between TSAs and the semantically looser TRAs, as they are recognized in TODM. The essence is that a TSA has to represent "the time something *happened*" and must be a time stamp in some sense, though not necessarily *RT*, the time stamp of the database. Many events outside the computer may have their own time stamps, and when those times are retained in the database, they can be designated as a TSA.

Consider, for example, a relation scheme pertaining to the scheduling of airplane maintenance:

MAINT-SCH
= {PLANE-ID, M-CREW, M-TYPE, START, END, STATUS, RT}

where PLANE-ID is the serial number that identifies an airplane, M-CREW identifies the group of people who are scheduled to perform an M-TYPE maintenance procedure on the airplane, START and END delineate the duration of the treatment, STATUS can either be "tentative," "confirmed," "ongoing," or "completed," and RT is the time stamp when different modifications of the maintenance schedule were recorded in the database.

Although START and END are temporal attributes, they cannot serve as the ordering attribute in a temporal interpretation. PLANE-ID, the natural key, and START could remain the same as STATUS goes from tentative to confirmed, resulting in simultaneous but conflicting TATs, and therefore {PLANE-ID, START} *cannot* be a key.

There could be other attributes besides the RT that exhibit the "time-stamping" behavior, later referred to as the Finality Property. Consider a similar relation about maintenance of aircraft:

MAINT-EVENT
= {PLANE-ID, PART, CONDITION, ME-TIME, SECTOR, RT}

where PLANE-ID is again the serial number of the aircraft, PART and CONDITION are obvious, ME-TIME is when the maintenance event occurred, SECTOR is the regional FAA center that handled the maintenance event, and RT is when it was reported to, and recorded in, the central database.

Each SECTOR reports to the central authority at different frequencies, and usually each report contains many events. It is easy to see that there may be two TATs with respect to the same aircraft serial number (i.e., same object) where the temporal orders along ME-TIME and RT are incompatible:

⟨123456, wing, cracked, 2/1/82/12:45, Boston, 2/5/82/3:00⟩
⟨123456, door, unhinged, 1/31/82/14:22, Houston, 2/7/82/2:00⟩.

ME-TIME reflects a physical constraint in the sense that no more than one (correct) maintenance report can be filed at any given point in time, as it is bound by the physical presence of the airplane on location. Thus, the temporal interpretation E^c(MAINT-EVENT, ME-TIME) is perfectly valid, even though it presents these events in different order than E^c(MAINT-EVENT, RT).

5.2 The Finality of TSAs

The preceding examples were meant to develop better intuition as to how TSAs and TRAs differ. To put it formally, when a TSA is considered the temporal determinant of a View, we require the following property [5]:

Finality Property. Let T be a TSA in a scheme $R = \{\text{OBJECT-ID}, \ldots, T\}$ and let $b_T = \langle \text{object-id}, \ldots, t \rangle \in E^c(R, T)$. Then the TAT b_T is the *final and only* set of values for the object ⟨object-id⟩ at time $T = t$.

This property captures the time-stamping nature of TSAs, which allows us to view the world through a TSA time.

In general, a relation scheme has at least one TSA, the internally controlled RT, but it no doubt may have more than one TSA. In practice, for TSAs other than RT it is possible that a TAT may be incorrectly entered and may have to be changed. This means that another TAT with the same TSA value (but not the same RT value) is recorded in the database, which violates the Finality Property for an otherwise perfect TSA. We assume that no such errors are made, though any actual implementation would have to deal with this difficult subject.

5.3 Functional Dependencies and Keys

The underlying notion of functional dependency is expressed in the same way as it is expressed in the nontemporal model. Specifically, let R be a scheme and X, $Y \subseteq R$, $X \rightarrow Y$ (read "X functionally determines Y") if for any extension, $E^c(R, T)$ and any two TATs b_T, $c_T \in E^c(R, T)$, if $b_T[X] = c_T[X]$ then $b_T[Y] = c_T[Y]$.

We require, however, that a unique key hold for any temporally selected View with respect to a TSA, and refer to any such key as a natural key with respect to that TSA. For instance, TATs are uniquely identified by the combination of their natural object identification and their RT value. Extending this property to other TSAs is technically more difficult, but nevertheless must be universally guaranteed [5]. One should note that the designation of attributes as TSAs is an application-dependent activity and that different TSAs may have different natural keys associated with them.

6. TOSQL: TODM-BASED SPECIFICATION OPERATIONS

The explicit treatment of time calls for the development of further specialized query language constructs. TODM is an attempt to systematically define a semantic basis for the implementation of a temporally oriented DBMS, and therefore the query language proposed in this section has been explicitly guided by it. In particular, TOSQL follows the fundamental design approach in TODM, that is, a unified access to both current data and their previous versions. The operations introduced in this section offer an immediate access to historical data, and in a form compatible with common relational query syntax. Consequently, only minimal changes are required in the specification of queries about current data.

6.1 Basic Retrieval Operations

The basic specification operation in TOSQL reflects the notions of TODM, and its three generic parts directly correspond to the three dimensions of $E^c(R, T)$, namely, the specification of attribute, object, and time context. The syntactic form we have selected is based on a subset of SQL [6], with necessary extensions for the specification of the query's time aspects. The default options are defined such that a query that omits the temporal portion retains the standard meaning of the corresponding SQL SELECT operation.

This extended query format is laid out in Figure 9. The syntax is presented in a close variant of Backus–Naur Form, where underlining designates default

```
⟨query⟩ ::= ⟨b-query⟩ ⟨obj-spec⟩ ⟨time-spec⟩ ⟨time-qualif⟩
⟨b-query⟩ ::= SELECT ⟨att-spec⟩ FROM ⟨cube-name⟩
⟨att-spec⟩ ::= * | ⟨att1⟩, . . . , ⟨attn⟩
⟨cube-name⟩ ::= a name of a properly defined database cube
⟨obj-spec⟩ ::= ALL-OBJECTS
              | WHERE ⟨selection-expression⟩ ⟨prevalence-mode⟩
⟨selection-expression⟩ relates attribute-references and literals through comparison operators
                       (e.g., =, >, <), or Boolean operators (AND, OR, and NOT). Parentheses
                       enforce desired order of evaluation.
⟨prevalence-mode⟩ ::= EVERYWHEN | SOMEWHEN
⟨time-spec⟩ ::= ⟨time-period⟩ ⟨time-dimension⟩
⟨time-period⟩ ::= AT ⟨time-point⟩
                 | WHILE ⟨selection-expression⟩ ⟨temp-boundaries⟩
                 | DURING (⟨t⟩ − ⟨t⟩)
                 | BEFORE ⟨t⟩
                 | AFTER ⟨t⟩
⟨time-dimension⟩ ::= ALONG RT | ALONG ⟨tsa⟩
⟨time-point⟩ ::= PRESENT | ⟨t⟩
⟨temp-boundaries⟩ ::= DURING (−∞ − +∞) | DURING (⟨t⟩ − ⟨t⟩)
⟨t⟩ ::= time value, in Chronons
⟨time-qualif⟩ ::= AS-OF ⟨time-point⟩ ⟨time-dimension⟩
```

Fig. 9. TOSQL: A TODM-based specification syntax.

values for omitted components. Time is mainly encapsulated in the ⟨time-spec⟩ and ⟨time-qualif⟩ component of the query, but has also affected the ⟨obj-spec⟩ component. The temporal context of the query, ⟨time-spec⟩, defines the range of both the object selection and the resulting cube. It allows one to construct cubes based on either an explicit time specification (e.g., DURING, AFTER, BEFORE) or implicit through the WHILE option.

As far as object selection is concerned, the formulation of the query has to indicate whether the specified condition should prevail *throughout* the time interval (i.e., "everywhen") or only at some point(s) (i.e., "somewhen"). There is an explicit differentiation between the semantics of WHERE and WHILE, even though they include a similar type of expression. For instance, the applica-ion of *WHERE* M-TYPE = "preventive" to the airplanes' MAINT-SCH cube (cf. Section 5.1) selects airplanes that underwent preventive maintenance, while applying *WHILE* M-TYPE = "preventive" to the same base cube singles out periods of time in which preventive maintenance was performed on the objects mentioned in the query. WHERE is embedded in the temporal context eventually designated by WHILE.

6.2 TOSQL: Examples and Interpretation

All the query examples in this section refer to the relation *maint-event*, concerning aircraft maintenance:

{plane-id, part, cond, status, me-time, m-crew, sector, RT}

where plane-id is the serial number of the aircraft, part and cond are obvious, status indicates the state of the repair activity (e.g., "scheduled," "ongoing," or "complete"), me-time is when the maintenance event occurred, m-crew is the

```
Source:         SELECT plane-id, m-crew
                FROM maint-event
                WHERE status = 'ongoing'

TODM/TOSQL
Interpretation:     SELECT plane-id, m-crew
                    FROM maint-event
                    WHERE status = 'ongoing' EVERYWHEN
                    AT PRESENT ALONG RT
                    AS-OF PRESENT ALONG RT
```

Fig. 10. Example—"standard" query.

```
Source:         SELECT plane-id, part, cond
                FROM maint-event
                WHERE sector = 'boston'
                AT 8/21/84 ALONG me-time

TODM/TOSQL
Interpretation:     SELECT plane-id, part, cond
                    FROM maint-event
                    WHERE sector = 'boston' EVERYWHEN
                    AT 8/21/84 ALONG me-time
                    AS-OF PRESENT ALONG RT
```

Fig. 11. Example—"time slice" query.

maintenance crew assigned to the job, sector is the regional supervision center which handled the maintenance event, and RT is when it was reported to and recorded in the central database.

Example 1. *A "standard" query.* An innocent query to produce a report concerning *current* assignment of the maintenance crews on duty appears at the upper portion of Figure 10. This query in TOSQL is augmented by the predetermined set of defaults, as interpreted in the lower part of the same figure.

The two time-related components AT and AS-OF are rather redundant in this query. Their meaning here is to specify that the query relates to *current* assignments, and uses the most up-to-date data about it. Their true power is demonstrated in the third example below.

Example 2. *A "time-slice" query.* This query invokes the reconstruction of the types of problems experienced in a given day, the Chronon of maint-event. The query, which accesses the historical "trail" of the maint-event relation, appears in the upper part of Figure 11. The explicit TOSQL interpretation is given at the bottom of Figure 11. The defaults result in a View based on the latest information available. This information might not have been available on the 8/21/84 due to delay in recording and entry, but it is known today, a few months later.

Example 3. *A "cubic" query.* Imagine that following a plane crash the investigation focuses on the sequence of events that preceded the accident, and the query in Figure 12 is then issued.

```
Source:   SELECT part, cond, sector
          FROM maint-event
          WHERE plane-id = 'AA4711'
          BEFORE ⟨crash-date⟩ ALONG me-time
          AS-OF PRESENT ALONG RT
```

Fig. 12. Example—standard "cubic" query.

```
Source:   SELECT part, cond, sector
          FROM maint-event
          WHERE plane-id = 'AA4711'
          BEFORE ⟨crash-date⟩ ALONG me-time
          AS-OF ⟨crash-date⟩ ALONG RT
```

Fig. 13. Example—cubic query with AS-OF qualification.

```
⟨att-spec⟩ ::= * | ⟨att_1⟩, . . . , ⟨att_n⟩ | ⟨function⟩
⟨function⟩ ::= COUNT (⟨att⟩) | COUNT-UNIQUE (⟨att⟩)
               | AVERAGE (⟨att⟩) | AVERAGE-UNIQUE (⟨att⟩)
               | SUM (⟨att⟩) | SUM-UNIQUE (⟨att⟩)
               | MAX (⟨att⟩)
               | MIN (⟨att⟩)
```

Fig. 14. TOSQL syntax modifications for reduction operations.

However, the absence of preventive action on the part of the otherwise flawless chief engineer to "this sequence of signals indicating a malfunction," as retrieved by the preceding query, may be explained by the response to the slightly different query in Figure 13.

Reviewing the two sequences (e.g., by means of TODMS's Graphic Interface [4]) might reveal what the chief engineer could have known, but actually did not know, at the time of the accident. This is indeed the mechanism that underlies the aforementioned retrospective analyses in a business environment.

6.3 Reductive Retrieval Operations

Reductive operations, that is, aggregate functions like summation and averaging, are included in relational query syntax out of practical necessity. The research of temporally oriented databases so far has failed to introduce them in a syntactically consistent fashion (e.g., [7]).

The syntax in Figure 9 accommodates these operators in a very natural way—one simply augments the ⟨att-spec⟩ of the ⟨b-query⟩ as indicated in Figure 14. Each of these operations results in a scalar that summarizes the data included in the construct defined by the remaining clauses of the query. The query syntax presented here is compatible with SQL, but is nevertheless restricted, that is, only single aggregations may be applied in a SELECT query. Moreover, the cubic data construct could be "collapsed" in two ways, namely, along both its time dimension and its object dimension. The full scope of aggregation, as well as the removal of the above-mentioned syntactic convention, necessitates the introduction of a temporal adaptation of SQL's GROUP component, which is beyond the scope of the current discussion.

7. DISCUSSION: A PRELIMINARY EVALUATION

Given the absence of established evaluation guidelines for temporal data models (e.g., completeness criteria), we resort to a less objective method of assessment and briefly contrast TODM and TOSQL with contemporary attempts to address related problems. The salient properties of these alternative models are briefly presented in this section, and commonalities and differences viz. TODM are highlighted.

In general, the various temporally sensitive conceptual models seem to suggest five major guidelines to which TODM does respond:

(1) Time is a major, if elusive, attribute of real phenomena, and thus the images of these phenomena, as captured in an information system, would be more faithful if they too included the element of time. TODM's data construct is inherently temporal.

(2) The notion of *event* as a fundamental modeling concept enjoys a broad consensus. It is quite imperative that a temporally oriented information system should incorporate this notion in its data model. TODM's TAT is a direct modeling of an event.

(3) It seems that an ideal information system should be capable of relaxing the assumption of temporal correspondence between the states of the database and the modeled environment. This raises the need to deal with more than one dimension of time within the same system. Specifically, the information system should be cognizant of at least two dimensions: its own recording history and the extrinsic time of the facts it contains. Both TODM's Views and TOSQL syntax begin to address this perplexing issue.

(4) There are apparent differences among conceptual data models with regard to the place of time in the hierarchy of constructs, namely, "above" or "underneath" other objects: that is, whether every fact is embedded in a time context ("Time precedes the objects") or whether data are organized as objects' history. TODM's time is *orthogonal* to the other constructs, and thus simultaneously supports both views through the object selection and the time selection operations.

(5) There are also apparent differences with regards to the criteria or mechanism for determining an entity's existence. TODM follows Bubenko's approach [9] and does not designate a special universal attribute but rather assumes that existence is interpreted from events. For instance, the firing of an employee, which would typically set off his or her existence condition, is not a general rule, since the employee may still exist in terms of retirement benefits, or be eligible for retroactive salary adjustments.

The remainder of this section includes specific comments concerning these conceptual issues, arranged as a comparative review of related research.

The infological data model [27, 28] associates the concepts of an object, a property (or relation), and time to form an "atomic" elementary fact. Such constructs are assumed to reflect the "natural" structure of human perception and knowledge and are recorded as elementary messages, which include the *references* to an object, a property (attribute or relation), and a time point. An elementary message informs that an "elementary situation" prevails, either at a time point or during a time period. The infological concept of elementary message directly corresponds to the TAT.

Both the infological model and TODM assume that messages are always accumulated in the database, even though they could be of type insert, delete, or update. With this conceptual memory mechanism, the system can retain information about properties or relations that are no longer in effect.

Although time is identified as one of the three primary components of the original infological approach, its role in it was limited to the mere built-in temporal indexing of recorded facts. This notion of "infological" time was nevertheless refined in [9]. For instance, a further distinction was made between *extrinsic* and *intrinsic* times. Specifically, extrinsic time reflects the fact that every statement is passively embedded in a temporal context—whether or not this context is formally preserved (e.g., the statement "*Y*, moves to, New York" made in October 1982 has an extrinsic time index "October 1982" associated with it), while intrinsic time is part of the *content* of a statement (e.g., the statement, made in October 1982 that "*Y*, moves to, New York, 9/82," includes the intrinsic time "9/82"). This notion is handled naturally in TODM. Specifically, *RT* is the ultimate extrinsic time, generally a TSA, and TRAs are intrinsic time attributes, as they are part of the *content* of a statement.

A much broader perspective of information modeling and time as a modeling construct was developed in [10]. In it a "time model" is specified *before* any other concept or data modeling construct is defined, reflecting the primary nature of the temporal context. This time model focuses on the hierarchical aspects of the structure of time and provides a formal calendar system, with strict, unambiguous hierarchy of time periods. TODM clearly subscribes to this broader perspective of time as a modeling construct. Nevertheless, the hierarchical time model in [10] is much more elaborate than the one included in TODM.

The temporal extension of the Entity-Relationship model (TERM) [26] concentrates on a set of modeling primitives based on the constructs of the Entity-Relationship model [11]. The history structure, a data construct designed to capture special properties of time, augments the basic E-R constructs to create extended constructs such as attribute history or role history. An entity history is represented by the histories of the attributes that make up this type of entity (a composite history). Time in TERM is thus captured "underneath" or "within" the regular E-R modeling constructs (i.e., under each basic fact, attribute, or relationship, one can find its history, the history "owned" by it). This is quite different from the way it is handled in most of the models mentioned in this paper, where each time point designates an entire database state (i.e., the time point "owns" the corresponding descriptions of attributes and relationships that prevailed at that time). TODM allows us to view data in both ways, that is, through temporal and object selection.

In TERM, an entity history automatically includes a latent attribute of existence-description, indicating the periods of time during which the corresponding entity "exists." In a constantly growing database, setting off the existence attribute of an entity simulates the effect of the ordinary deletion operation. As mentioned above, existence in TODM is interpreted through application-dependent rules.

The basic operation in TERM is the addition of a state to a history structure. Such a database "registration" transaction can either initiate or complete a history. However, a designated privileged authority has also the power to issue a correction transaction to erase or alter facts already in the information system. This introduces the special notion of *recording history*, namely, tracking the development of the content of the database over time and preserving the sequence of changes in the database. In TODM this issue is being addressed rather naturally through the concept of RT, and in a fully uniform way, without any special mechanism for handling it.

TERM also deals with the representation of an infinite history by a finite set of states. For that purpose it recognizes a (finite) set of *characteristic states* and a *derivation function* by which states that are not enunciated explicitly are inferred (e.g., linear interpolation, exponential smoothing, splines, or step functions). This function explicates the temporal density discussed in Section 2, which is inherent in TODM.

The Historical Database (HDB) model [13, 14] aims at providing a rigorous semantic basis for temporal databases. This is achieved through a linguistic view of databases and by exploiting the analogy or parallelism between HDB queries and high-order intentional logic operations. The HDB and TODM models differ in this respect. The semantic basis of TODM is more figurative and intuitive, supported mainly by the appeal of the cubic metaphor for temporal representations.

Time is captured in the HDB through a set of time-stamped relation instances (i.e., a database state, snapshot, or version). Each such relation instance is "completed" so that every object that ever existed is represented in it, which allows the alignment of parallel slices (i.e., states) to form a regular cube similar to the one depicted in Figure 4, but with unchanged tuples repeated in each explicitly defined database instance. The "historical relation," which is the basic data construct in the HDB model, is defined as the flat union of all the completed instances. A historical relational database is therefore a collection of historical relations over the same set of states. "Historical relations" are informationally equivalent to TODM's cubes, but are clearly highly redundant. In general the HDB model depends on rather rigid database states and concentrates on a single dimension of time.

Unlike TODM, HDB uses some designated internal attributes as the basis for incorporating explicit temporal semantics into the model, that is, relation schemes generically include two implicit attributes: (1) a time-stamp attribute that marks the set of all tuples in a relation instance (an RT of sorts), and (2) a Boolean indicator for determining the existence of an object.

HDB augments the set of explicitly defined database states with a function that explicates the mechanism by which data values *between* states are derived. In TODM the equivalent function is assumed to be a part of the basic data construct, the cube. In both cases the underlying premise is a combination of the principles of temporal density and of temporal completeness. There is, though, a definitional confusion in HDB with regards to these time concepts: The principles of temporal density and completeness are defined in HDB only in the context of "filling the gaps" between database states. However, these principles underlie the very notion of these states and the completion process that creates them. In TODM time is included as a *primary* database concept, and is defined before any notion of extension can be elaborated.

With respect to specialized time-related operations, HDB includes a "historical database select," a restricted version of TODM's $/_t$, that selects, for temporally anchored queries, the single appropriate database state upon which the query should operate. $/_t$ in TODM selects arbitrarily deep cubes.

The approach adopted in [5] can be viewed as a "lazy" version of HDB, in the sense that the basic data constructs are regular relations and the completion process is dynamic and carried out only upon demand. Time is captured in the Temporal Data Model (TDM) by any number of time-related attributes, which are used similarly in TODM. Some temporally oriented higher level operations are added in TDM to the standard repertoire of relational operations. In particular, an "object history" operator is defined, as well as a function that derives from it the view of an object at any point in time. The latter is a temporal density inference mechanism that chooses the latest tuple of the object before the time designated in the query. Needless to say, TODM and TDM have been influenced by each other. In particular, the "object history" resembles TODM's /, the object selection, and the "inferred time view" function is similar to the $/_t$ when the latter derives the view of an object at a given point in time.

The novel element in [5] is the definition of *Natural Time Joins,* where time is fully considered. This definition reveals some interesting characteristics (e.g., that joining R to S, and joining S to R, which represent two aspects of the same object, are not symmetric operations). Such operations yield different results, as joining R and S means matching each tuple in R with the temporally corresponding inferred View of the same object in S. The basic temporal join is not commutative or symmetric, as the operation is sensitive to the relative temporal ordering of tuples in the joined relations. Even though the difference between the JOINs might have semantic justification in some cases, it probably will not exist in the corresponding JOIN in the cubic data model. This issue is designated for future research, and thus our current thinking about it is included in Section 8.

Another central issue in TDM is the long-term identification of the external, real-world entities and the mapping of these entities to their database representations. Over long periods of time one cannot assume that natural keys will remain unchanged, since employees get married, files are renamed, companies merge, and postal ZIP codes are extended. For that purpose the notion of TIK, for Temporally Invariant Key, is developed and examined. The TIK is a transparent, arbitrary, and internal identification of entities, a temporally extended surrogate key [16]. In TODM this issue is resolved differently: The position of a TAT on the "object coordinate" of the cube is a de facto TIK. By assuming this inherent arrangement of TATs, an object's identity is maintained over time. This approach is weaker in a formal sense, but is nonetheless sufficient from the vantage point of implementation (e.g., the actual positioning of TATs could be achieved by a pointer-chain mechanism).

Legol 2.0 [21] offers a temporal join through the use of the WHILE operator to identify overlapping periods of time. The premise is that "Comparison [of values] implies a time intersect. It is of course only sensible to compare [say] salaries in force at the same time" [21, p. 78]. The complementary WHILE-NOT operator provides the time-difference operator, and DURING augments the repertory of temporal key words with a time-oriented set-membership test.

The semantic distinction between WHERE and WHILE with regard to their time aspects has been blurred in [21]. Even though they may include a similar type of expression, the application of *WHERE* conditions is temporally nested within the time range designated by the *WHILE* condition. WHERE may result in limiting the temporal range of view, but only in an indirect fashion (e.g., through the existence period of the selected objects), while WHILE treats time directly, and delineates the range of interest. The distinction is obvious within TODM, as the two refer to different dimensions of the cube. Legol 2.0 reference to time is always "everywhen," while TOSQL provides more freedom in this respect.

Unlike TODM, Legol 2.0 captures time by two mandatory TRAs ("start" and "end") that appear in each tuple. Nevertheless, these time attributes do not carry a uniform meaning throughout the database—in some cases they define calendars and time periods, in others delineate the existence interval, and in yet other cases delineate the period of relevance. Moreover, the way in which the original time attributes are inherited by derived views is unspecified and not always consistent. As in TODM, the results of any query are displayed and then discarded; they are never retained.

The Time Relational Model (TRM) [7] offers an elegant solution to some of the complexities involved in incorporating time aspects in a relational model. Time-Relation (TR), the data construct in this model, is the unordered collection of an exhaustive set of object histories. TRM captures time in five mandatory temporal attributes, associating each tuple with various periods of time. These fixed time attributes are (1) the recording time, the subject of the "as-of" statement; (2) the time when the statement included in the tuple starts to take effect (this time stamp and the recording time are recorded simultaneously in the database, but there are no mutual constraints on their values: the assertion could be either retroactive or proactive); (3) the time when the tuple is marked as "physically" deleted; (4) the time the statement in the tuple ceases to be effective, that is, it is "no longer valid" and therefore logically deleted; and (5) the time when the tuple is marked "void" for reasons of input error of a technical nature. The last indicates that the statement contained in the tuple should be ignored in any database query as of a later point in time.

TRM allows users to view the database content *as of* a predetermined date. The primary access mechanism to data is a procedure that extracts a regular flat relation from a TR (i.e., a time slice through the cube in terms of Section 2). This extracted time slice includes the tuples that are "*in effect*" at a designated point in time, say p_1, *as of* a possibly different point in time p_2. This mandatory procedure of access to data is equivalent to the TODM's View

$(E^c(S, \mathrm{RT})/_t\mathrm{RT} \leq p_2)/(T_{es} \leq p_1$ and $T_{ee} \geq p_1)$, that actually factors out the time element. T_{es} and T_{ee} are the time attributes that define the period during which the statement is effective. TODM's View reflects the fact that T_{es} and T_{ee} are only TRAs, as in their natural meaning they most likely cannot qualify as TSAs. TRM does not distinguish among the different categories of temporal expressions. Of the five TRAs mandatory in each tuple in the TRM, only the recording time and the "physical" deletion time are TSAs and are equivalent to TODM's RT.

Any data manipulation (e.g., SELECT, PROJECT, or JOIN) in TRM is preceded by an application of the AS-OF extraction procedure mentioned above. As a result, the operations simply retain their standard relational definition. However, this also means that the user of TRM does not have an immediate access to time, but rather to derived, flat, and static relations. This limitation is apparent in the way the set of Time-Aggregates is introduced. These functions allow the user to access information *along* the recording time and find, for example, the number of changes or periodical averages. Nevertheless, these Time-Aggregates are inconsistent with the other operations in TRM, as they are defined to ignore or circumvent parts of the as-of extraction process that precedes them [7, p. 134]. In TODM these operations are consistent with the overall syntax, as time is explicitly addressed (cf. Section 6.3).

TQuel, a temporal query language described in [33], is a superset of Quel, an established query language based on relational tuple calculus. As such it provides language constructs for manipulation (actually only retrieval) of facts that have been time-stamped with their validity interval (i.e., the period of time during which they were "true"). The *retrieve* operation of Quel is modified to include two additional components that deal with the temporal aspects of the query, namely, (1) a definition of the "mechanism" by which the implicit time attributes of the derived relation are to be constructed from the corresponding attributes of the source relations, and (2) a temporal conditional: a temporal predicate concerning the implicit time attributes of the associated relation. The latter is the basis for selecting source tuples, and the interaction between this "WHEN" aspect and the regular "WHERE" in a query is through a logical AND relationship.

The underlying data model of TQuel is basically the relational, with each tuple augmented with 1, 2, or 4 "transparent" time attributes, augmenting the tuples with information concerning their time points or intervals during which they prevailed, or were probably prevailing (the latter allows some initial treatment of fuzzy reference to temporal relationships). TQuel enjoys the rigorous basis of Quel and is constructed to conform to the tuple calculus, which provides the basis for the formal interpretation of its semantics. Nonetheless, TQuel is not based on an explicit temporal data model, and the focus is mainly on the compiler aspects of a temporally oriented query language. As a result the temporal properties of data are only discussed in the context of valid expressions in the sequencing of events and periods of time. Unlike TODM and TOSQL, TQuel relies on fixed time attributes in the assumed model and allows two methods of time recording (point and interval values). As a result, TQuel rules become unnecessarily complex, and they do not maintain the property of closure. The comments made with regard to the insensitivity of Legol 2.0 to the differences between time-related WHERE expressions and those that are expressed through WHILE are applicable to TQuel as well.

8. CONCLUSION AND FURTHER RESEARCH

TODM *preserves* the basic, implicit nature of time: Even though we may not be fully aware of it, we are constantly moving in time. Accordingly, events under TODM are captured in, and accessed through, the data cube, where they are necessarily anchored in a temporal context, *apart* from other semantics and contexts. The essence of this paper is an examination of a way to translate the pervasive three-dimensional representation of temporal relations from a mental image into a primary and tangible data structure. TODM and TOSQL are therefore meant to allow their user to retain that comfortable spatial metaphor for time while browsing through the database.

This enhanced capacity comes with a price: it complicates the nature of the (validity) constraints in the model. For instance, the convenient "closed-world assumption" [30] is eliminated as, say, a person can be correctly recorded as married and single in the same instance of the database, though in reference to different times.

TODM was designed to accommodate salient temporal characteristics of major managerial tasks, temporal concerns in management and management science, as well as time-related characteristics of individuals. The key temporal requirements or guidelines for information systems seem to be [4]:

—Time, as captured in historically oriented databases, is not an isolated concern but rather an inseparable attribute of operational data. TODM therefore consolidates the operational (i.e., short-term) and historical concerns of the information system into a single unit of data storage.

—The impact of single events should be preserved. The atomicity of events provides the level of detail needed for retrospective restructuring of information. TODM's TAT is the direct result of this consideration.

—The need to deal with more than one "time line" in the system has been recurrently emphasized. In particular, human awareness of the occurrence of events is not necessarily linear, uniform, or unidimensional—there is a fundamental difference between irreversible physiological (database recording) time and psychological time; the latter is a result of the cognitive reconstruction of past events which is not confined to their original, physiological order [19]. TODM's definition of Views, with the explicit assignment of temporal schemes, is an attempt to deal with this issue.

In TODM we have tried to examine feasible responses to this set of "ideal" requirements. Needless to say, the research reported in this paper can be deepened and expanded. In the remaining part of this section we elaborate on further TODM-related research issues.

Handling Future Data. An intriguing question relates to the generalized storage of data that pertain to the future as it is perceived at a given point in time. Each point in time has its own future, and this temporally anchored view is needed for understanding the decisions made in that period.

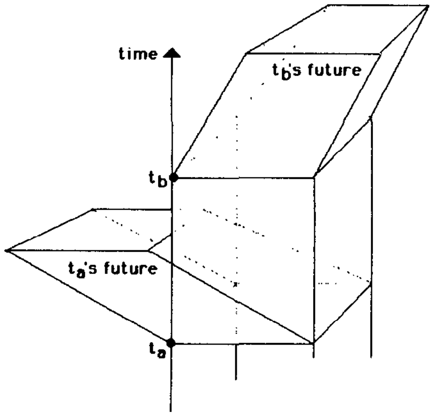

Fig. 15. A cube with multiple futures—preliminary model.

Our current work has implicitly focused on the historical perspective, that is, cubes were typically constructed such that they are viewed "after the fact," namely, the AS-OF time argument in the query that created them is later than other temporal attributes of their data. In the general case, though, the cubes' time range could *include* their AS-OF time. TODM's data construct can basically accommodate time-anchored views of the future. Nevertheless, handling the future entails immediate questions with regard to both abstract data modeling and implementation issues.

For instance, personnel assignments are usually known in advance (even though they may not always materialize as expected), and a cube AS-OF Recording Time June 1984 could refer to all the assignments during the entire year of 1984. Those in July and later are clearly different from those before—they are only planned and are tentative. As a result, the same cube of assignments during 1984, AS-OF Recording Time *November* 1984, will possibly differ from the former. An extension of the cubic model that addresses this issue is visualized in Figure 15.

Although this recommendation is quite obvious, its implications and the issue of multiple futures and "branching time" are not. This mental model has to be formalized, and language constructs to properly deal with it should be devised and examined.

The Nature of Captured Time. Ideally, the periodicity of time should be integrated into the mechanisms of data management. In many operational aspects, time is periodic and composed of intervals, and in many instances it is even cyclic (e.g., "by the end of each spring term," "on every Friday throughout the next month"). This issue is further complicated by the subjective and social (or microsocial) nature of some periodical concepts. For instance, the week is not anchored in any natural phenomenon, and the same applies to five-year plans, quarterly business results, and others. The specific research issue relates both to the way these "times" should be captured and to how they should be presented.

Another issue relates to the way time is specified. Specifically, fuzzy time expressions are used very commonly, both in memory (where we may not care for exact time reference) and certainly in the recall and search for partially

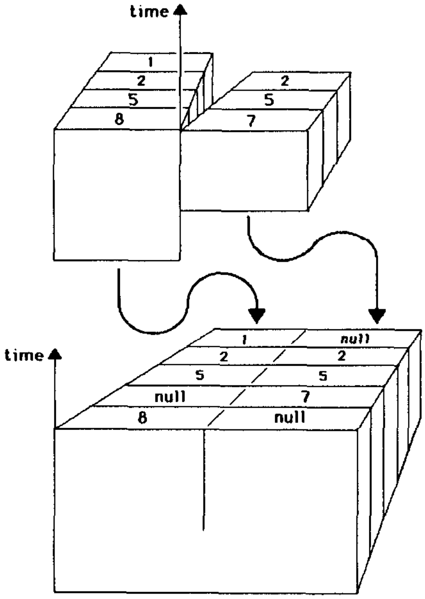

Fig. 16. A spatial metaphor for a natural time join.

specified time data (where we may not know the exact time reference). Implementing an operator for the selection of chronologically similar sequences of events is an immediate example of such an operator. This last subject received only precursory attention, mostly in AI contexts (e.g., [22]), and its implication for database and user interface design is still an open issue.

Advanced Manipulations. One of the motivating issues for the study of time and databases has been the increasing popularity of retrospective analyses (i.e., "what-if" scenarios). TODM and TOSQL in their current versions support this end in an indirect manner, as they only provide the necessary primitives. Language constructs should be devised such that they will deal directly with "what-if" manipulations within the cubic model, preferably in a style compatible with the extension of SQL presented in Section 7. This would most likely entail the study and development of the mechanism for the manipulation of retrieved Views. In that way a formula could be applied to augment a cubic View with a new derived attribute and values.

Temporal Joins. The intuitive equivalent of these operations in the model developed here is the placement of two cubic extensions next to each other, and the creation of a new cubic extension with a combined set of attributes, matched objects, and matched time values. The tentative perception of this operation is given in Figure 16.

Implementation and Performance Evaluation. A basic premise of TODM is that no information is ever lost due to updates. The physical database retains all the changes, and once they are recorded they are never modified. This concept introduces some interesting features in the areas of data integrity, system availability and recovery, complex alerting, and database restructuring.

However, the actual construction of a DBMS based on TODM opens a whole set of efficiency-related design questions. In the abstract model each TAT is a complete (cumulative) tuple, but in actual storage a balance could be struck between the storage of single transactions and duplication of the cumulative status of the entity. A dynamic decision rule with regard to redundancy could be formulated, and marked redundant summaries could be created to minimize seek times. For instance, a summary record could be stored with the record reflecting a change following long periods without any such event (i.e., the previous TAT is relatively far behind). In an ongoing research effort [32], there is an attempt to implement a DBMS where the external data model is based on TODM. The cubic model is then translated into internal structures based on the concept of temporal differentiation of attributes.

The nature of time in computer-based information systems and its handling in such systems have recently caught the attention of the database community, both researchers and practitioners. The recent intensifying attention to the role of time in information systems is a manifestation of the never ending quest, to use Codd's words [16], to capture more meaning and enhance the semantic capacity of data models and systems for the management of data. Time touches almost every aspect of information system design, implementation, and operation; and there is still a lot to be understood.

ACKNOWLEDGMENTS

I wish to express my gratitude to Norman L. Badler, Aaron Beller, O. Peter Buneman, Eric K. Clemons, James L. Emery, Howard L. Morgan, and Bonnie L. Webber for their helpful comments on earlier incarnations of this paper, and to Linda-Jo Calloway and the referees for their comments on the current manuscript.

REFERENCES

1. AARONSON, B. S. Time, time stance, and existence. In *The Study of Time*, Fraser et al., Eds. Springer-Verlag, 1972, New York, 293–311. (Reprinted. Original publication, 1971).
2. ACKOFF, R. L. *Creating the Corporate Future.* Wiley, New York, 1981.
3. AFSARMANESH, H., AND MCLEOD, D. A framework for semantic database models. In *New Directions for Database Systems*, G. Ariav and J. Clifford, Eds. Ablex, Norwood, N.J., 1986, 149–167.
4. ARIAV, G. Preserving the time dimension in information systems. Tech. Rep. DS-WP 83-12-06, Decision Sciences Dept., Univ. of Pennsylvania, Philadelphia, Dec. 1983. (Ph.D. dissertation).
5. ARIAV, G., BELLER, A., AND MORGAN, H. L. A temporal model. Tech. Rep. DS-WP 82-12-05, Decision Sciences Dept., Univ. of Pennsylvania, Philadelphia, Mar. 1983.
6. ASTRAHAN, M. M., AND CHAMBERLIN, D. D. Implementation of a structured English query language. *Commun. ACM 18*, 10 (Oct. 1975), 580–588.
7. BEN-ZVI, J. The time relational model. Ph.D. thesis, Dept. of Computer Science, Univ. of California, Los Angeles, 1982.
8. BOLOUR, A., ANDERSON, T. L., DEKETSER, L. J., AND WONG, H. K. T. The role of time in information processing: A survey. *ACM SIGMOD Rec.* (Spring 1982), 28–48.
9. BUBENKO, J. A. The temporal dimension in information modeling. In *Architecture and Models in Data Base Management Systems*, G. M. Nijssen, Ed. North-Holland, Amsterdam, 1977, 93–118.
10. BUBENKO, J. A. Information modeling in the context of system development. In *Information Processing 80*, S. H. Lavington, Ed. North-Holland/IFIP, Amsterdam, 1980, 395–411.

11. CHEN, P. P. S. The entity-relationship model—Toward a unified view of data. *ACM Trans. Database Syst. 1*, 1 (Mar. 1976), 9–36.
12. CHI, C. S. Advances in computer mass storage technology. *Computer 15*, 5 (May 1982), 60–74.
13. CLIFFORD, J. A logical framework for the temporal semantics and natural-language querying of historical databases. Ph.D. thesis, Dept. of Computer Science, SUNY at Stony Brook, Dec. 1982.
14. CLIFFORD, J., AND WARREN, D. S. Formal semantics for time in databases. *ACM Trans. Database Syst. 6*, 2 (June 1983), 123–147.
15. CODD, E. F. A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (June 1970), 377–387.
16. CODD, E. F. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst. 4*, 4 (Dec. 1979), 397–434.
17. COPELAND, G. What if mass storage were free? *Computer 15*, 7 (July 1982), 27–35.
18. DAYAL, U., AND BERNSTEIN, P. A. On the updatability of relational views. In *Proceedings of the 4th International Conference on Very Large Data Bases.* 1978, 368–377.
19. DOBBS, H. A. C. The dimensions of the sensible present. In *The Study of Time*, Fraser et al., Eds. Springer-Verlag, New York, 1972, 274–292. (Reprinted. Original publication, 1971).
20. FUJITANI, L. Laser optical disk: The coming revolution in on-line storage. *Commun. ACM 27*, 6 (June 1984), 546–554.
21. JONES, S., AND MASON, P. J. Handling the time dimension in a data base. In *ICOD Proc.*, (June 1980), 65–83.
22. KAHN, K. M. Mechanization of temporal knowledge. Tech. Rep. MIT-MAC-TR-155, MIT, Cambridge, Mass., Apr. 1975.
23. KEEN, P. G. W., AND SCOTT-MORTON, M. S. *Decision Support Systems: An Organizational Perspective.* Addison-Wesley, Reading, Mass., 1978.
24. KENT, W. *Data and Reality.* North-Holland, Amsterdam, 1978.
25. KENVILLE, R. F. Optical disk data storage. *Computer 15*, 7 (July 1982), 21–26.
26. KLOPPROGGE, M. R. TERM: An approach to include the time dimension in the entity-relationship model. In *Entity-Relationship Approach to Information Modeling and Analysis*, P. P. S. Chen, Ed. ER Institute, 1981, 477–512.
27. LANGEFORS, B. *Theoretical Analysis of Information Systems.* Student Litterature and Auerbach, Lund, Sweden, 1973. (First edition, 1966).
28. LANGEFORS, B., AND SUNDGREN, B. *Information Systems Architecture.* Petrocelli/Charter, New York, 1975.
29. LANING, L. J., WALLA, G. O., AND AIRAGHI, L. S. A DSS oversight—Historical databases. In *DSS-82 Transactions*, G. W. Dickson, Ed. June 1982, 87–95.
30. REITER, R. On closed world databases. In *Logic and Databases*, H. Gallaire and J. Minker, Ed. Plenum, New York, 1978, 55–76.
31. ROTHCHILD, E. Optical-memory media. *BYTE 8*, 3 (Mar. 1983), 86–106.
32. SHIFTAN, J. Assessing the temporal differentiation of attributes as an implementation strategy of temporally oriented relational databases. Ph.D. dissertation, Dept. of Computer Applications and Information Systems, New York Univ., Aug. 1986.
33. SNODGRASS, R. The temporal query language TQuel. In *Proceedings of the 3rd ACM SIGMOD Symposium on Principles of Database Systems* (Waterloo, Ont., Apr.). ACM, New York, 1984, 204–212.
34. TSICHRITZIS, D. C., AND LOCHOVSKY, F. H. *Data Models.* Prentice-Hall, Englewood Cliffs, N.J., 1982.

Received March 1985; revised December 1985; accepted December 1985