On Data Representation and Use In A Temporal Relational DBMS

James Clifford, Albert Croker and Alexander Tuzhilin

Department of Information, Operations and Management Sciences

Leonard N. Stern School of Business, New York University

44 West 4th Street, New York, NY 10012

(acroker, atuzhili)@stern.nyu.edu

ON DATA REPRESENTATION AND USE IN A TEMPORAL RELATIONAL DBMS

Abstract

Numerous proposals for extending the relational data model to incorporate the temporal dimension of data have appeared over the past decade. It has long been known that these proposals have adopted one of two basic approaches to the incorporation of time into the extended relational model. Recent work formally contrasted the expressive power of these two approaches, termed *temporally ungrouped* and *temporally grouped*, and demonstrated that the temporally grouped models are more expressive. IN the temporally ungrouped models, the temporal dimension is added through the addition of some number of *distinguished attributes* to the schema of each relation, and each tuple is "stamped" with temporal values for these attributes. By contrast, in temporally grouped models the temporal dimension is added to the types of values that serve as the domain of each ordinary attribute, and the application's schema is left intact. The recent appearance of TSQL2, a temporal extension to the SQL-92 standard based upon the temporally ungrouped paradigm, means that it is likely that commercial DBMS's will be extended to support time in this weaker way. Thus the distinction between these two approaches - and its impact on the day-to-day user of a DBMS - is of increasing relevance to the database practitioner and the database user community. In this paper we address this issue from the practical perspective of such a user. Through a series of example queries and updates, we illustrate the differences between these two approaches and demonstrate that the temporally grouped approach more adequately captures the semantics of historical data.

1 Introduction

Many database applications deal with temporal data. Virtually any on-line transaction processing system, such as airlines reservation, credit card approval, or an electronic banking system, collects and processes temporal data. Other examples would include human resources applications containing employment and compensation histories, financial applications, various scientific databases, and data warehousing.

Over the years, organizations have been storing this temporal data in "traditional" databases, using such relational DBMS's as DB2, Oracle, or Ingres, that do not provide any special support for managing temporal data beyond handling the data type DATE. The needed functionality for storing and querying temporal data was achieved by treating time as just another column in a relational table and simulating temporal database queries with regular SQL queries whenever this was possible, or resorting to programming using embedded SQL facilities [Elmasri and Navathe, 1994] whenever these temporal queries could not be implemented in SQL directly [Chalfin, 1994]. Since both of these solutions produced unnecessarily complicated queries, such an approach resulted in an excessive time and resource consuming application development efforts and in more error-prone applications.

Although cumbersome and difficult, this solution has out of necessity been the norm in the past [Chalfin, 1994]. However, competitive pressures and new trends in information systems, such as establishment of virtual corporations, automation of increasingly complex transactions, and developments of inter-organizational systems, force businesses to develop more complex database applications and at a faster speed. For this reason, implementing temporal database applications, such as trend analysis, audit trails, data warehousing and technical analysis in financial applications, with standard relational databases that do not provide any special support for time appears more and more as a loosing proposition. As McFadden and Hoffer observe in their textbook on database management [McFadden and Hoffer, 1994, p. 140]:

> "We have discussed the problem of time-dependent data with managers in several organizations who are considered leaders in the use of data modeling and database management. These discussions revealed that current data models (and database management systems based on those models) are generally inadequate in handling time-dependent data, and that organizations often ignore this problem and hope that the resulting inaccuracies balance out."

Over more than the past decade, there has been a growing body of research in the area of temporal databases aimed at addressing this problem. In fact, a growing series of bibliographies keeping track of the published research on the subject ([Bolour et al., 1982], [McKenzie, 1986], [Stam and Snodgrass, 1988], [Soo, 1991],and [Kline, 1993]) shows exponential growth! All of this research has contributed to our understanding of many of the facets of this interesting and important area in the management of data. At the same time, it has led to a growing consensus in how temporal data ought to be modeled and queried, and this consensus has led to considerable government and industry interest in the development of appropriate standards. Such standards (as any other database standards) would be able to reduce training costs, provide for more portable and longer lasting applications, and reduce dependency on a single vendor [McFadden and Hoffer, 1994, p. 285].

The need for a standardized view of modeling temporal information was recently recognized by both DARPA and NSF. In the summer of 1993 they sponsored a three-day international workshop aimed at developing a consensus on the logical and physical requirements for modeling temporal information in the next generation of SQL database management systems [Pissinou et al., 1994]. This workshop had 45 participants, including academic researchers, government observers, and representatives of some major vendors of database software. An outgrowth of this workshop was the creation of the TSQL2 Language Design Committee, whose mission was to develop an extension of the language SQL-92, called *TSQL2*, to incorporate treatment of the temporal dimension into SQL-92. The committee has issued a report [Snodgrass et al., 1994b] containing a complete syntactic extension to SQL, for which informal semantics has been provided via a series of "commentaries" some of which were also published as separate reports, such as "A TSQL2 Tutorial" ([Snodgrass et al., 1994a]), "A Consensus Glossary of Temporal Database Concepts" ([Jensen et al., 1994]). Finally, the complete description of the TSQL2 language is presented in the book [Snodgrass, 1995].

It was the goal of DARPA and NSF, and it is expected by the committee, that their report will have a widespread impact on the SQL industry. Specifically, it is expected that there will soon be TSQL2 implementations, upwardly compatible with SQL-92 ([Snodgrass et al., 1994b]), supplied by a number of major database vendors. It is also expected that the recommendations of this committee will significantly impact the proposal for SQL3, which is currently in the design stage but for which there is already an ISO-ANSI Working Draft ([Committee, 1993]). Thus, organizations interested in managing temporal data need to be aware of what is in the proposed TSQL2 standard, and perhaps have their voice heard before the SQL3 language design is finalized.

In [Clifford et al., 1994] we showed that all of the proposed temporal extensions to the relational model were of one of two types, temporally grouped or temporally ungrouped, and we examined formally the differences between these two types. In this paper we discuss and explore the practical ramifications of the theoretical distinction between these two paradigms of temporal relational data modeling for *users* of commercial DBMS's based on one or the other paradigm. In particular, we argue that the temporally ungrouped approach has some severe limitations and that the temporally grouped approach, based on viewing temporal data as time series, solves these limitations. It is important to note that, in order to be as close as possible to the SQL-92 standard, the TSQL2 Design Committee decided to base TSQL2 on the temporally ungrouped paradigm for temporal relational data modeling. Therefore, as we shall discuss, the language represents a hybrid between an inherently temporally grouped approach and a temporally ungrouped approach that simulates grouping by means of an explicit grouping construct (called a surrogate) whose maintenance is almost entirely up to the user. While the goal of this compromise solution was clearly stated to be the maintenance of "upward compatibility" with the SQL-92 standard, it appears that the design decisions made for TSQL2 may have a strong influence on the final design of the temporal component of SQL3. We therefore point out some problems with the simulated grouping mechanism that is incorporated into TSQL2, and conclude by arguing that inherent support for temporal grouping *at the conceptual level* is the appropriate model to adopt for SQL3.

The rest of the paper is organized as follows. In the next section we take a brief aside to introduce some basic temporal database terminology, in order to put the remainder of the paper into perspective. Then in Section 3 we discuss the distinction between the representation of temporal information in temporally grouped and temporally ungrouped models. Finally, we look at how the differences between these two representation paradigms affect how users update (Section 4) and query (Section 5) the information in the database. We conclude in Section 6 with a summary of our discussion and some directions for future research.

2 Temporal Databases

In this section we provide an overview of the major issues that arise in the modeling of temporal information, and discuss approaches for dealing with them in the temporal data models that have been proposed in the literature. [Tansel et al., 1993] presents a good overview of the state of the art of the field of temporal databases, and indicates directions for future research.

2.1 Kinds of Time

A dominant area of research on temporal databases has focused on the proper way of incorporating time as an intrinsic component of the underlying data model. Since perhaps the dissertation of BenZvi [Ben-Zvi, 1982], it has been recognized that multiple temporal dimensions can be associated with data. Thus, one distinguishing characteristic of temporal data models is the number and kind of temporal dimensions supported. Among the dimensions that have been proposed, it is widely accepted that there are two principle temporal dimensions to data stored in a temporal database, the *valid time* of the data and the *transaction time* of the data. Although the representation of the temporal dimensions may vary depending upon the particular model considered, typically they are represented either as a collection of time points or of time intervals.

Valid Time According to [Jensen et al., 1992, Jensen et al., 1994], "the *valid time* of a fact (i.e., datum) is the time when that fact is true in the modeled reality." Most of the temporal data models that have appeared in the literature have incorporated valid time as the single temporal dimension. These data models, commonly called *valid-time* or *historical* models, and sometimes real-world time, intrinsic time, logical time, or data time models include, among others, the models proposed in [Jones and Mason, 1980, Ben-Zvi, 1982, Clifford and Warren, 1983, Ariav, 1986, Tansel, 1986, Clifford and Croker, 1987, Lorentzos, 1987, Snodgrass, 1987, Gadia, 1988a, Navathe and Ahmed, 1989, Sarda, 1990].

Transaction Time According to [Jensen et al., 1992], " the *transaction time* of a database fact is the time when the fact is current in the database and may be retrieved." Unlike valid time, transaction time is not under the explicit control of the user. For example, it could correspond to the transaction *timestamps* used to serialize a system's set of transactions, and it cannot be changed. Transaction time, also called registration time, extrinsic time, physical time, transaction commit time, or database time, is used to model the changing state of the database's *knowledge* of its facts and when they became known. Few of temporal data models that have appeared in the literature have incorporated only the transaction time dimension. These data models, called either *transaction-time* or *rollback* models, include the models proposed in [Jensen et al., 1989, Lomet and Salzberg, 1992].

Both Valid Time and Transaction Time There have been a few models, called *bitemporal* data models in [Jensen et al., 1992], which have incorporated both temporal dimensions, including [Ben-Zvi, 1982, Snodgrass, 1987, McKenzie and Snodgrass, 1991, Gadia, 1992]. Moreover, a few models have tried to generalize the notion of temporal dimensions of data to a general treatment of data dimensions, for example the spatial dimension, or the observer dimension. Models of this variety include [Clifford, 1992], [Gadia and Nair, 1993], and [Lorentzos, 1993].

2.2 Different Manners of Incorporating Time

Another aspect which has distinguished temporal data models in the literature relates to the manner in which the temporal dimensions are incorporated into a data model. Specifically, how is the temporal dimension associated with a given "fact," and what constitutes a "fact" in the first place.

Two general approaches have been taken to this problem.

Attribute Timestamping In this approach, the "fact" with which a temporal dimension is associated is considered to be the value of an attribute. The resulting time-varying attribute can thus be viewed as a function from the temporal dimension into the underlying domain of values for that attribute. For example, an Employee's SALARY would, in this approach, be viewed not as a single value such as $35,000, but rather as a function which, for each time, specifies the employee's salary at that time. For example, the salary function of a particular employee might specify a salary of $30,000 from 1987 through 1989, $32,500 from 1990 through 1992, and $35,000 from 1993 through the present. Since attribute timestamping essentially amounts to treating as a fundamental data type what in statistics is called a time series, it has long been recognized that in this approach relations are no longer in first normal form.

Tuple Timestamping In this approach, a "fact" is considered to be a tuple, and the temporal dimension is associated with all of the information in the (full) tuple. This approach has often been referred to in the literature as "tuple timestamping." Depending on the model being considered, anywhere from one ([Lorentzos, 1987]) to two ([Sarda, 1993]) to four ([Snodgrass, 1987]) or even five [Ben-Zvi, 1982, Gadia, 1993] distinguished temporal attributes have been incorporated into the schema to "timestamp" each tuple. For example, a tuple for an employee might look something like <John, Marketing, $35,000, 1988> in one such model, or <John, Marketing, $35,000, 1988, 1990> in another. Since the temporal dimension is associated with the full tuple, each timestamp can be viewed as a *distinguished* attribute of the tuple and the relation can still, as these examples illustrate, be kept in first normal form.

Comparison of Two Approaches In [Clifford et al., 1994] we explored the difference between the so-called attribute timestamping and tuple timestamping approaches to incorporating time into the relational model. We termed these two approaches *temporally grouped* and *temporally ungrouped*, respectively, to more accurately reflect the intent of their modeling approach. We next argued that, contrary to popular belief, the two approaches were not just "two different ways of doing the same thing." In fact we proved that the simple temporally ungrouped models in the literature were *not as expressive* as the temporally grouped models. We then demonstrated a technique for augmenting the temporally ungrouped models with an additional explicit grouping attribute which could simulate the inherent grouping of the temporally grouped models.

In the next sections we explore more fully the differences between these two modeling approaches, and then demonstrate that these differences affect the way that users must interact with the database to perform the ordinary functions of updating and querying the information that it contains.

3 Temporally Ungrouped and Temporally Grouped Relational Models

In this section we discuss and define canonical relational structures for the temporally grouped and temporally ungrouped approaches mentioned in Section 2.2. These relational structures are used in later sections to discuss other aspects of temporal relational data models, in particular, querying and updating. To make the discussion concrete, we first present an example application that we will use throughout the paper in order to illustrate the fundamental difference between these two approaches.

3.1 An Example Enterprise

Consider the following simplification of a typical business application that might benefit from the use of a temporal database. We choose this application because it was used in [Jensen (ed.), 1993] to serve as a generic application which was used both to illustrate the semantics of the data model of TSQL2, as well as to gauge the expressiveness of its query language. The detailed description of this application, as taken from [Jensen (ed.), 1993], can be found in the Appendix. It contains employment histories (**EMP**) of various persons that worked for an organization, the history of a set of departments in that organization (**DEPT**), and a list of the skills that employees have. In particular, **EMP** models employment histories by modeling the histories of their `Name`, `Salary`, `Gender`, and date of birth (`D-birth`) attributes. The **DEPT** entity models the histories of the department's `Name` and `Budget`. Moreover, [Jensen (ed.), 1993] describes various relationships between these entities, the detailed description of which can be found in the Appendix[1].

To simplify our discussion, we assume that there are only two employees to be modeled in the database. In order to distinguish between an entity, such as an *employee*, and the value of some attribute of that entity, such as **Name**, we will refer to these two employees as $\mathcal{ED}$ and $\mathcal{DI}$. Note that the histories of $\mathcal{ED}$ and $\mathcal{DI}$ are stated here in English, and not in any specific data model, so as not to bias the reader toward any particular *representation* of this information. Later, in Figures 1 and 2, we will contrast two methods for representing this information in tables in two different extended relational models from the literature.

$\mathcal{ED}$ worked in the Toy department from 2/1/82 to 1/31/87, and in the Book department from 4/1/87 to the present. From 4/1/87 to the present, he managed the Book department. The budget

[1]For simplicity of this exposition, all attributes of the database are assumed to be temporal, and to be defined over the same time period (what Gadia [Gadia, 1988b] termed temporal homogeneity). In [Clifford et al., 1995] we present a more general model that relaxes these restrictions.

of that department has been \$50K since $\mathcal{ED}$ became its manager. $\mathcal{ED}$'s name was "Ed" from 2/1/82 to 12/31/87, and "Edward" from 1/1/88 to the present. His salary was \$20K from 2/1/82 to 5/31/82, then \$30K from 6/1/82 to 1/31/85, then \$40K from 2/1/85 to 1/31/87 and 4/1/87 to the present. $\mathcal{ED}$ is male and was born on 7/1/55. Several skills are recorded for $\mathcal{ED}$. He has been qualified for typing since 4/1/82 and qualified for filing since 1/1/85. He was qualified for driving from 1/1/82 to 5/1/82, and from 6/1/84 to 5/31/88.

$\mathcal{DI}$ worked in and managed the Toy department from 1/1/82 to the present. Her name has been "Di" throughout her employment. The budget of the Toy department was \$150K from 1/1/82 to 7/31/84, \$200K from 8/1/84 to 12/31/86, and \$100K from 1/1/87 to the present. $\mathcal{DI}$'s salary was \$30K from 1/1/82 to 7/31/84, \$40K from 8/1/84 to 8/31/86, then \$50K from 9/1/86 to the present. $\mathcal{DI}$ is female and was born on 10/1/60. $\mathcal{DI}$ has been qualified for directing from 1/1/82 to the present.

3.2 Temporally Ungrouped Models

Temporally ungrouped models use the tuple timestamping approach discussed in Section 2.2. These models support either valid-time, or transaction-time, or both kinds of time. Following the work of [Clifford et al., 1994], we consider only the valid-time temporally ungrouped models in the paper.

Figure 1 shows an example of one commonly proposed convention for incorporating temporal attributes into a 1NF relation. In this approach, each relation is required to include among its attributes a distinguished temporal attribute (*VALID-TIME*) that specifies the interval of temporal validity of the corresponding tuple. For example, the tuple (Toy, 150, Di, [1/1/82 - 7/31/84]) from Figure 1(b) specifies that the budget of the Toy department when $\mathcal{DI}$ was its manager was 150 from 1/1/82 until 7/31/84. This interval of temporal validity is also called the *lifespan* of a tuple [Clifford and Croker, 1987].

The granularity of the lifespan is generally not considered to be an intrinsic property of a model, but is more appropriately determined by the application being modeled by a given database. For our examples we use the granularity *DAY*. For other applications *SECOND*, *HOUR*, *WEEK* or even *YEAR* might be more appropriate. For the right end-point of the lifespan we assume an additional value *NOW* which is used to denote a moving time reference that always represents the current time.

The temporally ungrouped model is a direct extension of the standard relational model to incorporate time. It is simple, easy to understand and compatible with the standard relational model. However, the temporally ungrouped model also has certain limitations.

Name	Salary	Gender	D-birth	DeptName	VALID-TIME
Ed	20	M	7/1/55	Toy	[2/1/82 - 5/31/82]
Ed	30	M	7/1/55	Toy	[6/1/82 - 1/31/85]
Ed	40	M	7/1/55	Toy	[2/1/85 - 1/31/87]
Ed	40	M	7/1/55	Book	[4/1/87 - 12/31/87]
Edward	40	M	7/1/55	Book	[1/1/88 - *NOW*]
Di	30	F	10/1/60	Toy	[1/1/82 - 7/31/84]
Di	40	F	10/1/60	Toy	[8/1/84 - 8/31/86]
Di	50	F	10/1/60	Toy	[9/1/86 - *NOW*]

(a) **EMP**

Name	Budget	MgrName	VALID-TIME
Toy	150	Di	[1/1/82 - 7/31/84]
Toy	200	Di	[8/1/84 - 12/31/86]
Toy	100	Di	[1/1/87 - *NOW*]
Book	50	Ed	[4/1/87 - 12/31/87]
Book	50	Edward	[1/1/88 - *NOW*]

(b) **DEPT**

Figure 1: Temporally Ungrouped Relations **EMP** and **DEPT**.

A major, although not necessarily obvious, problem with temporally ungrouped models is that they lack any inherent mechanism for associating those tuples in a relation that together model the same real world object. Most of the proposals found in the literature for such models have, implicitly or explicitly, assumed that each object represented in their temporally ungrouped relations could be uniquely identified by the values of some subset of the relation's attributes. In other words, these models assume that for each object modeled by a relation the values of these attributes, which together with the temporal attributes would form a key to the relation, are constant-valued over time. For example, in the **DEPT** relation the combination of NAME and VALID-TIME would be assumed to constitute a key. A direct consequence of this assumption is that, if a DEPT ever changes its NAME, these models would never be able to associate the information from all of the tuples with these two different NAMES as belonging to the same real-world object.

The problem with this approach is that the specification of such a set of attributes in a relation is based on the semantics of an application; it is not an inherent property of a relation. Further, requiring the use of such attributes runs counter to the spirit of the goals of temporal databases to store information as it evolves over time, and as our knowledge about it evolves over time. Specifically, we believe that a temporal data model should not assume *a priori* that every application can identify a set of attributes which are assured to remain unchanged over time. Even attributes that we may intuitively feel to be time-invariant, such as a social security number or financial security identifiers such as CUSIP, are in the real world known to change. Note that in the **EMP** relation the Name attribute, the obvious choice for identifying an employee, is not appropriate since at some point in time, $\mathcal{ED}$ changed his name from Ed to Edward.

Thus, in temporally ungrouped models, if for some application no such set of attributes can be specified, the connections between tuples in a relation that relate to the same employee, may be lost. In the next section we discuss a type of temporal model that remedies this problem inherent in the temporally ungrouped models.

3.3 Temporally Grouped Models

A second set of proposals for extending the relational model ([Clifford, 1982, Tansel, 1986, Clifford and Croker, 1987, Gadia, 1988a, Grandi and Scalas, 1991]) breaks free of the first normal form constraint of the standard relational data model. Under these proposals it becomes possible to represent all of the data pertaining to a real world object in a single *historical tuple* which *groups* together all of the information about that object. It is for this reason that we call these models temporally grouped.

In a temporally grouped model there are no additional distinguished temporal attributes. Rather, each tuple of a temporally grouped relation can contain multiple values for each of its attributes, because each value is associated with a time interval that indicates the time for which the associated value is (or was) valid. Figure 2 shows the temporally grouped analogs of the temporally ungrouped relations in Figure 1.

There are several obvious differences between temporally grouped relations and their temporally ungrouped counterparts. Looking at the temporally grouped *EMP* relation we see first that each real-world entity represented in the temporally grouped variant is modeled by a single tuple. Second, there are no timestamp attributes. The interval in which each value is valid, i.e., its lifespan, is incorporated along with the value, and (because of homogeneity) the aggregate or union of the intervals of all the attributes of a tuple are equivalent. For example $\mathcal{ED}$ has a value for each attribute for every time in the intervals [2/1/82 - 1/31/87] and [4/1/87 - *NOW*].

Finally, we wish to make two points with respect to the temporally grouped paradigm and the model presented here. First, it is clear that there is a relationship between the grouped vs. ungrouped dichotomy of temporal relations, on the one hand, and the more general dichotomy of the First vs. Non-First Normal Form (1NF and N1NF) relations [Jaeschke and Schek, 1982], [Roth et al., 1988], [Tansel and Garnett, 1992] on the other hand. However, we believe that the contrasting approaches of handling time in either a 1NF (tuple-timestamping, or temporally ungrouped) or a N1NF (attribute time-stamping or temporally grouped) fashion can be viewed as orthogonal and in some sense independent of the choice of the representation of the data itself. In other words, a temporally grouped model is only N1NF in the way that it incorporates the temporal dimension (allowing time series as a primitive data type, and providing decomposing operators to access the domain and the range of these time series functions). Thus, for example, the model that we discuss in this paper is temporally grouped, but is not a fully N1NF model, whereas the temporally grouped model of Tansel ([Tansel, 1993]) is fully N1NF.

Second, for reasons of simplicity — and to make a more direct comparison with the temporally ungrouped approach — we have here modeled each attribute as taking a time series as a value, including an arguably time-invariant attribute as *Gender*, and a clearly temporally valued attribute as *D-birth*. In [Clifford et al., 1995] we consider a more general temporally grouped inhomogeneous model which allows for attribute values of three different sorts: simple values, time values, and time-series values.

Name	Salary	Gender	D-birth	DeptName
{[2/1/82-1/31/87], [4/1/87-12/31/87]} → Ed [1/1/88-NOW]} → Edward	{[2/1/82-5/31/82]} → 20 {[6/1/82-1/31/85]} → 30 {[2/1/85-1/31/87], [4/1/87-NOW]} → 40	{[2/1/82-1/31/87], [4/1/87-NOW]} → M	{[2/1/82-1/31/87], [4/1/87-NOW]} → 7/1/55	{[2/1/82-1/31/87]} → Toy { [4/1/87-NOW]} → Book
{[1/1/82-NOW]} → Di	{[1/1/82-7/31/84]} → 30 {[8/1/84-8/31/86]} → 40 {[9/1/86-NOW]} → 50	{[1/1/82-NOW]} → F	{[1/1/82-NOW]} → 10/1/60	{[1/1/82-NOW]} → Toy

(a) **EMP**

Name	Budget	MgrName
{[1/1/82-NOW]} → Toy	{[1/1/82-7/31/84]} → 150 {[8/1/84-12/31/86]} → 200 {[1/1/87-NOW]} → 100	{[1/1/82-NOW]} → Di
{[4/1/87-NOW]} → Book	{[4/1/87-NOW]} → 50	{[4/1/87-12/31/87]} → Ed {[1/1/88-NOW]} → Edward

(b) **DEPT**

Figure 2: Temporally Grouped Relations **EMP** and **DEPT**

3.4 Surrogates and Keys

The goal of any data model is to appropriately and adequately model the "objects" (in a neutral sense, indicating entities and/or relationships) of interest to its users. In order to achieve this goal, it is necessary that the model be able to uniquely identify and reference data associated with each object being modeled. In the traditional relational data model this association was accomplished through the use of primary keys.

In a temporal data model, which is intended to model the *history* of objects over some period of time, it is possible that there is no collection of attributes of an object that remains constant over time. That is, it is possible that there is no *time-invariant* key. For example, although in the conceptual model discussed in the previous section no two employees are assumed to have the same name at the same point in time (`Name` is an entity key), it is possible that at some point an employee does undergo a name change. In fact, in the data instance associated with this conceptual model, $\mathcal{ED}$ undergoes a name change on 1/1/88 from Ed to Edward.

The proposal in [Snodgrass et al., 1994] addresses this issue when it assumes that `Name` in the **EMP** relation is only a *snapshot primary key*, i.e. it determines uniquely the rest of the tuple *only* at individual time values[2]. Moreover, [Snodgrass et al., 1994] goes on to say that

> It is emphasized that the notion of key does not capture correspondence between attribute values and the real-world objects they represent. As one consequence, it is possible in this ER schema, e.g., for an employee to change `Name` attribute value over time.

Since it is possible that all of the data attributes associated with an object can vary over time, the adequacy of a temporal relational extension should, in part, be judged on its ability to identify in its temporal relational structure all of the data associated with a given object modeled in that relation. In order to distinguish the identification of objects modeled by a temporal relation and the methods used to identify tuples in a relation, we will use the term **surrogate** (first introduced in [Codd, 1979]) to refer to a unique object identifier, and **key** to refer to a collection of one or more relation (data) attributes that are used to uniquely identify tuples in a relation (the usual definition).

It is our belief that user-defined, time-invariant keys, are impractical in temporal databases, and are contrary to the spirit of a temporal database – storing information as it evolves over time, and

[2]More precisely, for any time value t, `Name`, as it is known at time t, uniquely determines the rest of the attributes taken at time t [Snodgrass et al., 1994].

as our knowledge about it evolves over time. It is well-known that there are no good time-invariant keys; even such invented keys as SSN's have to be changed occasionally. In a temporal database, therefore, it is unreasonable to impose this strict requirement. Thus, the temporal database community developed the notion of a *snapshot key* [Jensen et al., 1994] as the appropriate extension to the temporal case of the notion of a relational key in so-called *static* or *snapshot* relations.

However, this illustrates an obvious problem with temporally ungrouped models — with the information about some real-world entity or relationship stored in multiple tuples, how does a user get all the information about the objects of interest? The traditional function of a *key*, providing unique identification of the record (tuple) for a desired object, can no longer be relied upon. By contrast, the key to a relation in the temporally grouped approach is in fact a temporal function. In the **EMP** relation, for example, no two employees can have the same *Name* history, nor can two different employees have the same *Name* at the same point in time, though the same *Name* could be used by different employees at different points in time.

Group id's or *surrogates*, a special type maintained carefully by the system to function as time-invariant identifiers, were proposed as a solution to this problem in temporally ungrouped models ([Clifford et al., 1994]). In the next section we illustrate how group IDs can be added to a temporally ungrouped model, to simulate the inherent grouping of the temporally grouped models.

3.5 Temporally Ungrouped Models With Surrogates

An alternative, and temporally ungrouped, approach to relating all of the data in a relation that pertains to a single object was proposed in [Clifford et al., 1994]. In this approach, which can be viewed as a compromise between the temporally ungrouped models discussed earlier and the temporally grouped model discussed in the previous section, a second type of distinguished attribute, a *grouping* attribute that we label *ID* is incorporated into each temporally ungrouped relation.

The grouping attribute *ID* serves the role of a surrogate that is used to bind together, through the use of a unique and time-invariant value, all of the tuples of a temporally ungrouped relation that relate to a single real world object. For example, in the **EMP** relation of Figure 3 the first five tuples all pertain to a single employee. Thus we have given each of these tuples the same *ID* value, *100*.

In the proposal of [Clifford et al., 1994] the value of *ID* is *system-generated* and *system-maintained*, and the actual values are available in only a very restricted way to the user. In addition, it was shown that with certain rather strict constraints on the use of the surrogates, these attributes are adequate for binding together a collection of tuples that pertain to a single real-world object. We

ID	Name	Salary	Gender	D-birth	Dept	TIME
100	Ed	20	M	7/1/55	Toy	[2/1/82 - 5/31/82]
100	Ed	30	M	7/1/55	Toy	[6/1/82 - 1/31/85]
100	Ed	40	M	7/1/55	Toy	[2/1/85 - 1/31/87]
100	Ed	40	M	7/1/55	Book	[4/1/87 - 12/31/87]
100	Edward	40	M	7/1/55	Book	[1/1/88 - *NOW*]
101	Di	30	F	10/1/60	Toy	[1/1/82 - 7/31/84]
101	Di	40	F	10/1/60	Toy	[8/1/84 - 8/31/86]
101	Di	50	F	10/1/60	Toy	[9/1/86 - *NOW*]

Figure 3: Temporally Ungrouped Relation **EMP** with Surrogates

emphasize that [Clifford et al., 1994] was *not* a proposal to incorporate surrogates as a user-level, conceptual model construct. Rather, in that work we showed that without the addition of some new construct, the temporally ungrouped models as proposed in the literature are strictly less expressive than the temporally grouped models. Surrogates were introduced there as a purely formal mechanism for proving that it was possible to add one additional column, with certain constraints, to achieve a formally equivalent model. In this paper we will argue in detail that the inherently temporally grouped approach is a more natural conceptual level model.

The TSQL2 language proposal supports a surrogate type in a manner similar to the technique proposed in [Clifford et al., 1994]. However, TSQL2 proposes surrogates as a conceptual model construct, i.e., surrogate support is not fully automatic. Specifically, TSQL2 supports a data type SURROGATE which the user is free to incorporate (or not) into some or all of the base tables. The values of a SURROGATE attribute are assigned and removed by the system; the user cannot modify or even "see" them. The only operation allowed on the SURROGATE type is comparison for equality. Surrogates can appear in the SELECT clause of a *nested* query but not in the outermost SELECT statement.

TSQL2 provides a good approach to handling temporal grouping through the use of surrogates. However, it partially delegates the task of defining and maintaining temporal grouping to the user by letting him or her define SURROGATE attributes in temporal relations, and allowing explicit reference to surrogates in queries rather than letting the DBMS define and maintain temporal grouping entirely on its own. This approach requires more of a user and, therefore, is more error-prone than the alternative approach of letting the DBMS handle grouping. Furthermore, TSQL2's solution to the temporal grouping problem has some other problems that we discuss in Section 5.

In the rest of the paper, we will explore updates and queries in the context of the temporally grouped/ temporally ungrouped modeling distinction.

4 Updates in Temporally Grouped and Ungrouped Models

In this section we discuss the process of updating a temporal database and highlight the differences between updating in the temporally grouped and temporally ungrouped approaches. To be specific, consider the case of our employee $\mathcal{ED}$ who, as we discussed in Section 3.1, works for our enterprise. $\mathcal{ED}$ began his employment on 2/1/82, and wanted to be known by the name "Ed". Sometime on or about 1/1/88 our employee $\mathcal{ED}$ informs his company that as of 1/1/88 he wants his name to be "Edward."

We claim that such a change (and we use this specific change merely as an illustrative example) is supported in a temporally grouped model in a more direct and natural way than in the temporally ungrouped model. To substantiate this claim, let us consider this change in the context of the temporally grouped and temporally ungrouped models, respectively.

In the temporally ungrouped model, the information about $\mathcal{ED}$ before his request for a name change might look like the following:

Name	Salary	Gender	D-birth	DeptName	V
Ed	20	M	7/1/55	Toy	$\{[2/1/82 - 5/31/82]\}$
Ed	30	M	7/1/55	Toy	$\{[6/1/82 - 1/31/85]\}$
Ed	40	M	7/1/55	Toy	$\{[2/1/85 - 1/31/87]\}$
Ed	40	M	7/1/55	Book	$\{[4/1/87 - NOW]\}$

whereas in the temporally grouped case, the same information would be represented as:

Name	Salary	Gender	D-birth	DeptName
{[2/1/82-1/31/87], [4/1/87-NOW] } $\rightarrow$ Ed	{[2/1/82-5/31/82]} $\rightarrow$ 20 {[6/1/82-1/31/85]} $\rightarrow$ 30 {[2/1/85-1/31/87]} $\rightarrow$ 40 {[4/1/87-NOW]} $\rightarrow$ 40	{[2/1/82-1/31/87], [4/1/87-NOW]} $\rightarrow$ M	{[2/1/82-1/31/87], [4/1/87-NOW]} $\rightarrow$ 7/1/55	{[2/1/82-1/31/87]} $\rightarrow$ Toy { [4/1/87-NOW]} $\rightarrow$ Book

In the case of a temporally ungrouped model, the change of $\mathcal{ED}$'s name would involve the following steps: (a) stop the validity of the Name "Ed" for $\mathcal{ED}$, and (b) start the validity of the Name "Edward" for this same employee. More specifically, the following operations are required:

1. Modify the temporal component of the "current" tuple for $\mathcal{ED}$ changing the interval of its validity from $\{[4/1/87 - NOW]\}$ to $\{[4/1/87 - 12/31/87]\}$. This operation is called a "logical delete."

2. Insert a new tuple for $\mathcal{ED}$ whose values for all of the user-defined attributes other than **Name** are copied from the current $\mathcal{ED}$ tuple, whose value for **Name** is "Edward", and whose interval of validity is $\{[1/1/88 - NOW]\}$.

These two actions at the conceptual level can be achieved in TSQL2 with a single operation, UPDATE:

```
UPDATE Emp
    SET NAME TO ''Edward'' VALID PERIOD [1/1/88, NOW]
    WHERE Name = ''Ed'' VALID 1/1/88
```

This operation "deletes" the tuple for $\mathcal{ED}$ which is valid on 1/1/88 by terminating its valid time and inserts a new tuple for $\mathcal{ED}$ into **Emp** relation with the value of **Name** being "Edward" and the temporal validity interval [1/1/88, NOW]. These two changes result in the following information for $\mathcal{ED}$ in the database:

Name	Salary	Gender	D-birth	DeptName	V
Ed	20	M	7/1/55	Toy	$\{[2/1/82 - 5/31/82]\}$
Ed	30	M	7/1/55	Toy	$\{[6/1/82 - 1/31/85]\}$
Ed	40	M	7/1/55	Toy	$\{[2/1/85 - 1/31/87]\}$
Ed	40	M	7/1/55	Book	$\{[4/1/87 - 12/31/87]\}$
Edward	40	M	7/1/55	Book	$\{[1/1/88 - NOW]\}$

In the case of a temporally grouped model, the DBMS would have to take only the following single action:

Modify the value of the **Name** attribute of $\mathcal{ED}$'s tuple, terminating the validity of the name "Ed" as of 12/31/87, and initiating the validity of the name "Edward" as of 1/1/88.

While this update could be expressed in the update language of a temporally grouped model in precisely the same way, the action taken would, unlike in the case of the temporally ungrouped model, be localized to a *single* attribute **Name**, and would result in the following new value for $\mathcal{ED}$'s **Name** attribute:

Name
{[2/1/82-1/31/87], [4/1/87-12/31/87] } → Ed {[1/1/88-NOW]} → Edward

If we compare the effects of updates in the temporally grouped and the temporally ungrouped models, we observe that the update of an attribute in the temporally grouped model is localized to that attribute only and does not produce any new tuples. Note that this is in line with what the user intuitively expects from this type of an update. In contrast to this, the update in the temporally ungrouped model creates a *new* tuple and, thus, its effects are *not* limited only to the attribute being updated. This type of behavior is clearly counterintuitive to what an update should do from the end-user standpoint, i.e., make changes only to a single attribute. In addition, the update for the temporally ungrouped case is more involved than the update for the temporally grouped case at the implementation level because it requires an insertion of a new tuple.

5 Querying Temporally Grouped and Ungrouped Models

As we did in the previous section for updates, we explain in this section how temporally grouped and ungrouped models can be queried and highlight the differences between temporally grouped and ungrouped query languages. We begin our presentation with a query language for the temporally grouped model.

5.1 Querying Temporally Grouped Models

There have been several query languages proposed for the temporally grouped model in the past, including [Clifford and Tansel, 1985, Tansel, 1986, Clifford and Croker, 1987, Gadia, 1988a, Grandi and Scalas, 1991]. In [Clifford et al., 1994], a temporal calculus for the temporally grouped model, L_h, was presented, and it was argued that this calculus has all the minimally necessary important features that a temporally grouped model should have. Therefore, it was argued that this language should serve as a basis for the *temporally grouped historical relational completeness*, i.e. this language should serve as a "greatest common denominator" of temporally grouped query languages.

To give a flavor of this language, we provide examples of some of the queries expressed in L_h.

However, to make the presentation more intuitive, we present these queries using SQL-like syntax that is less technical than the syntax of L_h. We call this language SQL_h.

Example 1 Find names and salary histories of employees who worked in the Toy department on July 4, 1987.

SELECT	EMP.Name, EMP.Salary : Time2
FROM	EMP : Time1, Time2
WHERE	EMP.DeptName.Time1 = "Toy" AND Time1 = "7/4/87"

This SQL_h query first finds all the temporally grouped tuples of employees that on 7/4/87 had DeptName = "Toy." Then for these tuples it retrieves employee's name and salary histories. This query on our example database from Figure 2(a) returns the following answer:

Name	Salary (K)
[1/1/82 - *NOW*] → Di	[1/1/82 - 7/31/84] → 30 [8/1/84 - 8/31/86] → 40 [9/1/86 - *NOW*] → 50

The syntax of this query can be interpreted as follows. EMP in the **FROM** clause of this query can be interpreted as a *historical tuple* (see Section 3) ranging over the temporally grouped relation with the same name (EMP). For example, in the EMP relation from Figure 2(a), historical variable EMP can be associated either with the first tuple ($\mathcal{ED}$) or with the second tuple ($\mathcal{DI}$). Then the expression "EMP : TIME1, TIME2" in the **FROM** clause means that TIME1 and TIME2 are temporal variables ranging over the lifespan of the tuple EMP. For example, if EMP is associated with $\mathcal{DI}$ in Figure 2(a), then TIME1 and TIME2 range over the lifespan [1/1/82 - NOW]. Then the expression EMP.DeptName.Time1 = "Toy" is interpreted as follows. For some time Time1 ranging over the lifespan of historical tuple EMP, the value of the attribute DeptName at that time (Time1) is equal to "Toy." Then the **SELECT** clause says that we want to retrieve Name and Salary attributes of all the historical tuples EMP restricted to those times Time2 that satisfy the conditions of the **WHERE** clause. Note that in this case the **WHERE** clause imposes only restrictions on the historical variable EMP (e.g. only the tuple corresponding to $\mathcal{DI}$ is retrieved). There is no restriction on the temporal variable Time2 (because it does not appear in the **WHERE** clause), and thus Time2 can take any value from the lifespan of EMP; thus the entire history of the tuple appears in the answer. However, this is not true in general, as will be shown in Example 5.

Example 2 Who are the managers of the departments for whom Ed has worked.

```
SELECT   E1.Name : Time1
FROM     EMP AS E1 : Time1 , EMP AS E2 : Time2, DEPT : Time2
WHERE    E2.Name.Time2 = "Ed" AND E2.DeptName.Time2 = DEPT.Name.Time2
         AND DEPT.MgrName.Time2 = E1.Name.Time2
```

The syntax of this query says that historical variables E1 and E2 range over relation EMP. Moreover, temporal variable Time1 ranges over the lifespan of E1 and temporal variable Time2 over the lifespan of E2 *and* over the lifespan of historical variable DEPT. Note that the temporal variable Time2 does not appear in the **SELECT** clause and therefore, as in the standard SQL, is existentially quantified, i.e., it is interpreted as "there *exists* some time Time2 such that ..." (e.g. Name of E2 at that time was "Ed"). Note that variable E1 ranges over relation EMP, and refers to the manager's record in that relation. This means, among other things, that E1.Name provides the name history of that manager (i.e., how manager's name changed over time). Moreover, E1.Name : Time1 restricts this history to the times Time1 that satisfy the conditions of the WHERE clause (but since Time1 does not have any restrictions, E1.Name : Time1 returns the entire history of that manager/employee).

This query finds all the tuples E2 in EMP relation that have "Ed" as one of the names in their lifespans. Then it joins those tuples with the DEPT relation having the same department name at the same time, and retrieves the manager names and the corresponding times from the E1 relation that correspond to the joined tuples from the DEPT relation. This query returns the following answer evaluated on relations from Figure 2.

Name		
[1/1/82 - NOW]	→	Di
[2/1/82 - 1/31/87]	→	Ed
[4/1/87 - 12/31/87]	→	Ed
[1/1/88 - *NOW*]	→	Edward

As we can see from these examples, SQL_h is a temporally grouped language that operates on historical tuples by using historical variables (such as E1 and E2 from Example 2). It also supports temporal variables, such as Time, Time1 and Time2 from Examples 1 and 2 that allow accessing individual time instances within the lifespans of historical tuples.

5.2 Querying Temporally Ungrouped Models

In Section 1 we discussed the recent proposal of the TSQL2 Language, and the effort in the temporal database community to propose this language as the standard query language for the temporally ungrouped data model. Therefore, we will use this language in the paper as a representative example of temporally ungrouped query languages. To give a flavor of this language, we will express the queries from Section 5.1 in TSQL2.

Example 3 Consider the query from Example 1, i.e. "Find names and salary histories of employees who worked in the Toy department on July 4, 1987." It can be expressed in TSQL2 as

```
SELECT  E2.Name, E2.Salary
FROM    EMP As E1 E2
WHERE   E1.DeptName = "Toy" AND E1 contains | 7/4/87 |
        AND E1.Name = E2.Name
```

This query uses tuple variable *E1* to find employees who worked in the Toy department on the specified date, and then uses tuple variable *E2* to return all of the historical Name and Salary information (in potentially many tuples) about employees with the same Name. The answer to this query evaluated on the relation from Figure 1(a) is

Name	Salary	VALID
Di	30	[1/1/82 - 7/31/84]
Di	40	[8/1/84 - 8/31/86]
Di	50	[9/1/86 - *NOW*]

Example 4 Consider the query from Example 2, i.e. "Who are the managers of the departments for whom Ed has worked." It is expressed in TSQL2 as

```
SELECT  SNAPSHOT E1.Name
FROM    EMP AS E1 E2; DEPT AS D
WHERE   E2.Name = "Ed" AND E2.DeptName = D.Name AND
        D.MgrName = E1.Name AND E2 overlaps D
```

In this query, variable E2 corresponds to Ed's record, and variable D to the department in which Ed worked at some time (condition "E2.DeptName = D.Name AND E2 overlaps D" assures this). Moreover, the SNAPSHOT operator returns only the values of the application-specific attribute(s) (without the corresponding times).

The answer to this query is { Ed, Di }.

The language TSQL2 is upwardly compatible with SQL-92([Snodgrass et al., 1994b]. In addition to the regular SQL features, it has extra constructs that are included in the language in order to explicitly support time, such as temporal operators "overlaps" and "contains" (in queries from Examples 3 and 4) and non-temporal operators, such as SNAPSHOT (in the query from Example 4).

5.3 Comparison Between Temporally Grouped and Ungrouped Queries

In Sections 5.1 and 5.2, we expressed the same queries in the temporally grouped and the temporally ungrouped models. This raises the question about the relationship between temporally grouped and ungrouped query languages. It was formally shown in [Clifford et al., 1994] that temporally grouped models with their query languages are more expressive than corresponding temporally ungrouped counterparts. In this section, we will illustrate the differences between temporally grouped and ungrouped query languages with some more problematic examples. We will first consider purely temporally ungrouped models, and then in Section 5.4 we will discuss how the addition of surrogates to the model (as in TSQL2) solves some, but not all, of these problems, and in any case place a heavy burden on the user to formulate the queries properly.

Example 5 Consider the query

Find salary histories of people when they worked in the Toy department.

It can be expressed in SQL_h as

SELECT EMP.Salary : Time
FROM EMP : Time
WHERE EMP.DeptName.Time = "Toy"

This query returns the following answer when evaluated against relation **EMP** from Figure 2(a).

Salary (K)		
[2/1/82 - 5/31/82]	$\rightarrow$	20
[6/1/82 - 1/31/85]	$\rightarrow$	30
[2/1/85 - 1/31/87]	$\rightarrow$	40
[1/1/82 - 7/31/84]	$\rightarrow$	30
[8/1/84 - 8/31/86]	$\rightarrow$	40
[9/1/86 - *NOW*]	$\rightarrow$	50

If this query is expressed in TSQL2 in an "obvious" way as

```
SELECT  EMP.Salary
FROM    EMP
WHERE   EMP.DeptName = "Toy"
```

then the answer is

Salary	VALID
20	[2/1/82 - 5/31/82]
30	[6/1/82 - 1/31/85]
40	[2/1/85 - 1/31/87]
30	[1/1/82 - 7/31/84]
40	[8/1/84 - 8/31/86]
50	[9/1/86 - *NOW*]

Clearly, this answer does not make sense: it does not adequately represent salary *histories* of employees since information on how to *group* salary figures into salary histories is lost. For example, the second tuple in the answer corresponds to $\mathcal{ED}$'s salary and the fourth tuple to $\mathcal{DI}$'s salary. However, we cannot make this distinction from the answer.

One can argue that if you want to make this distinction in the temporally ungrouped model, then you should retrieve person's name in addition to the salary information. This argument does not hold for two reasons. First, in some situations the user does not want to show employees names because of the confidentiality of the salary information. Second, as will be shown below, the problem still exists even if the name is shown, since the name can change over time.

The problems with the answer to this TSQL2 query become even worse if we *coalesce* the resulting relation[3] as is typically done in the temporally ungrouped models. If we coalesce the answer above, then we obtain the relation

Salary	VALID
20	[2/1/82 - 5/31/82]
30	[1/1/82 - 1/31/85]
40	[8/1/84 - 1/31/87]
50	[9/1/86 - *NOW*]

Clearly, this relation makes no sense at all as the answer to the query presented above.

[3]Coalescing is merging one or several tuples with the same values of non-temporal attributes, whose lifespans intersect, into a single tuple whose lifespan is the union of the lifespans of the merged tuples. It is discussed in [McKenzie and Snodgrass, 1991] in the context of merging what are called *value-equivalent tuples*.

The next example illustrates additional problems with asking temporal questions in temporally ungrouped models.

Example 6 Assume we want to find the times when $\mathcal{ED}$ was working in the Book department. We çan express this request in a kind of "pseudo-SQL" as

```
SELECT Time
FROM Emp
WHERE information is about ℰ𝒟 and Department is ''Book''
```

When we considered updates to the information in the database, it was reasonable to require the updater to know something about the data in the database to assure that the update is performed correctly. However, someone querying can reasonably be presumed to know little (or at least less) about the database, and is in fact posing a query to learn more. So, it seems reasonable to demand of a temporal database model that it requires as little as possible of the queryer. For example, expecting the queryer to know at least something about $\mathcal{ED}$ — like his name at some point in time — in order to learn more about him seems reasonable, while expecting the queryer to know $\mathcal{ED}$'s name at every point in time does not. For example, assume that the queryer knows $\mathcal{ED}$ by the name "Ed." Then the previous query can be expressed in SQL_h as

```
SELECT   Time1
FROM     EMP : Time1, Time2
WHERE    EMP.DeptName.Time1 = "Book" AND EMP.Name.Time2 = "Ed"
```

and returns the following answer

Time
[4/1/87 - *NOW*]

Note that this query finds the lifespans of the (temporally grouped) SQL_h tuples for which $\mathcal{ED}$ was known as "Ed" at some point in time. This query returns the correct answer because of the grouping mechanism of SQL_h.

In contrast to this, the following TSQL2 query

```
SELECT   VALID(E)
FROM     EMP(Name, DeptName) AS E
WHERE    E.DeptName = "Book" AND E.Name = "Ed"
```

returns the answer {[4/1/87 - 12/31/87]} which is incorrect. This TSQL2 query returns incorrect answer because it cannot access the *whole* employment history of $\mathcal{ED}$, but rather only those records that correspond to $\mathcal{ED}$ when his name was "Ed." Furthermore, since TSQL2 without surrogates does not support grouping, it is impossible to retrieve the correct employment history of $\mathcal{ED}$, unless the queryer knows *all* the names that $\mathcal{ED}$ had throughout his employment history, which may be an unreasonable requirement imposed on the user by the DBMS developer.

To illustrate still another problem with temporally ungrouped queries, assume that $\mathcal{ED}$'s employment history was incorrectly specified in Figure 1, and assume that his salary was 35K rather than 40K since January 1, 1989 (while he was called Edward). This means that the record (Edward, 40, M, 7/1/55, Book, [1/1/88 – NOW]) in Figure 1 should be replaced with two records

(Edward, 40, M, 7/1/55, Book, [1/1/88 – 12/31/88])

(Edward, 35, M, 7/1/55, Book, [1/1/89 – NOW])

Then consider the query

Example 7 "How many employees had salaries which were always rising (i.e. never decreased over time)?"

This query can be expressed in SQL_h as

```
SELECT  COUNT (NAME)
FROM    EMP
WHERE   NAME NOT IN
        (SELECT EMP.NAME : TIME1
        FROM EMP : TIME1, TIME2
        WHERE EMP.SAL.TIME1 > EMP.SAL.TIME2 AND TIME1 < TIME2)
```

The answer to this query is 1 (for Di). Note that Ed is not counted in the final answer because he received a cut in his salary on January 1, 1989.

If we express this query in TSQL2 as

```
SELECT  COUNT UNIQUE (NAME)
FROM    EMP
WHERE   NAME NOT IN
        (SELECT SNAPSHOT E1.NAME
        FROM EMP AS E1 E2
        WHERE E1.Salary > E2.Salary AND
          E1.Name = E2.Name AND E1 precedes E2)
```

then the answer to this query is 2 (Ed and Di were counted once, and Edward not at all), which is incorrect because $\mathcal{ED}$'s salary decreased at some point (on January 1, 1989).

Note that the TSQL2 query returns incorrect answer for the same reason as before: it cannot properly identify all the records belonging to the same logical group of records ($\mathcal{ED}$ in our case).

These examples show the importance of grouping since they demonstrate that temporally ungrouped query languages cannot simulate grouping. Nevertheless, the temporally ungrouped data model has its own advantages, including the fact that most of commercial databases at present are based on the temporally ungrouped model. Therefore, it is also important to add grouping mechanisms to the temporally ungrouped data model and incorporate these mechanisms into temporally ungrouped query languages. We consider such mechanisms in the next section.

5.4 Querying Temporally Ungrouped Models with Surrogates

As we explained in Section 3, one way to support grouping in a temporally ungrouped model is to add a *surrogate* field to the structure of a record. For example in Figure 3, we added the field ID to the schema of the EMP relation to uniquely identify each person in that relation.

It was formally shown in [Clifford et al., 1994] that a canonical temporally ungrouped query language on databases with surrogates have the same expressive power as L_h, the formal specification of the language SQL_h we have used here. This means that surrogates simulate grouping at the data modeling and data querying levels. To illustrate these concepts, we will show in this section how TSQL2 queries on temporally ungrouped relations with surrogates attempt to simulate SQL_h queries.

Example 8 Consider the query from Example 5 "Find salary histories of people when they worked in the Toy department." The obvious expression of this query in a temporally ungrouped model with surrogates would be

```
SELECT     EMP.ID, EMP.Salary
FROM       EMP
WHERE      EMP.DeptName = "Toy"
ORDER-BY   EMP.ID
```

In this case, the query retrieves group ID's in addition to salaries, and the answer to the query is:

ID	Salary	VALID
100	20	[2/1/82 - 5/31/82]
100	30	[6/1/82 - 1/31/85]
100	40	[2/1/85 - 1/31/87]
101	30	[1/1/82 - 7/31/84]
101	40	[8/1/84 - 8/31/86]
101	50	[9/1/86 - *NOW*]

Note that this query makes explicit reference to the surrogate EMP.ID in the SELECT and ORDER-BY clauses, and it is the responsibility of the user to do this. Moreover, the ordering of the tuples by the surrogate EMP.ID is crucial for the user to make any sense of the list of the tuples in the answer. In fact, ordering the tuples by the surrogate ID only partially provides the temporal grouping information. The user is still required to search the answer table, which can often be quite large, and find those places where the surrogate values change, in order to form the groups. Thus, this solution only approximates the solution provided by inherently temporally grouped models, and imposes more of a burden on the user.

Example 9 Consider the query from Example 6, i.e. "Find the times when $\mathcal{ED}$ was working in the Book department." Assume that the queryer knows that $\mathcal{ED}$ was known as "Ed" at some point in time. This query can be expressed correctly in TSQL2 with surrogates as

```
SELECT  VALID(E2)
FROM    (SELECT ID, Name
        FROM EMP
        WHERE DeptName = "Book") AS E1 E2
WHERE   E1.Name = "Ed" AND E1.ID = E2.ID
```

It would return the answer {[4/1/87 - 12/31/87], [1/1/88 - *NOW*]} (or after coalescing, {[4/1/87 - *NOW*]}). In other words, this query first selects the tuples corresponding to the "Book" department and calls the resulting relation E1 or E2. Then it retrieves the tuples corresponding to "Ed" and takes all the tuples in the relation E2 that have the same group IDs (i.e. belong to the same group) as the tuples corresponding to Ed. Clearly, this TSQL2 query with group IDs simulates the SQL_h query from Example 6. It can simulate this SQL_h query because surrogates simulate the grouping mechanism of SQL_h.

Example 10 Consider the query from Example 7 "How many employees had salaries which were always rising (i.e. never decreased over time)?" The obvious expression of this query in a temporally ungrouped model with surrogates would be

```
SELECT   COUNT UNIQUE (ID)
FROM     EMP
WHERE    ID NOT IN
         (SELECT SNAPSHOT E1.ID
         FROM EMP AS E1, EMP AS E2
         WHERE E1.Salary > E2.Salary E1.ID = E2.ID AND E1 precedes E2)
```

and returns the correct answer 1.

These three examples illustrate potential solutions to the queries discussed in Section 5.3 within a hypothetical temporally ungrouped model with surrogates to simulate temporal grouping. In fact, the TSQL2 proposal has some additional problems with these queries besides the ones already pointed out in Example 8. First of all, TSQL2 does not allow surrogates in the outermost SELECT clause because of their "special semantics." In particular, [Snodgrass et al., 1994] says that

> Surrogates are unique identifiers that can be compared for equality, but *the values of which cannot be seen by the users.*

Thus, it appears to be impossible to properly express the query from Example 8 in TSQL2 even with its attempt to simulate grouping with the surrogate mechanism. In addition, it is not clear whether the query from Example 10 is valid in TSQL2 because the interaction between aggregates and surrogates is not discussed in the TSQL2 proposal [Snodgrass et al., 1994b].

In summary, we have shown in this section that querying temporally ungrouped historical models can be problematic: the answers that some queries return may not correspond to what the user has in mind, and there is no way to obtain the information that the user wants to obtain, given his or her knowledge of the data. We have also shown how the query languages for the temporally grouped models solve the problems encountered by the temporally ungrouped models. We pointed out that there are problems related to the proper semantics of surrogates which remain to be resolved before it is clear that the TSQL2 language proposal is capable of representing time series data properly. Finally, even if the problems with surrogates were worked out, we showed that many queries in temporally grouped models are simpler to express than their counterpart in a temporally ungrouped model with surrogates, because such a model places the burden of the management of the surrogates at the user or conceptual level, instead of building this management into the model itself, and leaving the management of surrogates and such things to the implementation level.

6 Conclusions

The temporal database models in the literature have been effectively characterized in [Clifford et al., 1994] as either temporally ungrouped or temporally grouped. Although for a long time it was thought that these two approaches were equivalent, [Clifford et al., 1994] proved that they were not and that the temporally grouped approach was more expressive.

In this paper we discussed the effort in the temporal database community towards the development of a temporal SQL standard, the so-called language TSQL2. This effort is an attempt to consolidate all of the many proposals in the literature for query languages based upon temporally ungrouped models. This is an important effort supported by NSF and DARPA that has also attracted the attention of some of the database vendors. It should help to move some of the theoretical results from the temporal database research community into the realm of commercial database systems. While we support this effort as an immediate and practical solution to providing better temporal support within the context of the SQL-92 standard, we believe that the temporally grouped approach better models the temporal nature of data.

Therefore, in this paper we presented a detailed comparison between the grouped and ungrouped approaches, focusing on the process of updating and querying the temporal database, since that is what is of most interest to the end user. We argued that the grouped model is better suited than the ungrouped model for querying and updating temporal data. Intuitively, this is the case because the grouped model represents time varying attributes as functions of time, i.e., as time series, and provides for the direct manipulation of these objects that naturally arise when modeling temporal phenomena. In contrast to this, the ungrouped model has to simulate this functionality, and, as we argue in the paper, cannot do so adequately or naturally in a number of situations.

In this paper, we discussed further the approach introduced in [Clifford et al., 1994] to extend temporally ungrouped models with a carefully controlled system surrogate to "simulate" grouping. While we showed in [Clifford et al., 1994] that this technique is formally equivalent to the grouped approach, in this paper we argued that as a conceptual level model, such as TSQL2, this approach seems to be less natural and less convenient than directly modeling time-varying attribute values as first-class objects, as the temporally grouped approach does.

In the course of this discussion we pointed out that the TSQL2 language design is a compromise, constrained by the need to be upwardly compatible with the existing relational model of SQL-92. Thus, its designers felt the need to remain in the realm of first normal form relations, and to make as little changes as possible to the standard SQL view of data and of querying. We note that although the TSQL2 proposal currently allows for a `SURROGATE` data type, it still has some problems with supporting this data type. Moreover, as we observed in the paper, TSQL2 delegates the tasks of creation, deletion, and retrieval of grouped temporal tuples to the end-user, whereas a temporally grouped model has this functionality built in. We hope that this paper, by demonstrating the advantages of temporal grouping and some of the problems with incorporating surrogates at the conceptual level, might contribute to a further refinement of the surrogate notion in TSQL2.

Finally, we believe that the effort to influence the design of the SQL3 standard, in order to have a truly satisfactory treatment of temporal data, ought to be focused on ensuring that SQL3 incorporate temporal grouping as a fundamental construct. The proponents of the object-oriented approach to data modeling have argued that in many applications information is complex in structure and cannot be easily represented in the classical relational model. The preliminary design of SQL3 ([Committee, 1993]) incorporates this philosophy by allowing the representation of complex objects with object identifiers. We likewise believe that the evidence presented in this paper indicates that the effective modeling of temporal data requires direct modeling of the complex nature of temporal information, that is, the modeling of temporal grouping at the conceptual level.

References

[Ariav, 1986] Ariav, G. (1986). A temporally oriented data model. *ACM Transactions on Database Systems*, 11(4):499–527.

[Ben-Zvi, 1982] Ben-Zvi, J. (1982). *The Time Relational Model.* PhD thesis, University of California at Los Angeles.

[Bolour et al., 1982] Bolour, A., Anderson, T. L., Dekeyser, L. J., and Wong, H. K. T. (1982). The role of time in information processing: A survey. *SigArt Newsletter*, 80:28–48.

[Chalfin, 1994] Chalfin, M. (1994). The past is now. *Database Programming & Design*, pages 32–41.

[Clifford, 1982] Clifford, J. (1982). A model for historical databases. In *Proceedings of Workshop on Logical Bases for Data Bases*, Toulouse, France.

[Clifford, 1992] Clifford, J. (1992). Indexical databases. In *Proceedings of Workshop on Current Issues in Database Systems*, Newark, N.J. Rutgers University. (published as *Lecture Notes in Computer Science 759*, Springer-Verlag, 1993).

[Clifford and Croker, 1987] Clifford, J. and Croker, A. (1987). The historical relational data model HRDM and algebra based on lifespans. In *Proc. Third IEEE International Conference on Data Engineering*, pages 528–537, Los Angeles, CA.

[Clifford and Croker, 1988] Clifford, J. and Croker, A. (1988). Objects in time. *IEEE Data Engineering*, 7(4):189–196.

[Clifford et al., 1995] Clifford, J., Croker, A., Grandi, F., and Tuzhilin, A. (1995). On temporal grouping. In Clifford, J. and Tuzhilin, A., editors, *Recent Advances in Temporal Databases*, Springer-Verlag.

[Clifford et al., 1994] Clifford, J., Croker, A., and Tuzhilin, A. (1994). On completeness of historical relational query languages. *ACM Transactions on Database Systems*, 19(2):64–116.

[Clifford and Tansel, 1985] Clifford, J. and Tansel, A. (1985). On an algebra for historical relational databases: Two views. In Navathe, S., editor, *Proceedings of ACM SIGMOD Conference*, pages 247–265, Austin, TX. ACM.

[Clifford and Warren, 1983] Clifford, J. and Warren, D. S. (1983). Formal semantics for time in databases. *ACM Transactions on Database Systems*, 8(2):214–254.

[Codd, 1979] Codd, E. F. (1979). Extending the data base relational model to capture more meaning. *ACM Transactions on Database Systems*, 4(4), pp. 397–434.

[Committee, 1993] Committee, S. L. D. (1993). ISO-ANSI working draft database language SQL (SQL3). work in progress, available via anonymous ftp from `gatekeeper.dec.com`.

[Elmasri and Navathe, 1994] Elmasri, R. and Navathe, S. (1994). *Fundamentals of Database Systems*. Benjamin-Cummings. 2nd Edition.

[Gadia, 1992] Gadia, S. (1992). A seamless generic extension of SQL for querying temporal data. Technical report, Iowa State University.

[Gadia, 1993] Gadia, S. (1993). *Ben-Zvi's Pioneering Work in Relational Temporal Databases*, chapter 8, pages 202–207. In [Tansel et al., 1993].

[Gadia and Nair, 1993] Gadia, S. and Nair, S. (1993). *Temporal Databases: A Prelude to Parametric Data*, chapter 2, pages 28–66. In [Tansel et al., 1993].

[Gadia, 1988a] Gadia, S. K. (1988a). A homogeneous relational model and query languages for temporal databases. *ACM Transactions on Database Systems*, 13(4):418–448.

[Gadia, 1988b] Gadia, S. K. (1988b). A homogeneous relational model and query languages for temporal databases. *ACM Transactions on Database Systems*, 13(4):418–448.

[Grandi and Scalas, 1991] Grandi, F. and Scalas, M. (1991). A history-oriented temporal query language. In *Proceedings of the 5th IEEE Compeuro*, Bologna, Italy.

[Jaeschke and Schek, 1982] Jaeschke, G. and Schek, H. (1982). Remarks on the algebra of non-first normal form relations. In *Proceedings of the ACM Symposium on Principles of Database Systems*. ACM, New York, pp. 124–138.

[Jensen et al., 1994] Jensen, C. S., Clifford, J., Elmasri, R., Gadia, S. K., Hayes, P., and Jajodia [eds], S. (1994). A glossary of temporal database concepts. *SIGMOD Record*, 23(1):52–64.

[Jensen et al., 1992] Jensen, C. S., Clifford, J., Gadia, S. K., Segev, A., and Snodgrass, R. T. (1992). A glossary of temporal database concepts. *SIGMOD Record*, 21(3):35–43.

[Jensen et al., 1989] Jensen, C. S., Mark, L., and Roussopoulos, N. (1989). Incremental implementation model for relational databases with transaction time. Technical Report UMIACS-TR-8963/CS-TR-2275, University of Maryland, College Park, MD.

[Jensen (ed.), 1993] Jensen (ed.), C. S. (1993). A consensus test suite of temporal database queries. Technical Report R 93-2034, Department of Mathematics and Computer, Institute for Electronic Systems, Denmark.

[Jones and Mason, 1980] Jones, S. and Mason, P. (1980). Handling the time dimension in a data base. In *Proc. International Conference on Data Bases*, pages 65–83, Heyden. British Computer Society.

[Kline, 1993] Kline, N. (1993). Temporal database bibliography. TempIS Technical Report 53, Department of Computer Science, Tucson, AZ.

[Lomet and Salzberg, 1992] Lomet, D. and Salzberg, B. (1992). Rollback databases. Technical Report NU-CCS-92-3, Northeastern University.

[Lorentzos, 1993] Lorentzos, N. (1993). *The Interval-extended Relational Model and Its Application to Valid-time Databases*, chapter 3, pages 67–91. In [Tansel et al., 1993]

[Lorentzos, 1987] Lorentzos, N.A.; Johnson, R. (1987). TRA: A model for a temporal relational algebra. In *Proceedings of the Conference on Temporal Aspects in Information Systems*, pages 99–112, France. AFCET.

[McFadden and Hoffer, 1994] McFadden, F. and Hoffer, J. (1994). *Modern Database Management.* Benjamin-Cummings. 4th Edition.

[McKenzie, 1986] McKenzie, E. (1986). Bibliography: Temporal databases. *ACM SIGMOD Record*, 15(4):40–52.

[McKenzie and Snodgrass, 1991] McKenzie, E. and Snodgrass, R. (1991). Supporting valid time in an historical relational algebra: Proofs and extensions. Technical Report TR–91–15, Department of Computer Science, University of Arizona, Tucson, AZ.

[Navathe and Ahmed, 1989] Navathe, S. B. and Ahmed, R. (1989). A temporal relational model and a query language. *Information Sciences*, 49:147–175.

[Pissinou et al., 1994] Pissinou, N., Snodgrass, R., Almasri, R., Mumick, I., Özsu, M. T., Pernici, B., Segev, A., Theodoulidis, B., and Dayal, U. (1994). Towards an infrastructure for temporal databases: Report of an invitational ARPA/NSF workshop. *ACM SIGMOD Record*, 23(1):35–51.

[Roth et al., 1988] Roth, M. A., Korth, H., and Silberschatz, A. (1988). Extended algebra and calculus for nested relational databases. *ACM Transactions on Database Systems*, 13(4):388–417.

[Sarda, 1990] Sarda, N. (1990). Algebra and query language for a historical data model. *The Computer Journal*, 33(1):11–18.

[Sarda, 1993] Sarda, N. (1993). *HSQL: A Historical Query Language*, chapter 5, pages 110–140. In [Tansel et al., 1993].

[Snodgrass, 1995] R. T. Snodgrass, editor (1995). *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers.

[Snodgrass et al., 1994a] Snodgrass, R.T., Ahn, I., Ariav, G., Batory, D.S., Clifford, J., Dyreson, C.E., Elmasri, R., Grandi, F., Jensen, C.S., Kafer, W., Kline, N., Kulkanri, K., Leung, T.Y.C., Lorentzos, N., Roddick, J.F., Segev, A., Soo, M.D. and Sripada, S.M. (1994). A TSQL2 Tutorial. *SIGMOD Record*, 23(3):27–34.

[Snodgrass et al., 1994b] Snodgrass, R.T., Ahn, I., Ariav, G., Batory, D.S., Clifford, J., Dyreson, C.E., Elmasri, R., Grandi, F., Jensen, C.S., Kafer, W., Kline, N., Kulkanri, K., Leung, T.Y.C., Lorentzos, N., Roddick, J.F., Segev, A., Soo, M.D. and Sripada, S.M. (1994). TSQL2 Language Specification. *SIGMOD Record*, 23(1):65–86.

[Snodgrass et al., 1994] Snodgrass, R. T., et al. (1994). An evaluation of TSQL2. Commentary, TSQL2 Design Committee.

[Snodgrass, 1987] Snodgrass, R. T. (1987). The temporal query language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298.

[Soo, 1991] Soo, M. (1991). Bibliography on temporal databases. *ACM SIGMOD Record*, 20(1):14–23.

[Stam and Snodgrass, 1988] Stam, R. and Snodgrass, R. (1988). A bibliography on temporal databases. *Database Engineering*, 7(4):231–239.

[Tansel, 1986] Tansel, A. (1986). Adding time dimension to relational model and extending relational algebra. *Information Systems*, 11(4):343–355.

[Tansel, 1993] Tansel, A. (1993). *A Generalized Relational Framework for Modeling Temporal Data*, chapter 7, pages 183–201. In [Tansel et al., 1993]

[Tansel et al., 1993] Tansel, A., Clifford, J., Gadia, S., Jajodia, S., Segev, A., and Snodgrass (eds.), R. (1993). *Temporal Databases: Theory, Design, and Implementation.* Database Systems and Applications Series. Benjamin/Cummings, Redwood City, CA.

[Tansel and Garnett, 1992] Tansel, A. and Garnett, L. (1992). On Roth. Korth, and Silberschatz's extended algebra and calculus for nested relational databases. *ACM Transactions on Database Systems*, 17(2):374–383.

Appendix: An Example Enterprise

In this Appendix, we present an example of a fragment of an enterprise model as specified in [Snodgrass et al., 1994]. In this application, an organization models information about its employees and the departments in which they work. The conceptual model for this application, represented as an E-R diagram, is taken from [Snodgrass et al., 1994] and is shown in Figure ??. (Note that the underlined attributes represent the *entity keys*, and that all attributes are considered to be time-varying).

There are three entity sets:

1. The entity set **Emp** models the history of those employees which are of interest to the organization. In particular, it models the history of their `Name`, `Salary`, `Gender` and date of birth (`D-birth`). While the name and salary of an employee vary over time, both the gender and the date of birth are assumed to be time-invariant. The `Gender` attribute of `Emp` is one of `''F''` (female) and `''M''` (male).

2. The entity set **Dept** models the history of Departments within the organization. In particular, it models the history of their `Name` and their `Budget`. While the budget of a department varies over time, the name is assumed to be time-invariant[4].

[4]We think that except in very rare cases it is unreasonable in a temporal database to make the assumption that any attribute will be time-invariant. However, this assumption is made for the Department `Name` in [Snodgrass et al., 1994], so we will keep that assumption here.

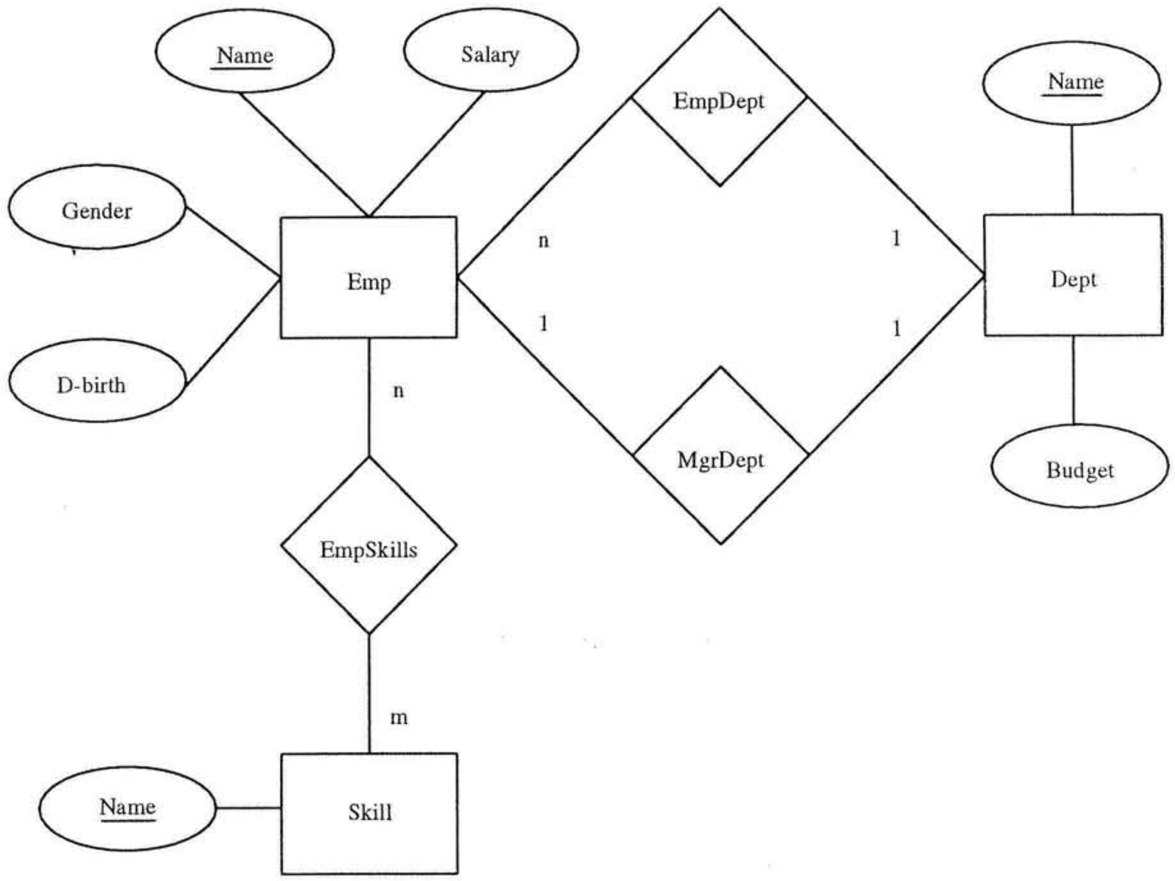

Figure 4: ER Diagram of Enterprise.

3. The entity set **Skill** models the history of a set of skills which are of interest to the organization. They are represented by a single attribute, `Name`, which records the names of individual skills. The name of a skill is time-invariant.

In addition to these entity sets, there are three relationships among them which are to be modeled for the enterprise:

1. The relationship set **EmpSkill** models the history of a many-to-many relationship set between Employees and Skills, specifically, the relationship of *having* a given skill. The skills of an employee may vary over time. For example, employees are considered to have the skill "driving" only during those interval(s) when they hold valid licenses.
2. The relationship set **EmpDept** models the history of the time-varying many-to-one relationship between Employees and Departments, specifically, the relationship of *working in* a given department.
3. The relationship set **MgrDept** models the history of the time-varying 1–1 relationship between Employees and Departments, specifically, the relationship of *managing* a given department (on a given date, one department has one manager).

Note that none of these relationships is represented explicitly as a relation in the body of the paper. Relationship (1) is omitted for space purposes; relationships (2) and (3), being N:1 and

1:1, respectively, are represented in the standard way by migrating the foreign key into the **EMp** relation. Note that this raises the issue of *referential integrity* which, while orthogonal to the temporally grouped/temporally ungrouped distinction discussed in the paper, is nonetheless an interesting question in the realm of temporal databases. A discussion of some of the important aspects of this issue appears in [Clifford and Croker, 1988].