Formal Semantics for Time in Databases

JAMES CLIFFORD and DAVID S. WARREN
State University of New York at Stony Brook

The concept of a historical database is introduced as a tool for modeling the dynamic nature of some part of the real world. Just as first-order logic has been shown to be a useful formalism for expressing and understanding the underlying semantics of the relational database model, intensional logic is presented as an analogous formalism for expressing and understanding the temporal semantics involved in a historical database. The various components of the relational model, as extended to include historical relations, are discussed in terms of the model theory for the logic IL_s, a variation of the logic IL formulated by Richard Montague. The modal concepts of intensional and extensional data constraints and queries are introduced and contrasted. Finally, the potential application of these ideas to the problem of natural language database querying is discussed.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design—*data models*

General Terms: Theory, Design, Language

Additional Key Words and Phrases: Relational database, entity-relationship model, intensional logic, historical databases, temporal semantics

1. INTRODUCTION

The relational database model proposed in [13] views a database as a collection of "time-varying relations of assorted degrees" [8]. However, the model itself incorporates neither the concept of time nor any theory of temporal semantics. This paper suggests that the concept of time can be of interest in real-world databases and presents a technique for incorporating a semantics of time into a database model. The relational model is used as the formal database framework within which the work is cast, but it is not an essential ingredient in the work discussed.

A great deal of attention has been given lately to the role that formal logic can play in providing a formal mathematical theory to unify the theory and semantics of database concepts and operations (cf. [17]). We believe that this is a healthy trend that can only serve to clarify and make precise otherwise vague ideas and theories. Moreover, a great deal of the metatheory of formal logic can be applied directly to the understanding and the proof of many notions in database theory.

This material is based upon work supported by the National Science Foundation under Grant IST-8010834,

Authors' addresses: David S. Warren, Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794. James Clifford, Graduate School of Business Administration, New York University, 90 Trinity Place, New York, NY 10006.

In this paper we propose the concept of a historical database as a tool for modeling the changing states of information about some part of the real world. Most conventional databases are *static*, representing a snapshot view of the world at a given moment in time; changes in the real world generally are reflected in the database by changes to its data, thereby "forgetting," as it were, the old data. By contrast, a historical database is a model of the dynamically changing real world. Changes in the real world are reflected in such a database by establishing a new state description; no data are ever "forgotten." As such, the historical database can be viewed intuitively as a collection of static databases organized in a coherent fashion. This paper provides a detailed description of such an organization and a discussion of the usefulness of the historical database concept for modeling the real world (or some "possible world") more closely than is possible with a static database. A good overview of the issues involved in incorporating a temporal dimension in databases is provided in [5].

We believe that providing a formal semantics for a database model is of paramount importance to its usefulness. The concept of time is crucial to all databases, but is only treated implicitly in the existing database models. Databases exist in time and model changes that occur temporally in the world via database state changes. In order to have a proper understanding of how an explicit representation of time interacts with all of the data in the database, it is not enough simply to allow users to utilize "time attributes" where they seem appropriate. By incorporating a general temporal semantics directly within the database model, not only do we spare the user the task of defining such a semantics, but we also can ensure that time is treated in a uniform and consistent manner. Moreover, if the temporal semantics is built into the model, implementations of a historical database can take advantage of this standard semantics to increase the efficiency of database operations. The basis which we suggest for the semantics of a historical database model is the formulation of an intensional logic IL_s, a modification of the language IL of Richard Montague [35], whose work has profoundly influenced current research in linguistics and the philosophy of language.

The major reason for preferring a Montague-type logic over other formulations of temporal or intensional logics (as in [42]) is the framework he provides [35] for defining a formal syntax and semantics of English using IL. The development of the historical database model is part of our research into the larger area of natural language database querying (NLQ). Our approach is motivated by the desire to develop a framework for NLQ that is founded squarely on a fully formalized syntax and semantics in the sense of Montague [35]. In [11] we discuss the translation of English database queries into the logic IL_s, and provide a general schema for defining an English query language specific to a given database domain. In this paper we show how the model theory of the logic IL_s influences our view of the objects in the historical database. In particular, database attributes are viewed in our historical database model as functions from moments in time to values (in the appropriate domain), and IL_s gives us the power to speak directly about these "higher order" objects and to incorporate them into a general temporal semantics for the database. We can therefore express both static and dynamic constraints (as discussed in [38]) in the same language, by quantifying over variables of the appropriate types.

It should perhaps be noted that a historical database, as we define it, is a theoretical object, and a rather large one at that; no remarks in this paper should be construed as referring to any techniques for implementing this object. Obviously a direct implementation would be prohibitively costly for any real database. Reasonable implementations that eliminate much of the inherent data redundancy of the formal model are not difficult to imagine. We are currently in the process of developing a number of different implementations and algorithms for a historical database (HDB).

After a brief introduction to our notation in Section 2 and a discussion in Section 3 of the motivation behind the historical database concept, we provide in Section 4 a stepwise development of a historical relational database for a simplistic database consisting of a single "entity" relation. Section 5 introduces the intensional logic and model theory that we use to describe the semantics of this model. In Section 6 we discuss in detail the relationship between a historical database and its representation in a model for intensional logic. We adopt the entity-relationship view of data semantics, modified slightly to incorporate a semantics for time; as a working example in this section we use a historical version of the entity-relationship department-store database described by Chang [8], of which the example in Section 4 was a part. Finally in Section 7 we discuss a variety of issues that this research raises in the area of database semantics.

2. DEFINITIONS AND NOTATION

This section introduces some of the standard definitions from the relational database model (mostly from [29]), along with a few remarks about our notation.

A *relation scheme* $R = \langle A, K \rangle$ is an ordered pair consisting of a finite set of attributes $A = \{A_1, A_2, \ldots, A_n\}$ and a finite set of key attributes $K = \{K_1, K_2, \ldots, K_m\}$, where $K \subseteq A$. To say that $K = \{K_1, K_2, \ldots, K_m\}$ is a *key* of scheme R is to say that any valid relation r on R has the property that for any distinct tuples t_1 and t_2 in r, $t_1(K) \neq t_2(K)$, and no proper subset of K has this property. We generally underline the key attributes and write such a relation scheme as $R(\underline{A_1 A_2 \cdots A_m}, A_{m+1} \cdots A_n)$; in this case, it is to be understood that $A = \{A_1, \ldots, A_n\}$ and $K = \{A_1, \ldots, A_m\}$. We will occasionally refer to such an R as an *n-ary relation scheme.* The attributes $A_{m+1}, \ldots, A_n$ are referred to as *role attributes.*

The values for the attributes come from a set D of *domains*, $D = \{D_1, D_2, \ldots, D_l\}$, each D_i being any nonempty set. We let UD denote the union of these domains, that is, $UD = D_1 \cup D_2 \cup \ldots \cup D_l$.

In order to relate the attributes with their domain, we assume that U is the set of all the attributes in the database, and that there is a function DOM: $U \rightarrow D$ which maps each attribute onto its corresponding domain, that is, DOM(A_i) is the domain of the attribute A_i.

Finally, we say that a *relation* r on relation scheme $R = \langle A, K \rangle$ is a finite set of mappings $\{t_1, t_2, \ldots, t_n\}$, where each t_i is a function from A to UD such that $t_i(A_j) \in$ DOM(A_j) for all $t_i \in r$ and all $A_j \in A$, and for any distinct tuples t_i and t_j in r, $t_1(K) \neq t_2(K)$.

For a relation r on $R = \langle A, K \rangle$, if $X \subseteq A$ and $t \in r$, by $t(X)$ we mean the restriction of t to X. We sometimes use the notation $t(R)$ to mean $t(A)$, that is, we use the name of the scheme, R, to stand for the set, A, of all of its attributes.

If r is a relation on scheme $R = \langle A, K \rangle$, $A_i \in A$ and $a \in \text{DOM}(A_i)$, the usual relational operations are defined. *Select A_i equal to a in relation r*, written $\sigma_{A_i=a}(r)$, is the relation r' on scheme R, such that

$$r' = \{ t : t \in r \text{ and } t(A_i) = a \},$$

that is, that subset of the tuples of r which have the value a for the attribute A_i. If $X \subseteq A$, the *projection of r onto X*, written $\Pi_X(r)$, is the relation r' on X, such that

$$r' = \{t(X) : t \in r\},$$

obtained by deleting all the columns corresponding to the attributes in $A - X$, and then removing any duplicate tuples that remain.

An *entity relation* is a relation r on a scheme R of the form $(\underline{K_1}\ A_1 \cdots A_n)$, where K_1 is the key and any k-value for K_1 uniquely determines the values for each of the other attributes. (This essentially means that each entity relation is in Boyce-Codd Normal Form (BCNF); see [48] for a discussion.) Intuitively, a K_1-value k uniquely identifies some entity of interest to the database, and each A_i-value associated with k gives one of the attributes of k. We use the notation t_k to denote the tuple whose key value is k.

A *relationship relation* is a relation r on scheme R of the form $(\underline{K_1 \ldots K_n}\ A_1 \cdots A_m)$, where $\{K_1, \ldots, K_n\}$ is the key and determines the values of the other attributes. Intuitively, a $\langle K_1, \ldots, K_n \rangle$-value $\langle k_1, \ldots, k_n \rangle$ represents an n-ary relationship among the n entities $k_1, \ldots, k_n$, and each A_i-value associated with $\langle k_1, \ldots, k_n \rangle$ gives an attribute of that relationship.

3. MOTIVATION

Consider a static database with the relation scheme EMP_REL(<u>EMP</u> MGR SAL DEPT) and a relation emp_real on EMP_REL. A typical query to such a relation, of the sort that has been treated in the literature, might be, "What is employee John's salary?" In the relational algebra this would be expressed as $\Pi_{\text{SAL}}(\sigma_{\text{EMP=John}}(\text{emp\_rel}))$. A first-order language would express this same query as something like $\{z \mid \exists x\ \exists y\ \text{emp\_rel}(\text{John}, x, y, z)\}$, where x, y, and z are individual variables, and John is an individual constant. In order to answer such a query, a data manipulation language (DML) simply accesses the relation instance emp_rel on EMP_REL, such as the one in Figure 1. In recent database literature (e.g., [8, 33, 41]), such a relation instance has been termed the *extension* of the relation scheme EMP_REL, a term borrowed from logic.

One could imagine other sorts of queries that casual users might want to ask about the employees in this company, for example:

"Has John's salary risen?"
"When was Peter rehired?"
"Did Rachel work for the toy department last year?"
"Has John ever earned the same salary as Peter?"
"Will the average salary in the linen department surpass $30,000 within the next five years?"

Time-dependent questions of this sort are not handled by existing static database models or systems, and have not received adequate attention within the database

EMP_REL

EMP	MGR	DEPT	SAL
John	John	Linen	25K
Mike	John	Linen	17K
Elsie	Elsie	Toy	26K
Liz	Liz	Hardware	30K
Rachel	Liz	Hardware	29K
Peter	Liz	Hardware	29K

Fig. 1. Relation emp_rel.

literature (although the need for a temporal semantics in databases is discussed, for example, in [5, 7, 26, 28, 38, 44]). Real database administrators faced with the need to process particular instances of queries of this sort have undoubtedly used some version of the technique that we present here of incorporating a time attribute into the database and providing this attribute with a special significance. We are interested in developing a unified and formal theory of database semantics that includes time. In other words, given the need for maintaining a historical record of changing data, and a language (English) that makes (explicit or implicit) reference to the concept of time, we would like a theory that provides a database semantics capable of interpreting sentences in the language correctly, that is, in a way that corresponds with our intuitive understanding of the relation of time to the semantics of the real world.

Let us consider more closely the query, "Has John's salary risen?" Even with time represented explicitly in the database, there is no apparent simple relational algebraic formulation for this query. With the first-order representation for John's salary given above, as a first guess we might imagine that RISE ($\{z \mid \exists x\ \exists y$ emp_rel (John, x, y, z)$\}$) would represent this new query, where RISE is a predicate symbol. However, even with an FD that ensured that John had only one salary, say, $25,000 (25K), it clearly makes no sense to ask whether 25K "rises." In order to answer this question, more data are needed than the current extension of John's salary: The values of John's salary for some other point(s) in time (in this instance, in the past) are needed. In the model that we present, we identify such things as SALaries, not with individual dollar amounts, but with dollar amounts in the role of an EMPloyee's salary.

It is not difficult to see that if we need to keep track of when the facts we record in our database are to be considered "true," then we need to "time-stamp" these facts in some way. Exactly how we propose to do this, and how this proposal will extend the concept of and intuition about relations, are the subjects of the remaining sections of this paper. For the moment we take a simplified look at this suggestion and discuss some of the issues involved. A first point to notice is that the expression $\{z \mid \exists x\ \exists y$ emp_rel (John, x, y, z)$\}$ has, in these two queries, two very different meanings. The simple query $\{z \mid \exists x\ \exists y$ emp_rel (John, x, y, z)$\}$ denotes the extensional value 25K, the salary that John is making now. The second query, however, RISE ($\{z \mid \exists x\ \exists y$ emp_rel (John, x, y, z)$\}$), is not to be interpreted as asking RISE (25K). Some other meaning, involving more than the current extension of John's salary, must be given to John's salary in order to determine whether the predicate RISE is true of it. This other meaning for John's salary, we shall see, is what is called (in intensional logic) its *intension*. (The terms "extension" and "intension" are given formal definitions in intensional

$$\begin{array}{ccc}
\begin{bmatrix} 1977 \rightarrow \text{John} \\ 1978 \rightarrow \text{John} \\ 1979 \rightarrow \text{John} \\ 1980 \rightarrow \text{John} \\ 1981 \rightarrow \text{John} \end{bmatrix} &
\begin{bmatrix} 1977 \rightarrow \text{linen} \\ 1978 \rightarrow \text{linen} \\ 1979 \rightarrow \text{linen} \\ 1980 \rightarrow \text{shoe} \\ 1981 \rightarrow \text{shoe} \end{bmatrix} &
\begin{bmatrix} 1977 \rightarrow 25\text{K} \\ 1978 \rightarrow 25\text{K} \\ 1979 \rightarrow 27\text{K} \\ 1980 \rightarrow 27\text{K} \\ 1981 \rightarrow 30\text{K} \end{bmatrix} \\
\text{(a)} & \text{(b)} & \text{(c)}
\end{array}$$

Fig. 2. (a) Intension of "John"; (b) intension of "Department-of-John"; (c) intension of "Salary-of-John."

logic, and will be defined formally here. They should not be confused with their usage in some database papers, for example, [41], in which the term "intension" is used to refer to axioms which constrain the set of possible models for the database.) It is helpful to think of them in terms of roles, which at any moment of time might be filled by any appropriate individual.

The concept of intension dates back to Frege [16] and his distinction between the sense and denotation of an expression in a language. A full discussion of the history of these concepts in logic is beyond the scope of this paper. (Carnap [6] and Dowty [14], among others, provide a useful introduction to these issues.) Roughly speaking, the extension of a linguistic expression is some "object" or element of the appropriate kind in the model for that language. The extension of a name is some individual in the model; the extension of a formula is one of the objects "True" or "False"; the extension of a set is some collection of individuals, and so on. The concept of intension, on the other hand, is meant to capture the notion of the "sense" or "idea" or "meaning" of an expression. This somewhat vague idea is formalized in Montague's IL by defining the intension of any expression as a function from a set of points of reference (variously called "possible worlds" or indices) to extensions. Thus the intension of a name, called an *individual concept* (IC), is a function which, given any index, picks out some individual as the referent of that name at that index. Similarly, the intension of a set, called a *property*, picks out some collection of individuals which is the referent of the set-name at each index, and the intension of a formula, called a *proposition*, is that function which, for any index, tells whether the formula is true or false at that index.

For example, suppose that we are interested in maintaining a yearly record of the emp_rel relation, say for the period of the last five years. If we define a set of times, say, $S = \{1977, 1978, 1979, 1980, 1981\}$, as the complete set of indices or points of reference of interest to us, then the intension of a name in our language will be a function from this set S to individuals in the model. Thus, considering the employee John, we might have the intensions depicted in Figure 2 for the names "John," "Department-of-John," and "Salary-of-John" (assuming for the moment some linguistic mechanism for constructing these names). The function that is the intension of "Department-of-John," for instance, represents the role of John's department and tells what department "fills" that role in each state. We can now imagine a DML that could examine such a database and provide an affirmative answer to our query, "Has John's salary risen?" In the remaining sections we present a formalization of these ideas in terms of the relational database model using the intensional logic IL_s. We also discuss the application of

EMP	MGR	DEPT	SAL
John	John	Linen	23K
Mike	John	Linen	17K
Elsie	Elsie	Toy	26K
Liz	Liz	Hardware	30K
Rachel	Liz	Hardware	29K
Peter	Liz	Hardware	29K

(a)

EMP	MGR	DEPT	SAL
John	John	Linen	25K
Mike	Elsie	Toy	20K
Elsie	Elsie	Toy	27K
Rachel	Rachel	Hardware	28K
Sharon	Rachel	Hardware	25K

(b)

EMP	MGR	DEPT	SAL
Beth	Beth	Linen	23K
Elsie	Elsie	Toy	27K
Rachel	Peter	Hardware	28K
Sharon	Peter	Hardware	25K
Peter	Peter	Hardware	33K

(c)

Fig. 3. (a) emp_rel$_1$; (b) relation emp_rel$_2$; (c) relation emp_rel$_3$.

this logic to database querying in natural language, and to the unified expression of various kinds of data constraints.

4. HISTORICAL DATABASES

We imagine that an enterprise wishes to maintain a historical database, that is, one that models the dynamic nature of that part of the real world that is its concern. To simplify the discussion, we again consider only our entity relation scheme EMP_REL as representing the entire database; in Section 6, we present a more formal view, and include both entity and relationship relations. We suppose that we are given three static relation instances, emp_rel$_1$, emp_rel$_2$, and emp_rel$_3$ (Figure 3), that is, instances which each represent a single state of the world as modeled in the relational database.

We will proceed to develop the concept of a historical database in stages in order to provide some intuition for the more formal treatment given in the next sections. We will use the EMP_REL entity relation scheme as our running example. The first step is to incorporate a method for time-stamping the tuples ("facts") in our database. To do this we add a new attribute, STATE, to the relation scheme, creating the scheme EMP_REL′(STATE EMP MGR DEPT SAL). Each tuple t in an instance emp_rel$_i$ is extended accordingly, by adding the value S_i for the attribute STATE. The extended relations emp_rel$'_i$ are shown in Figure 4. Formally,

$$\text{emp\_rel}'_i = \{t : t(\text{EMP\_REL}) \in \text{emp\_rel}_i \quad \text{and} \quad t(\text{STATE}) = S_i\}.$$

STATE	EMP	MGR	DEPT	SAL
S_1	John	John	Linen	23K
S_1	Mike	John	Linen	17K
S_1	Elsie	Elsie	Toy	26K
S_1	Liz	Liz	Hardware	30K
S_1	Rachel	Liz	Hardware	29K
S_1	Peter	Liz	Hardware	29K

(a)

STATE	EMP	MGR	DEPT	SAL
S_2	John	John	Linen	25K
S_2	Mike	Elsie	Toy	20K
S_2	Elsie	Elsie	Toy	27K
S_2	Rachel	Rachel	Hardware	28K
S_2	Sharon	Rachel	Hardware	25K

(b)

STATE	EMP	MGR	DEPT	SAL
S_3	Beth	Beth	Linen	23K
S_3	Elsie	Elsie	Toy	27K
S_3	Rachel	Peter	Hardware	28K
S_3	Sharon	Peter	Hardware	25K
S_3	Peter	Peter	Hardware	33K

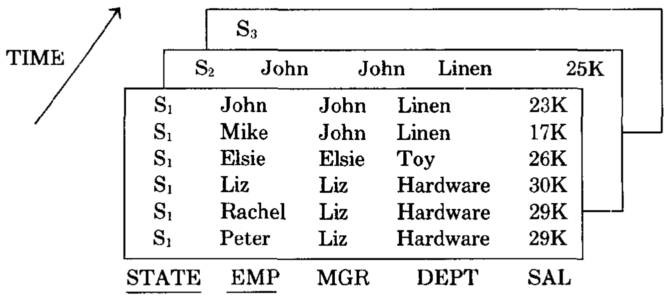
(c)

Fig. 4. (a) Relation emp_rel$'_1$; (b) relation emp_rel$'_2$; (c) relation emp_rel$'_3$.

We thus adopt an obvious notational convenience that a relation instance r_i is to be associated with state S_i.

We would like to view these new relation instances emp_rel$'_i$ as providing historical information about the changing values of the attributes of the objects denoted by values of the key, in this instance about EMPloyees. In order to visualize more clearly what is going on, we propose the picture of a historical relation as a "three-dimensional relation," each plane of which is a "static" or planar relation instance on EMP_REL for a given state of the world S_i. Time adds the third dimension to the normal flat-table view of relations. In a tabular relation, we understand that a row or tuple corresponds to the information about a particular object, and a column corresponds to the active domain of a particular attribute. We now propose to view each non-key attribute, such as SAL, as a set of roles related to the objects given by the key values, for example, John's SALary, Mike's SALary, and so on. In order to see more easily exactly what individuals fill these roles in each state, we want to "line up" the entities in the cube (sort on the key attribute). Figure 5 illustrates such a cube for the emp_rel relation.

Figure 5 also illustrates a problem that we must solve, namely, that some EMPloyees are not represented in every state. For example, John is not an EMPloyee in state S_3, and therefore there is no tuple for John in the plane for S_3 in this cube. Given the query, "What is John's salary in S_3?" we would want our model to give us the power to say not that there is no such employee, but rather that John does not work for us in S_3.



STATE	EMP	MGR	DEPT	SAL
S_2	John	John	Linen	25K
S_1	John	John	Linen	23K
S_1	Mike	John	Linen	17K
S_1	Elsie	Elsie	Toy	26K
S_1	Liz	Liz	Hardware	30K
S_1	Rachel	Liz	Hardware	29K
S_1	Peter	Liz	Hardware	29K

Fig. 5. Relation emp_rel.

In order to provide a framework in which these issues can be examined, we introduce the concept of a completed relation. Later this notion will be incorporated into a more formal definition of a number of assumptions on the interpretation of a historical database, assumptions with the same flavor as the Closed World Assumption of Reiter [41] but expanded to incorporate the temporal dimension. In order to indicate which entities are of interest in any state, we will use a special Boolean-valued attribute EXISTS?. In those states in which an entity does not exist as an EMPloyee, EXISTS? will be 0 for that EMP, and all of the other attributes will be given the value $\perp$, a distinguished entity whose meaning is that no individual fills the role of that attribute, that is, the attribute does not apply. A completed relation will have a tuple in each state for every entity that is an EMPloyee in any state in the entire database. In this way, we will be able to follow objects and their attributes throughout all of the states of the database. To do this, we determine all of the objects (key values) that are represented in any relation instance, and we extend with a null tuple each instance that does not represent information about this object.

We formalize these ideas as follows. Given a relation scheme $R'(\underline{\text{STATE } K} A_1 \cdots A_n)$ and an instance r_i' on R', we define the *active key domain* (AKD) of r_i' on R' to be the set of all key values (entities) in the relation instance r_i', that is,

$$\text{AKD}(r_i') = \Pi_K(r_i').$$

We then extend this definition to a set of instances $I = \{r_1', \ldots, r_n'\}$ on R' by defining the *complete active key domain* (CAKD) of a set of instances as

$$\text{CAKD}(I) = \cup \text{AKD}(r_i') \qquad \text{for all} \quad r_i' \in I.$$

CAKD(I) is exactly the set we need—it represents all of the EMP entities about which any information is stored in the database.

We then extend each relation instance r_i' so that it has a tuple for each entity in CAKD(I), the set of all "possible" EMPloyees that are "actual" in some state. Now by construction the projection of each expanded instance r'' onto the attribute K will correspond to *all* of the entities, that is,

$$\Pi_K(r_i'') = \text{CAKD}(I).$$

If the entity k is an actual entity in state S_i, then in the expanded relation r_i'' the tuple t_k'' will have $t_k''(\text{EXISTS?}) = 1$ and will agree with the tuple t_k' in r_i' on every other attribute. On the other hand, if k is not an actual entity in state S_i,

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S_1	John	1	John	Linen	23K
S_1	Mike	1	Mike	Linen	17K
S_1	Elsie	1	Elsie	Toy	26K
S_1	Liz	1	Liz	Hardware	30K
S_1	Rachel	1	Liz	Hardware	29K
S_1	Peter	1	Liz	Hardware	29K
S_1	Sharon	0	⊥	⊥	⊥
S_1	Beth	0	⊥	⊥	⊥

(a)

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S_2	John	1	John	Linen	25K
S_2	Mike	1	Elsie	Toy	20K
S_2	Elsie	1	Elsie	Toy	27K
S_2	Rachel	1	Rachel	Hardware	28K
S_2	Sharon	1	Rachel	Hardware	25K
S_2	Peter	0	⊥	⊥	⊥
S_2	Beth	0	⊥	⊥	⊥
S_2	Liz	0	⊥	⊥	⊥

(b)

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S_3	Beth	1	Beth	Linen	23K
S_3	Elsie	1	Elsie	Toy	27K
S_3	Rachel	1	Peter	Hardware	28K
S_3	Sharon	1	Peter	Hardware	25K
S_3	Peter	1	Peter	Hardware	33K
S_3	John	0	⊥	⊥	⊥
S_3	Liz	0	⊥	⊥	⊥
S_3	Mike	0	⊥	⊥	⊥

(c)

Fig. 6. (a) Relation emp_rel$_1''$; (b) relation emp_rel$_2''$; (c) relation emp_rel$_3''$.

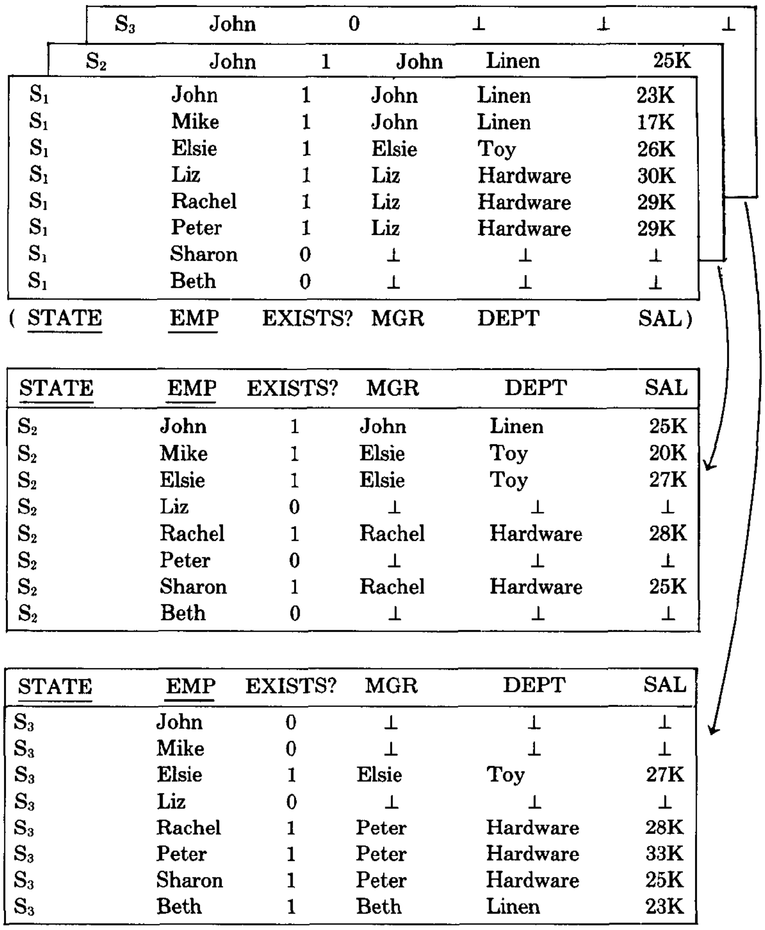
then the tuple t_k'' will have t_k'' (EXISTS?) = 0, but the distinguished value ⊥ for every other attribute other than STATE, indicating the *inapplicability* of this information for this entity, that is, that no individual fills the roles of these attributes for that entity. Formally, we define the *completed relation* as follows:

$$\begin{aligned} r_i'' = \{&t : t(R', \in r_i' \;\&\; t(\text{EXISTS?}) = 1\} \\ \cup \{&t : t(K) \in \text{CAKD}(I) - \text{AKD}(r_i') \;\&\; \\ &t(\text{STATE}) = S_i \;\&\; t(\text{EXISTS?}) = 0 \;\&\; \\ &t(A) = \bot \quad \text{for all} \quad A \in \{A_1, \ldots, A_n\}\}. \end{aligned}$$

The three completed EMPloyee relation instances are shown in Figure 6, arranged in a consistent (but arbitrarily chosen) order on key values.

The three-dimensional cube representation of the completed relation, such that the ith plane of the cube is r_i'', is shown in Figure 7.

The concept of a completed relation, combined with the EXISTS? attribute and the distinguished value ⊥, allows us to refer in any state to any entity (EMP) that is actual at any time in the database, and we can visually follow the changes



S_3	John	0	⊥	⊥	⊥
S_2	John	1	John	Linen	25K

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S_1	John	1	John	Linen	23K
S_1	Mike	1	John	Linen	17K
S_1	Elsie	1	Elsie	Toy	26K
S_1	Liz	1	Liz	Hardware	30K
S_1	Rachel	1	Liz	Hardware	29K
S_1	Peter	1	Liz	Hardware	29K
S_1	Sharon	0	⊥	⊥	⊥
S_1	Beth	0	⊥	⊥	⊥

Figure 7

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S_2	John	1	John	Linen	25K
S_2	Mike	1	Elsie	Toy	20K
S_2	Elsie	1	Elsie	Toy	27K
S_2	Liz	0	⊥	⊥	⊥
S_2	Rachel	1	Rachel	Hardware	28K
S_2	Peter	0	⊥	⊥	⊥
S_2	Sharon	1	Rachel	Hardware	25K
S_2	Beth	0	⊥	⊥	⊥

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S_3	John	0	⊥	⊥	⊥
S_3	Mike	0	⊥	⊥	⊥
S_3	Elsie	1	Elsie	Toy	27K
S_3	Liz	0	⊥	⊥	⊥
S_3	Rachel	1	Peter	Hardware	28K
S_3	Peter	1	Peter	Hardware	33K
S_3	Sharon	1	Peter	Hardware	25K
S_3	Beth	1	Beth	Linen	23K

in the facts about each EMPloyee through a three-dimensional row of the cube. In subsequent sections, we will introduce enough of the theory of IL_s to show how it can be applied to a historical database to provide a comprehensive database semantics capable of treating time-dependent queries and constraints.

At times we will want to consider all of these relation instances as comprising a single relation on the scheme EMP_REL. We can easily combine them into one large relation by taking their union. Accordingly, we define a *historical relation* r_h on a relation scheme $R''(\underline{\text{STATE}\ K}\ \text{EXISTS?}\ A_1 \cdots A_n)$ for a set of instances $I = \{r_1, r_2, \ldots, r_m\}$ as the union of the completed relations r_i'' that we have just constructed (Figure 8). There are no tuples lost in taking this union (i.e., there were no duplicates) because of the manner in which we have constructed each instance. Moreover, we now that {STATE, K} is a key of r_h. Finally, for each $S_i \in S$ the corresponding completed relation r_i'' is embedded in r_h, since

$$r_i'' = \sigma_{\text{STATE}=S_i}(r_h).$$

We shall use the term *historical relation* in the rest of this paper indiscriminately to refer either to this single relation or to the three-dimensional organization of

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S_1	John	1	John	Linen	23K
S_1	Mike	1	John	Linen	17K
S_1	Elsie	1	Elsie	Toy	26K
S_1	Liz	1	Liz	Hardware	30K
S_1	Rachel	1	Liz	Hardware	29K
S_1	Peter	1	Liz	Hardware	29K
S_1	Sharon	0	$\perp$	$\perp$	$\perp$
S_1	Beth	0	$\perp$	$\perp$	$\perp$
S_2	John	1	John	Linen	25K
S_2	Mike	1	Elsie	Toy	20K
S_2	Elsie	1	Elsie	Toy	27K
S_2	Rachel	1	Rachel	Hardware	28K
S_2	Sharon	1	Rachel	Hardware	25K
S_2	Peter	0	$\perp$	$\perp$	$\perp$
S_2	Beth	0	$\perp$	$\perp$	$\perp$
S_2	Liz	0	$\perp$	$\perp$	$\perp$
S_3	Beth	1	Beth	Linen	23K
S_3	Elsie	1	Elsie	Toy	27K
S_3	Rachel	1	Peter	Hardware	28K
S_3	Sharon	1	Peter	Hardware	25K
S_3	Peter	1	Peter	Hardware	33K
S_3	John	0	$\perp$	$\perp$	$\perp$
S_3	Liz	0	$\perp$	$\perp$	$\perp$
S_3	Mike	0	$\perp$	$\perp$	$\perp$

Fig. 8. Relation emp_rel$_h$.

the completed relation instances; no confusion should arise, since both of these representations represent the same information.

We can now define a *historical relational database* as a collection of historical relations over the same set of states. In what follows, we continue to use the term *static database* as a general term to describe those familiar databases which attempt to model only one state of the world.

The development of the historical relation emp_rel$_h$ in this section has been very informal; it has been presented in this way because viewing such a database as a three-dimensional object aids our intuition. The technique of time-stamping each tuple is a fairly simple idea, and many databases have kept information such as salary histories in a similar way. It is important to note, however, that the STATE and EXISTS? attributes are distinguished attributes that are an intrinsic part of the historical database model, and not ordinary attributes under the user's direct control. By this means an explicit temporal semantics can be incorporated directly within the framework of the relational model, provided that the model is extended to include a special treatment for these attributes. We have tried in this section to provide a reasonable intuition about the added dimension that time contributes to database semantics. In Section 6, we will show how the model, and not the user, provides a temporal semantics by the interpretation that it gives to these distinguished attributes and to its interaction with all of the other elements in the basic relational model. Through the technique of meaning postulates [6, 35], which are axioms that constrain the set of allowable models, the user is provided with the facility to make certain modifications to the general temporal semantics provided in the general historical relational model. Since this semantics depends upon the formalization of IL$_s$, we provide a brief overview of this logic and its model theory in the next section. Those familiar with Montague's formulation of IL [35] will see that in IL_s we have reformulated IL to include s as a basic type, along the lines suggested in [18].

5. INTENSIONAL LOGIC AND INTENSIONAL MODELS

Most database researchers have some degree of familiarity with the general concepts and some of the theory of first-order logic, if not with its model theory then at least with its deductive apparatus. We hope that what we provide here in the way of introduction to intensional logic will at least suffice to make the rest of this paper intelligible; should we inspire some readers to seek broader knowledge of the subject, we recommend [14] as an excellent introduction before plunging headlong into Montague's extremely terse presentation [35].

IL_s is a *typed*, higher order lambda calculus incorporating indexical semantics. It is typed: Every expression in IL_s has an associated type, which determines what kind of object in the intensional model for the language can be assigned to it by an interpretation function as its denotation. It is *higher order*: Unlike first-order languages which allow quantification only over individuals, or second-order languages which allow quantification only over individuals or sets of individuals, IL_s allows quantification over variables of every type. It is a *lambda calculus*: It provides a lambda operator which allows the formation of expressions denoting constructed functions of arbitrary type (see [10]). (Readers familiar with the programming language LISP [32] are familiar with the general concepts of lambda abstraction. Hobbs and Rosenschein [23] exploit this similarity in their attempt to interpret a simplified version of Montague's IL as LISP expressions.) Finally, it incorporates *indexical semantics* by including in the syntax expressions of a type whose interpretation is a special set of indices or states, and by having a model theory that is based upon a possible worlds/temporal (or indexical) semantics.

Definition. The set of types for IL_s is the smallest set T such that:

1. e, t, and s are in T, and
2. if $a, b \in T$, then $\langle a, b\rangle \in T$.

We anticipate the semantic discussion below to say that the interpretation function for the language will assign to expressions of type e (for entity) individuals in the model; to expressions of type t (for truth values), one of the truth values 0 (false) or 1 (true); to expressions for type s (for states), states or points of reference; and to expressions of type $\langle a, b\rangle$, some function from objects in the model of type a to objects of type b.

We shall not present the complete syntax of IL_s, since the examples we use in the following sections use only a portion of the language. Instead we stress the following points of notation and departures from standard first-order languages.

1. IL_s contains an infinite number of variables of the form $v_{n,a}$ for each type a and natural number n, and a set of constants C_a, possibly empty, for each type a.

2. IL_s contains the usual truth function operator $\neg$ (NOT), and truth functional connectives $\wedge$ (AND), $\vee$ (OR), $\rightarrow$ (material implication), $\leftrightarrow$ (mutual material implication), $=$ (equality), and $<$ (prior to).

3. IL_s contains the universal and existential quantifiers, $\forall$ and $\exists$, respectively.

The usual rules of formation apply to the above language elements. In addition, the following syntactic constructs are peculiar to IL_s.

4. If α is an expression of type $\langle a, b\rangle$ and β is an expression of type a, then $\alpha(\beta)$ is an expression of type b, and denotes the result of applying the function denoted by α to the object denoted by β as argument.

5. If x is a variable of type a, and β an expression of type b, then $\lambda x\beta$ is an expression of type $\langle a, b\rangle$, and denotes a particular function from objects of type a to objects of type b.

Not surprisingly, it is the model theory of IL_s that is of most interest to us here. We proceed by first formally defining a model for IL_s and then discussing its significance.

A model M for the language IL_s is an ordered 4-tuple $M = \langle E, S, < F\rangle$ where

1. E is a nonempty set (the set of basic entities);
2. S is a non-empty set (the set of states);
3. $<$ is a linear ordering on S (this gives the interpretation of the "prior to" symbol $<$ in the language);
4. F is the function which assigns to each constant $c_a \in C_a$ an element in D_a, the set of *possible denotations of expressions of type a*, which is defined recursively over the set of types T as follows.

$$D_e = E$$

$$D_t = \{0, 1\}$$

$$D_s = S$$

$$D_{\langle a,b\rangle} = D_b{}^{D_a}, \qquad \text{that is, the set of all functions from } D_a \text{ to } D_b.$$

The set E is intended to represent the set of possible individuals, and S the set of points of reference or states, ordered by $<$.

We point out that, in particular, an expression of type $\langle a, t\rangle$ for any type a denotes a function from D_a into $\{0, 1\}$ and can therefore be thought of as the characteristic function of a set of objects in D_a. Accordingly, we will often speak, for example, of sets of individuals, when we should more formally speak of functions from individuals to $\{0, 1\}$. For example, over a universe consisting of the set $\{a, b, c, d, e\}$, the set $\{a, c, e\}$ is equivalently represented by the following characteristic function.

$$\begin{bmatrix} a \rightarrow 1 \\ b \rightarrow 0 \\ c \rightarrow 1 \\ d \rightarrow 0 \\ e \rightarrow 1 \end{bmatrix}$$

Probably the best way to get a feeling for what these definitions say is to set up a small language and model and provide some examples. Let us therefore assume a language that contains the following constants of the indicated types.

$$\text{Peter, Liz, Elsie, and THE\_BOSS of type } \langle s, e\rangle$$

$$77,\ 78,\ 79,\ 80, \text{ and } 81 \text{ of type } s,$$

and

$$\text{EMP of type } \langle s, \langle e, t\rangle\rangle.$$

Let us also assume that our model $M = \langle E, S, <, F\rangle$ is defined as follows.

$$E = \{\underline{\text{Peter}}, \underline{\text{Liz}}, \underline{\text{Elsie}}\}$$

$$S = \{1977, 1978, 1979, 1980, 1981\}$$

with < the obvious ordering on S.

Assume that the interpretation function F makes the obvious assignments to the state constants. The other constants are interpreted as follows.

$$F(\text{Peter}) = \begin{bmatrix} 1977 \rightarrow \underline{\text{Peter}} \\ 1978 \rightarrow \underline{\text{Peter}} \\ 1979 \rightarrow \underline{\text{Peter}} \\ 1980 \rightarrow \underline{\text{Peter}} \\ 1981 \rightarrow \underline{\text{Peter}} \end{bmatrix} \qquad F(\text{Liz}) = \begin{bmatrix} 1977 \rightarrow \underline{\text{Liz}} \\ 1978 \rightarrow \underline{\text{Liz}} \\ 1979 \rightarrow \underline{\text{Liz}} \\ 1980 \rightarrow \underline{\text{Liz}} \\ 1981 \rightarrow \underline{\text{Liz}} \end{bmatrix}$$

$$F(\text{Elsie}) = \begin{bmatrix} 1977 \rightarrow \underline{\text{Elsie}} \\ 1978 \rightarrow \underline{\text{Elsie}} \\ 1979 \rightarrow \underline{\text{Elsie}} \\ 1980 \rightarrow \underline{\text{Elsie}} \\ 1981 \rightarrow \underline{\text{Elsie}} \end{bmatrix}$$

These functions, from states to individuals, are what we have defined above as *individual concepts* (ICs). They are intended to represent the sense of a name since they pick out the individual referred to by the name at every index. The ICs above all share the additional property of being constant ICs (or rigid designators): in each state S_i, they pick out the same individual. Compare how F interprets the constant THE_BOSS:

$$F(\text{THE\_BOSS}) = \begin{bmatrix} 1977 \rightarrow \underline{\text{Liz}} \\ 1978 \rightarrow \underline{\text{Peter}} \\ 1979 \rightarrow \underline{\text{Peter}} \\ 1980 \rightarrow \underline{\text{Peter}} \\ 1981 \rightarrow \underline{\text{Elsie}} \end{bmatrix}$$

This function is also an IC, but it is not a constant IC. Later we shall see how this distinction between constant and unconstrained ICs will be related to the database concepts of key and non-key attributes, respectively. We can think of this function as representing the role of the boss, in that it tells who fills that role in every state. The interpretation of EMP will be a function which, for any state, picks out a set of individuals (the intended interpretation being that set of individuals who are EMPloyees in that state):

$$F(\text{EMP}) = \begin{bmatrix} 1977 \rightarrow \{\underline{\text{Liz}}\} \\ 1978 \rightarrow \{\underline{\text{Peter}}, \underline{\text{Liz}}\} \\ 1979 \rightarrow \{\underline{\text{Peter}}, \underline{\text{Liz}}\} \\ 1980 \rightarrow \{\underline{\text{Peter}}\} \\ 1981 \rightarrow \{\underline{\text{Elsie}}\} \end{bmatrix}$$

Such a function is often called a property of individuals. Notice that we have used set notation instead of the more cumbersome, though equivalent representation by characteristic functions.

Rather than giving the semantic rules for IL_s which, for each expression A, define the *extension of A with respect to a model M, a state i, and a variable assignment g,* we provide some examples. Consider the expression EMP(78). Since EMP is of type $\langle s, \langle e, t\rangle\rangle$, and 78 is of type s, this expression is well formed and is of type $\langle e, t\rangle$. Its interpretation is given by applying the function which is the interpretation of EMP to the interpretation of 78, namely, 1978, as shown:

$$\begin{bmatrix} 1977 \rightarrow \{\underline{\text{Liz}}\} \\ 1978 \rightarrow \{\underline{\text{Peter}}, \underline{\text{Liz}}\} \\ 1979 \rightarrow \{\underline{\text{Peter}}, \underline{\text{Liz}}\} \\ 1980 \rightarrow \{\underline{\text{Peter}}\} \\ 1981 \rightarrow \{\underline{\text{Elsie}}\} \end{bmatrix} \quad (1978) = \{\underline{\text{Peter}}, \underline{\text{Liz}}\}.$$

Thus we see that the interpretation rules give the expected meaning to EMP(78), namely, $\{\underline{\text{Peter}}, \underline{\text{Liz}}\}$, the set of individuals who are EMPloyees in 1978. Consider now the expression EMP(78) (Elsie), of type t. The denotation of this expression is "computed" by applying the set $\{\underline{\text{Peter}}, \underline{\text{Liz}}\}$, (considered as a function) to the argument $\underline{\text{Elsie}}$ to obtain the value 0 (false), that is, Elsie is not an EMPloyee in 1978.

Now, suppose we want to form an expression whose denotation is a function from states to those individuals who were not the boss in those states. Such an expression would be of the same type as the constant EMP, namely, $\langle s, \langle e, t\rangle\rangle$, and can be constructed from the constants we have so far defined using lambda abstraction over the set of states and the set of individuals. In order to do this, we need to use two variables in the logic: a variable i of type s, that is, a variable over states, and a variable u of type e, a variable over individuals. We already know that the interpretation function F gives the interpretation of each nonlogical constant. The variable assignment g, as in first-order languages, provides the denotation of variables. Explicitly, for every variable y of type a, $g(y) \in D_a$. With these two variables we can form an expression which denotes the function we want, namely,

$$\lambda i \lambda u\, [\neg\, \text{THE\_BOSS}(i)(u)].$$

The denotation of this function, let us call it N_T_B, is given in Figure 9. We have indicated the sets by their characteristic functions; in each year, one and only one person is *not* not the boss (namely, the one who *is* the boss).

Finally, we consider an example that makes explicit reference to time, the

$$\mathrm{N\_T\_B} = \begin{bmatrix} 1977 \rightarrow \begin{bmatrix} \text{Liz} \rightarrow 0 \\ \text{Peter} \rightarrow 1 \\ \text{Elsie} \rightarrow 1 \end{bmatrix} \\ 1978 \rightarrow \begin{bmatrix} \text{Liz} \rightarrow 1 \\ \text{Peter} \rightarrow 0 \\ \text{Elsie} \rightarrow 1 \end{bmatrix} \\ 1979 \rightarrow \begin{bmatrix} \text{Liz} \rightarrow 1 \\ \text{Peter} \rightarrow 0 \\ \text{Elsie} \rightarrow 1 \end{bmatrix} \\ 1980 \rightarrow \begin{bmatrix} \text{Liz} \rightarrow 1 \\ \text{Peter} \rightarrow 0 \\ \text{Elsie} \rightarrow 1 \end{bmatrix} \\ 1981 \rightarrow \begin{bmatrix} \text{Liz} \rightarrow 1 \\ \text{Peter} \rightarrow 1 \\ \text{Elsie} \rightarrow 0 \end{bmatrix} \end{bmatrix}$$

Fig. 9 The denotation of function N_T_B.

formula which translates the sentence "Elsie was the boss":

$$\exists i[[i < \mathit{now}] \wedge \mathrm{THE\_BOSS}(i)(\mathrm{Elsie})].$$

If we assume that *now* (of type *s*) is interpreted as 1981, this formula will be true just in case there is some time *i* prior to 1981 at which Elsie was "the boss." It is easy to see that with respect to the model *M* this formula is false, and the inductive definition of the interpretation of the language IL_s makes this formula denote 0.

This completes our brief introduction to the language IL_s and its semantics. It should be sufficient to enable the reader unfamiliar with formalized intensional logic to understand the following section, in which we present a detailed discussion of the model-theoretic implications of a historical relational database.

6. INTENSIONAL LOGIC AND HISTORICAL DATABASES

In this section we describe the relationship between the historical relational database model and the logic IL_s and its model theory. This relationship is first presented formally. It is then followed by an informal discussion that emphasizes insights that this relationship can provide into the way that a database models the "real world," and into the nature of entities and relationships, key and non-key attributes, queries and data constraints, and the interaction of time with all of these concepts. The formalism is presented in the interest of completeness and rigor, but it is easy to get lost in some of the notation; the informal discussion provides a better overview of how the temporal dimension is incorporated into the traditional relational model and of how it affects this model.

In the previous section, we described the syntax and semantics of the language IL_s. To be more precise we should rather say the family of IL_s languages, since any particular language in this family is determined by the set *C* of nonlogical constants. The general, intuitive description of the historical relational database concept presented in Section 4 will now be formalized and related to the discussion of the intensional logic as follows. First, we show that a particular HDB scheme defines a particular logic in the family of IL_s languages that provides a formal expression of the historical database semantics and that serves as the target language for translations from our English Query Language (described in [11]). Second, we show how the interpretation of the set of nonlogical constants of this applied IL_s is given by an instance of an HDB on this scheme at any moment in its history.

6.1 Introduction

In IL_s, as in Montague's formulation of IL, all functions are defined as taking only one argument. It is well known, however (see the discussion in [10]), that any function of n arguments can be represented by an equivalent function of one argument whose value is a function of $n - 1$ arguments. Thus, for example, if f is a function of two arguments, $(f(a))(b)$ represents the value of f for the arguments a and b. The function $f(a)$ represents a function of one variable whose value for any argument x is $(f(a))(x)$. We shall abbreviate this notation as $f(a)(x)$ or as $f(\langle a, x\rangle)$, and assume that the generalization to functions of n arguments is obvious. Thus, if g is a function of n arguments, $g(x_1)(x_2) \cdots (x_n)$ or $g(\langle x_1, x_2, \ldots, x_n\rangle)$ abbreviates $(((g(x_1))(x_2)) \cdots)(x_n)$, which represents the value of the n-ary function g for the arguments $x_1, x_2, \ldots, x_n$.

In our discussion of functions, we will have occasion to speak of particular *function spaces*, that is, the set of all functions with the same domain and the same range. For example, the set of all functions with domain S (states) and range E (individuals) is written E^S. Recalling our notation for the denotation sets corresponding to a given type in IL_s, this function space can also be written $D_e^{D_s}$, and represents the set of all ICs. We will sometimes refer to a given function in this function space as being of *type* $\langle s, e\rangle$, although strictly speaking we should rather say that if, for example, X is a term in the language IL_s that names this function, then X is of type $\langle s, e\rangle$. In general, a function from A to B will be said to be of type $\langle A, B\rangle$. Many of the nonlogical constants that we will be discussing will be of types such as $\langle s, \langle e, \langle e, \ldots, \langle e, t\rangle, \ldots \rangle\rangle\rangle$, where there are n e's before the t. Instead of this cumbersome notation, we will abbreviate such a type as $\langle s, \langle e^n, t\rangle\rangle$.

Definition. The *logical domain of a database attribute* A, $\mathrm{LD}(A)$, that is, the domain in the logical model that corresponds to the values in the database domain of the attribute A, is defined as follows.

$$\mathrm{LD}(A) = \left| \begin{array}{ll} \mathrm{S} & \text{if } A = \mathrm{STATE} \\ \mathrm{TV} & \text{if } A = \mathrm{EXISTS?} \\ \mathrm{E} & \text{otherwise} \end{array} \right| .$$

Definition. We say that X is an $\langle A_1, A_2, \ldots, A_n\rangle$-value for a relation scheme $R(\cdots A_1A_2 \cdots A_n \cdots)$, if $X = \langle x_1, x_2, \ldots, x_n\rangle$ where $x_i \in \mathrm{LD}(A_i)$, $1 \le i \le n$. If $n = 1$ we sometimes omit the braces and say simply that X is an A_1-value.

Definition. We modify the definition of the relational database *projection operator* Π to handle the special case of projection of a null relation differently according to the LD of the attribute projected upon:

$$\Pi_A(r) = \left| \begin{array}{llll} \{t(A) : t \in r\} & \text{if } r \ne \varnothing & & \\ \perp & \text{if } r = \varnothing & \text{and} & \mathrm{LD}(A) = \mathrm{E} \\ 0 & \text{if } r = \varnothing & \text{and} & \mathrm{LD}(A) = \mathrm{TV} \\ \varnothing & \text{if } r = \varnothing & \text{and} & \mathrm{LD}(A) = \mathrm{STATE} \end{array} \right| .$$

We call the elements $\perp$, 0, and $\emptyset$ *bottom* of the logical domains E, TV, and STATE, respectively. With this modified project operator we will be able to define a total function from a relation defined over only a subset of the set of STATES, given certain simple assumptions on how to interpret the database.

We have chosen in this work to adopt the entity-relationship view of data semantics [9], as applied to the relational model, for two main reasons. First, we view the constraints that the entity-relationship model makes upon the database view of an enterprise as rather "natural" constraints that accord with our intuition. Second, these same constraints appear to have direct logical analogues in the kinds of objects—entities, relationships, and properties—contained in the model theory of our logic. Since Montague's Intensional Model Theory and Chen's Entity-Relationship Model are two independent efforts to characterize real-world semantics, we feel that the similarity in some of their concepts strengthens their intuitive appeal. The constraints of the entity-relationship model applied to the historical database concept, combined with some simple assumptions on how to interpret a historical database, allow us to define a reasonably straightforward mapping between any relational HDB that conforms to these constraints and an IL_s model.

We proceed to define the entity-relationship constraints that we place upon the more general HDB model presented in Section 5. We then define first the IL_s language that a given HDB scheme defines, and second the model M_{hdb} for that language that is induced by an instance hdb on this scheme.

Definition. A *historical entity relation* is a historical relation r_h on a scheme of the form ($\underline{\text{STATE } K_1}$ EXISTS? $A_1 \cdots A_n$) with the following constraints.

1. K_1 and $A_1 \cdots A_n$ are as in an entity relation.

2. An entity can belong to only one "entity-set." That is, if r_1 is a historical entity relation on R_1 ($\underline{\text{STATE } K_1}$ EXISTS? $A_1 \cdots A_m$) and r_2 is a historical entity relation on R_2 ($\underline{\text{STATE } K_2}$ EXISTS? $A'_1 \cdots A'_n$), then for any $t_1 \in r_1$ and $t_2 \in r_2$, $t_1(K_1) \neq t_2(K_2)$.

3. For any tuple t in r_h, if t(EXISTS?) = 1 then the entity represented by $t(K_1)$ is said to exist in the state given by t(STATE), and the values of $t(A_1)$, $1 \leq i \leq n$, must not be $\perp$. (Note that we do not build into the model any other kinds of null values.)

4. For any tuple t in r_h, if t(EXISTS?) = 0 then the entity represented by $t(K_1)$ is said not to exist in the state given by t(STATE), and the values of $t(A_1)$, $1 \leq i \leq n$, must all be $\perp$.

Definition. A *historical relationship relation* is a relation r_h on a scheme of the form ($\underline{\text{STATE } K_1 \cdots K_n}$ EXISTS? $A_1 \cdots A_m$) with the following constraints.

5. $K_1 \cdots K_n$ and $A_1 \cdots A_m$ are as in relationship relations.

6. For any tuple t in r_h, if t(EXISTS?) = 1 then the relationship represented by $t(K_1 \cdots K_n)$ is said to exist in the state given by t(STATE), and the values of $t(A_1)$, $1 \leq i \leq n$, must not be $\perp$.

7. For any tuple t in r_h, if t(EXISTS?) = 0 then the relationship represented by $t(K_1 \cdots K_n)$ is said not to exist in the state given by t(STATE), and the values of $t(A_1)$, $1 \leq i < n$ must all be $\perp$.

Moreover, the following interrelational constraints must be satisfied.

8. Only one relationship is allowed among (between) the same entity sets. That is, it is not permitted to have more than one historical relationship relation whose object key is $K_1 \cdots K_n$.

9. For each historical relationship r_h with entity keys $\{K_1, K_2, \ldots, K_n\}$, there must exist, for each of the K_i, a corresponding historical entity relation r_i such that for each tuple t in r_h with $t(\text{EXISTS?}) = 1$, there must exist in the relation r_i corresponding to K_i a tuple t' such that $t'(K_i) = t(K_i)$, $t'(\text{STATE}) = t(\text{STATE})$, and $t'(\text{EXISTS?}) = 1$.

10. A role attribute A can appear as a role attribute in at most one relation. If role attribute A in r_1 is an entity attribute K in r_2, then for each tuple t in r_1 with $t(\text{EXISTS?}) = 1$ there must be a t' in r_2 with $t(\text{STATE}) = t'(\text{STATE})$ and $t(A) = t(K)$.

These last two interrelational constraints ensure that if an entity k participates in a relationship or fills a role in a state s, then the existence of k in state s must be predicated in the entity relation for k. All of these constraints are essentially the same as in the general entity-relationship model, extended to include a temporal semantics.

Definition. We will sometimes wish to refer to database entities or relationships by the neutral word *object* or *object of arity n;* if $n = 1$, this term refers to an entity, whereas if $n > 1$, it refers to an n-ary relationship.

Definition. By a *historical database* (HDB), we shall mean a collection of historical entity and historical relationship relations that satisfy the above constraints, which we shall refer to as the *historical entity-relationship constraints.*

6.2 The IL_s Language Defined by an HDB Scheme

The information in an HDB is organized in the form of historical entity and historical relationship relations. We represent this information in the logical model by some set of functions which are defined implicitly by the database. In this section we give the names of the functions that are needed to represent the HDB as a portion of an intensional model. These names are simply a set of nonlogical constants that define a particular IL_s language. In this section, we only briefly discuss the sorts of functions denoted by these constants; in the following section, we shall show how any instance of an HDB induces the interpretation of these constants.

For each HDB, we shall define six sorts of constants, corresponding to *domain values, time values, entity attributes, role attributes, relationships,* and the *associations* between objects (entities or relationships) and their role attributes. In the discussion to follow, we shall have occasion to make reference to a sample database to exhibit some of the ideas that we discuss. We therefore define a simple historical database based on the department-store relational database example in [8].

Historical Entity Relation Schemes

EMP_REL ($\underline{\text{STATE}}$ $\underline{\text{EMP}}$ EXISTS? MGR DEPT SAL)
DEPT_REL ($\underline{\text{STATE}}$ $\underline{\text{DEPT}}$ EXISTS? FLOOR)
ITEM_REL ($\underline{\text{STATE}}$ $\underline{\text{ITEM}}$ EXISTS? TYPE)

Historical Relationship Relation Schemes

SALES_REL ($\underline{\text{STATE}}$ $\underline{\text{DEPT}}$ $\underline{\text{ITEM}}$ EXISTS? VOL)

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S_1	Peter	0	$\perp$	$\perp$	$\perp$
S_1	Liz	0	$\perp$	$\perp$	$\perp$
S_1	Elsie	1	Elsie	Toy	50
S_2	Peter	1	Elsie	Hardware	30
S_2	Liz	1	Elsie	Toy	35
S_2	Elsie	1	Elsie	Toy	50
S_3	Peter	1	Liz	Linen	35
S_3	Liz	1	Liz	Hardware	50
S_3	Elsie	0	$\perp$	$\perp$	$\perp$

Fig. 10. Relation emp_rel.

STATE	DEPT	EXISTS?	FLOOR
S_1	Toy	1	F1
S_1	Hardware	1	F2
S_1	Linen	0	$\perp$
S_2	Toy	1	F2
S_2	Hardware	1	F2
S_2	Linen	1	F3
S_3	Toy	1	F2
S_3	Hardware	0	$\perp$
S_3	Linen	1	F3

Fig. 11. Relation dept_rel.

We will also have occasion to use the instances given in Figures 10–13 over these schemes.

6.2.1 *Domain Value Constants* (*DVCs*). Recall that the union of all of the domains of the database attributes is the set UD. Corresponding to UD we define the set of individual constants in IL_s, $C_e = \{d' \mid d \in UD\}$, so that we can refer in the logic to any value that might appear in any state of the database.

6.2.2 *Time constants* (*TCs*). The domain of the distinguished attribute STATE is the set S. Corresponding to this set, we define the set of state constants in IL_s, $C_s = S$. It will also prove useful to allow constants that refer to sets of states, in particular to contiguous states or intervals; for example, a constant 1978 of type $\langle s, t\rangle$ would denote the set of all moments of time in the year 1978. We will therefore allow in IL_s a set of constants of this type, namely, $C_{\langle s,t\rangle}$. These latter are not determined by the database, but rather by the kinds of users and queries that the database system is intended to support.

The general picture of the historical database as encoded in the IL_s model is provided by the denotations of the remaining four kinds of constants. Before stating formally the rules for deriving their denotations from the database, we give the following overview.

1. The set of entities (e.g., EMPloyees) in any state is given by the denotation of the corresponding entity constant (e.g., EMP'_*) for the entity set.
2. The set of n-tuples participating in any n-ary relationship in any state is given by the denotation of the relationship constant REL-n. All n-ary relation-

STATE	ITEM	EXISTS?	TYPE
S_1	Ball	1	5
S_1	Game	1	6
S_1	Glove	1	7
S_2	Ball	1	0
S_2	Game	1	6
S_2	Glove	1	5
S_3	Ball	1	10
S_3	Game	0	$\perp$
S_3	Glove	0	$\perp$

Fig. 12. Relation item_rel.

STATE	DEPT	ITEM	EXISTS?	VOL
S_1	Toy	Ball	1	3
S_1	Toy	Game	1	6
S_1	Hardware	Glove	1	9
S_1	Linen	Glove	0	$\perp$
S_2	Toy	Ball	1	3
S_2	Toy	Game	1	6
S_2	Hardware	Glove	1	9
S_2	Linen	Glove	1	2
S_3	Toy	Ball	1	4
S_3	Toy	Game	1	6
S_3	Hardware	Glove	0	$\perp$
S_3	Linen	Glove	0	$\perp$

Fig. 13. Relation sales_rel.

ships can be combined into a single function since the entity sets of the participants uniquely determine the relationship.

3. For each role (e.g., SALary), the set of ICs that fill that role in any state is given by the denotation of the corresponding role constant (e.g., SAL′). An IC fills a role only in those states in which its associated object exists (or, equivalently, in which its value is not $\perp$).

4. n-ary objects are bound permanently to each of their role ICs A_i by the denotation of the nonindexical constants AS-A_i. Thus, for example, each EMPloyee is permanently bound to three ICs which, in those states in which the employee exists, are its SAL, MGR, and DEPT selecting functions.

6.2.3 *Entity Existence Constants (EECs).* For each historical entity relation with entity key K, we use a nonlogical constant K'_* in IL_s of type $\langle s, \langle e, t\rangle\rangle$ which denotes, at each state, the set of individuals (subset of E) which exist as K-entities in the state. For example, the historical entity relation DEPT_REL with entity key DEPT induces in the logic the constant DEPT$'_*$ of type $\langle s, \langle e, t\rangle\rangle$. DEPT$'_*$ denotes at any state the set of entities which are departments in that state. $C_{\langle s,\langle e,t\rangle\rangle}$ is the set of all these entity-key constants.

6.2.4 *Relationship Existence Constants (RECs).* For each n for which there exists one or more n-ary historical relationship relations, the set $C_{\langle s,\langle e^n,t\rangle\rangle}$ consists of the single nonlogical constant REL-n, which denotes at each state the set of logical n-tuples (subset of E^n) which exist as n-ary relationships in that state. For example, SALES_REL is a binary historical relationship relation that induces in the logic the constant REL-2 of type $\langle s, \langle e, \langle e, t\rangle\rangle\rangle$. REL-2 denotes at any state the set of binary relationships (in this example, this is just the set of DEPT-ITEM pairs) that exist in that state.

6.2.5 *Role Constants* (*RCs*). For each role attribute A in the historical database scheme, we use a nonlogical constant A' of type $\langle s, \langle\langle s, e\rangle, t\rangle\rangle$ in IL_s which denotes, at each state, the set of A-ICs which exist in that state. $C_{\langle s,\langle\langle s,e\rangle,t\rangle\rangle}$ is the set of all of these role constants. For example, the role attributes DEPT (from EMP_REL) and VOL (from SALES_REL) induce in the logic the constants DEPT$'$ and VOL$'$ of type $\langle s, \langle\langle s, e\rangle, t\rangle\rangle$. DEPT$'$, for example, denotes in any state the set of DEPT-ICs (i.e., department-of-some-employee roles) that exist in that state. Notice that DEPT$'$ and DEPT$'_*$ are two different constants of different types, induced by two different "occurrences" (and two different uses) of the single database attribute DEPT. This distinction between object (entity or relationship) attributes and role attributes is an important one. The values of object attributes are entities, while the values of role attributes are functions (ICs). If, as in the case of departments in this example, an attribute is considered in one case (EMP_REL) as a role attribute (an attribute of the entity EMP) and in another as an object attribute (the entity department), two different constants denoting two different functions are induced in the logic. Attributes of a department are attributes of the department as an entity and not as a role.

6.2.6 *Association Constants* (*ACs*). For each n for which there is an object in the database the set $C_{\langle e^n,\langle\langle s,e\rangle,t\rangle\rangle}$ consists of a set of nonlogical constants AS-A_i which denote the permanent association (i.e., state-independent, or nonindexical) between each object of arity n and each of its role attributes A_i. For example, the constant AS-SAL of type $\langle e, \langle\langle s, e\rangle, t\rangle\rangle$ in the logic represents the association between each entity (object of arity 1) and its salary IC. AS-FLOOR associates each department with its floor IC, and so on. The constant AS-VOL of type $\langle e, \langle e, \langle\langle s, e\rangle, t\rangle\rangle\rangle$ represents the association between each binary DEPT-ITEM relationship and its sales-VOLume.

Any given HDB scheme thus determines a set C_{HDB} of constants in $\mathrm{IL}_{s,\mathrm{HDB}}$ from among these six categories of nonlogical constants. (These constants are uniquely determined except for the constants of type $\langle s, t\rangle$, for which many choices can be made.) In the case of the department-store database, the following set of constants is determined.

$$\begin{aligned} C_{\text{dept-store}} = {} & C_e \cup C_s \cup C_{\langle s,t\rangle} \\ & \cup C_{\langle s,\langle e,t\rangle\rangle} \\ & \cup C_{\langle s,\langle e,\langle e,t\rangle\rangle\rangle} \\ & \cup C_{\langle s,\langle\langle s,e\rangle,t\rangle\rangle} \\ & \cup C_{\langle e,\langle\langle s,e\rangle,t\rangle\rangle} \\ & \cup C_{\langle e,\langle e,\langle\langle s,e\rangle,t\rangle\rangle\rangle} \end{aligned}$$

where

C_e is the set of domain value constants,
C_s is the set of state constants,
$C_{\langle s,t\rangle}$ is some set of state-set constants,
$C_{\langle s,\langle e,t\rangle\rangle}$ = {EMP$'_*$, DEPT$'_*$, ITEM$'_*$} is the set of EECs,
$C_{\langle s,\langle e,\langle e,t\rangle\rangle\rangle}$ = {REL-2} is the set of RECs,
$C_{\langle s,\langle\langle s,e\rangle,t\rangle\rangle}$ = {MGR′, DEPT′, SAL′, FLOOR′, TYPE′} is the set of RCs, and
$C_{\langle e,\langle\langle s,e\rangle,t\rangle\rangle}$ = {AS-MGR, AS-DEPT, AS-SAL, AS-FLOOR, AS-TYPE} and $C_{\langle e,\langle e,\langle\langle s,e\rangle,t\rangle\rangle\rangle}$ = {AS-VOL}, and the union of these last two is the set of ACs.

In the following section, we will give formal definitions of an HDB scheme and an instance hdb on this scheme, and show how the interpretation of the constants determined by a given historical database scheme HDB is induced by an instance hdb over that scheme.

6.3 The Intensional Model Induced by a Database Instance

Before proceeding to define how a given instance of an HDB induces the definition of the interpretation-of-constants function F, we need to define some preliminary notions.

The view of a relational HDB as a three-dimensional cube composed of a sequence of static relations has served a useful purpose in guiding our intuition as to how time interacts with the other attributes in the database. It was this view which caused us to look at key attributes as constant ICs, functions from states to individuals, and at role attributes as unconstrained ICs. We will now argue that this view is inadequate in the face of the generally accepted notion of dense time. We will therefore fortify this view with two additional assumptions, the Comprehension Principle and the Continuity Assumption. These will enable us to view an HDB as modeling an enterprise completely over an interval of the real-time line, and to answer such crucial questions as what objects exist in any state s and what are the values of their A_i-ICs in these states.

Definition. A *closed interval* $[t_1, t_2]$ on the real-time line is defined, as usual, as the infinite set of all states in R between and including t_1 and t_2, that is, $[t_1 .. t_2] = \{t \mid t \in R \text{ and } t_1 \leq t \leq t_2\}$. The appropriately modified definitions for $[t_1, t_2)$, $(t_1, t_2]$, and (t_1, t_2) are assumed, and the general term interval will sometimes be used to refer to any of these.

For purposes of illustration let us consider again the historical entity relation scheme EMP_REL($\underline{\text{STATE}}$ $\underline{\text{EMP}}$ DEPT MGR SAL), and assume that we have an instance that is defined over this scheme for the sequence of states $\langle S_1, S_2, \ldots, S_7\rangle$. The first assumption which we shall make about any such relation is that it is intended to model EMPloyee entities over the entire closed interval of time $[S_1, S_7]$. Since under the most reasonable views of time this interval is assumed to be dense, the best that any finite relation can do is to provide a simulation of this infinite set of moments of time. If a relation is modeling contingent data, it simulates this dense interval by means of a sequence of snapshots, or still photos, in this case taken at each moment in the sequence $\langle S_1,$

$\ldots, S_7\rangle$. (Some relations model noncontingent data and can be computed, as described by Maier and Warren [30]; we will not consider such relations here.) Because we take this idea as basic, that is, because it seems to be the only reasonable interpretation to place on any historical database that records facts over some interval of time, we state it as the following principle.

Definition. The Comprehension Principle states that under any reasonable interpretation a historical database defined over a sequence of states $\langle S_1, S_2, \ldots, S_n\rangle$ should be considered as modeling an enterprise completely over the entire closed interval $[S_1, S_n]$. Any and all information about the objects of interest to the enterprise can be assumed to be contained in or implied by the historical database for the entire interval $[S_1, S_n]$. Moreover, for any state S not in the interval $[S_1, S_n]$, as far as the database is concerned, no entities or relationships exist, and the value of all ICs is $\perp$.

One area for further research would be the relaxation of the second part of this principle, related to the Closed World Assumption of Reiter [41], perhaps with the introduction of a many-valued logic. In our model, the set TV of truth values is the set $\{0, 1\}$, and we use 0 (false) as the obvious choice to mean *does not exist.* It is because no such obvious choice exists from the set E of entities that we have augmented E with the distinguished entity $\perp$, which can be considered as meaning "inapplicable." We do not thereby pretend to be offering anything more than a practical solution to the interesting philosophical problems of existence, properties of nonexistent but possible entities, and so on, which are of considerable philosophical and logical interest (Quine in particular [39, 40] has contributed a great deal to the understanding of these issues from both points of view). We point out that $\perp$ is the only so-called *null value* that we provide with a special semantics in this model. Future work might incorporate others as a formal null value semantics is developed. (The entire issue of null values in relational databases is discussed in [19].)

It remains only to make an assumption about what the database *means to say* about all those other moments of time which fall in the interval $[S_1, S_7]$ but which are not included in the sequence $\langle S_1, S_2, \ldots, S_7\rangle$ specifically mentioned in the database.

The problem stated in simple terms is this. The database samples the values of the ICs of interest for only some finite subset of states in $[S_1, S_n]$, yet we want to be able to consider that the database implicitly defines each IC as a total function from S into E. How are we to interpret the database, that is, what functions are we to assume that the ICs represent?

Definition. Any assumption which extends a mapping from a finite set of moments $\{S_1, S_2, \ldots, S_n\}$ (ordered as in the sequence $\langle S_1, S_2, \ldots, S_n\rangle$) into a set of individuals, into a mapping from all moments in the closed, dense interval $[S_1, S_n]$, into that set of individuals, will in general be called a *Continuity Assumption.*

We have looked at a number of different proposals for interpolating these role functions in the database, but for the sake of this exposition we will only discuss the following simple assumption. For all role attributes that record nonnumeric data (e.g., MGR, DEPT), and for some that record numeric data (e.g., SAL), it is clear that the IC intended by the discrete points recorded in the database in

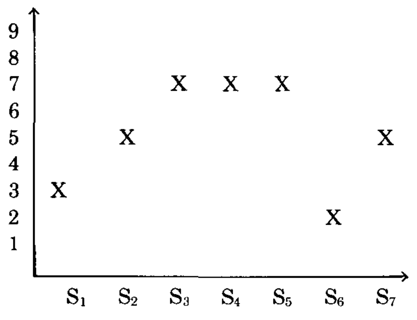


Fig. 14. Discrete points in database.

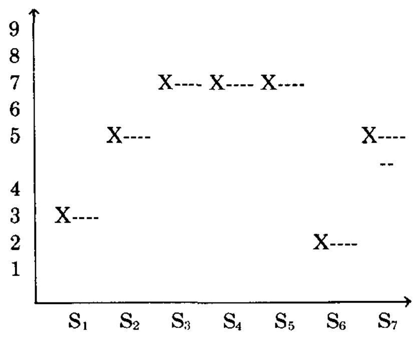


Fig. 15. Step-function.

Figure 14 is the step-function shown in Figure 15. In other words, under the *step-function* continuity assumption, the value of an IC for any state s within the database cube is given by the value of the function recorded in the database at the greatest state s' less than or equal to s. We assume that the HDB initially records information about an object X when it becomes of interest to the enterprise, say at state s_i. We then assume that a new tuple for X is added to the database at some subsequent state $s_j > s_i$ when and only when one or more of its A_i-ICs has changed value, or when it ceases to be an object of interest to the enterprise (EXISTS? becomes 0). In the interest of keeping our initial model simple, we will commit ourselves here to this view of the temporal semantics of an HDB. That is, for the remainder of this work we assume that all role attributes model step-functions.

We proceed now to formalize these notions, defining a database scheme HDB, a database hdb, and the model M induced by such a scheme and database.

Definition. A *historical relational database scheme HDB* is an ordered 8-tuple $\langle U, D, R, S, \mathrm{TV}, \mathrm{DOM}, <_D, f\rangle$, where:

1. $U = \{A_1, A_2, \ldots, A_m\}$ is a nonempty set, the set of attributes;
2. $D = \{D_1, D_2, \ldots, D_n\}$ is a nonempty set, the set of domains, such that each $D_i \in D$ is itself a nonempty set;
3. $R = \{R_1, R_2, \ldots, R_p\}$ is a set of historical-entity and historical-relationship

relation schemes, where each $R_i \in R$ is an ordered pair $\langle A_i, K_i \rangle$, such that

$$\bigcup_{i=1}^{p} A_i = U;$$

4. S is any nonempty set, the set of states;
5. TV is the set of truth values (we consider in this paper only the case of TV $= \{0, 1\}$);
6. DOM: $\{\text{STATE, EXISTS?}\} \cup U \rightarrow D \cup \{S, \text{TV}\}$ is a function that assigns to each attribute its corresponding domain, subject to the restriction that DOM(STATE) $= S$ and DOM(EXISTS?) $=$ TV;
7. $<_D$ is a partial ordering, possibly empty, on D;
8. $f: S \rightarrow R$ is an injective function that assigns to each state s a real number; it can thus be looked at as an embedding of the set of states onto the real-time line.

Given such a scheme, we define the following linear ordering on S consistent with the ordering of the image of S under f, that is, as reals.

$$<_S = \{\langle s_i, s_j \rangle \mid s_i, s_j \in S \quad \text{and} \quad f(s_i) < f(s_j)\}.$$

Definition. A *historical relational database hdb on scheme HDB* is a set of relations, hdb $= \{r_1, r_2, \ldots, r_p\}$ such that hdb is a set of completed historical-entity and historical-relationship relations such that for each relation scheme $R_i = \langle A_i, K_i \rangle$ in HDB, r_i is a relation on R_i that satisfies the appropriate historical entity-relationship constraints.

Given such a database hdb we can define the following concepts that pertain to its temporal dimension.

Definition. EDS_{hdb}, the set of explicitly defined states of the database hdb, is that finite set of states $\{s_1, s_2, \ldots, s_n\}$ given as follows.

$$\text{EDS}_{\text{hdb}} = \bigcup_{i=1}^{p} \Pi_{\text{STATE}}(r_i).$$

Note that, because each relation $r_i \in$ hdb is completed, we have

$$\Pi_{\text{STATE}}(r_i) = \text{EDS}_{\text{hdb}}.$$

Definition. The *initial state* SI_{hdb} of hdb is given as

$$\text{SI}_{\text{hdb}} = \text{the minimum element of } \text{EDS}_{\text{hdb}} \text{ under } <_S.$$

Definition. The *final state* SF_{hdb} of hdb is given as

$$\text{SF}_{\text{hdb}} = \text{the maximum element of } \text{EDS}_{\text{hdb}} \text{ under } <_S.$$

(Note that SI_{hdb} and SF_{hdb} are unique because $<_S$ is a linear ordering.)

Definition. The *database real time interval* RTI_{hdb} is the set of all times for which, under the comprehension principle, the database hdb is assumed to have complete information. This can be simply defined as follows.

$$\text{RTI}_{\text{hdb}} = [f(\text{SI}_{\text{hdb}}), f(\text{SF}_{\text{hdb}})].$$

The RTI should be at least this big; under different assumptions about the future and the past it might be defined as a larger interval. For instance, the upper endpoint might reasonably be taken from a real-time clock to represent the moment "now."

The step-function continuity assumption tells us that if we want to know whether an object exists in a state s, or what the value of any IC is in a state s, we should look at the information in the database for that state s' which is the latest state no later than s that is specifically mentioned in the database. We make this notion precise in the following definition.

Definition. The *database representative of a state s*, $[s]$, is defined as follows.

$$[s] = \left|\begin{array}{l} \text{if } s \in \mathrm{RTI}, \text{ then} \\ \quad \text{the largest state } s' \in \mathrm{EDS}_{\mathrm{hdb}} \text{ such that } s' \leq s \\ \text{if } s \notin \mathrm{RTI}, \text{ then } s \end{array}\right| .$$

Under this definition, we assume that no objects exist or have any role-ICs outside of the real-time interval $\mathrm{RTI}_{\mathrm{hdb}}$. If we wanted to model $\mathrm{SI}_{\mathrm{hdb}}$ as the "beginning of time," and/or $\mathrm{SF}_{\mathrm{hdb}}$ as the "end of time," we could modify this definition to map all times before $\mathrm{SI}_{\mathrm{hdb}}$ to $\mathrm{SI}_{\mathrm{hdb}}$, and similarly for all times after $\mathrm{SF}_{\mathrm{hdb}}$. We now define an extended Select operator which will enable us to select the value of any attribute in any state in I, regardless of whether that state is specifically represented in the database.

Definition. We define the operation σ^*, *the historical database select*, as follows.

$$\sigma^*_{\mathrm{STATE}\,=\,s,\,A\,=\,x}(r_i) = \sigma_{\mathrm{STATE}\,=\,[s],\,A\,=\,x}(r_i)$$

for any relation $r_i \in r$, $A \in U$, and $x \in UD$. (Note that this definition gives the empty relation for any s not in RTI.)

Definition. The *model M_{hdb} induced by hdb on HDB* is an ordered 5-tuple $M_{\mathrm{hdb}} = \langle E, S, <, <_E, F\rangle$, where:

1. $E = UD \cup \{\perp\}$, that is, the set of all individuals in the domain of HDB, plus the distinguished individual $\perp$, the null individual;
2. $S = R$, that is, the set of all times is just the set of real numbers;
3. $<$ is the linear ordering on the real numbers;
4. $<_E = <_D$ (given by HDB);
5. F is a function from the set of constants C_{HDB} into objects in M_{hdb} such that $F(c_a) \in D_a$; the exact specification of F is given in the following section.

6.4 The Interpretation of the Nonlogical Constants

Definition. Let r be a relation over the scheme $R(\underline{B_1} \cdots \underline{B_n}\ C_1 \cdots C_m)$. Then the *function to C_i represented by r*, θ_{r,C_i}, is a function in

$$\left(\cdots\left(D_{Ci}\right)^{D_{Bn}}\cdots^{D_{B2}}\right)^{D_{B1}}$$

whose interpretation is given as follows.

$$\theta_{r,C_i}(\langle x_1, x_2, \ldots, x_n\rangle) = \Pi_{C_i}(\sigma_{B_1=x_1,\ldots,B_n=x_n}(r)),$$

where

$$x_i \in \mathrm{LD}(K_i).$$

In other words, we say that for each non-key attribute C_i, the relation r represents a total function θ_{r,C_i} from the domain of $\langle \mathrm{LD}(B_1), \ldots, \mathrm{LD}(B_n) \rangle$ n-tuples to $\mathrm{LD}(C_i)$. If a tuple with a given $\langle \mathrm{LD}(B_1), \ldots, \mathrm{LD}(B_n) \rangle$ – value X appears in the relation r, then the value of the function θ_{r,C_i} for X is just the value of C_i associated with X in r. Otherwise, by our assumptions on the interpretation of the database, the value is bottom in $\mathrm{LD}(C_i)$.

We now define precisely what we mean by saying that a given instance hdb on a scheme HDB *induces* the definition of F_{hdb}, the function from the set of constants of our IL_s language to the function spaces in our model. We discuss in turn the interpretation of the six sorts of constants we introduced in the previous section as being induced by a particular HDB scheme: DVCs, TCs, EECs, RECs, RCs, and ACs.

6.4.1 *Interpretation of DVCs.* For any DVC $d' \in C_e$, $F_{\mathrm{hdb}}(d') = d \in UD$.

6.4.2 *Interpretation of TCs.* For any constant $c \in C_s$, $F(c) = f(c)$, that is, the interpretation given by the embedding of the states in the real numbers. For any set-of-states constant $c \in C_{\langle s,t \rangle}$, we insist that $F(c)$ defines an interval of time.

6.4.3 *Interpretation of EECs.* Let r be a historical entity relation on scheme $R(\underline{\mathrm{STATE}\ K_1}\ \mathrm{EXISTS?}\ A_1 \cdots A_n)$. Then $F_{\mathrm{hdb}}(K'_{1_*})$, the interpretation of the EEC K'_{1_*}, of type $\langle s, \langle e, t \rangle \rangle$, is that function f in $\langle s, \langle e, t \rangle \rangle$ whose value for any state $s \in S$ and individual $x \in E$ is given as follows.

$$f(s, x) = \Pi_{\mathrm{EXISTS?}}(\sigma_{\mathrm{STATE}\,=\,s,\ K_1\,=\,x}(r)).$$

Thus, under our interpretation of the historical database the only K_1-entities that exist are those that the database historical entity relation r with entity key K_1 says exist.

As an example, let us consider the interpretation of the constant EMP'_*, and evaluate it for some elements in its domain.

Example 1. "Is Peter an employee in state S_1?"

$$\begin{array}{llllllll}
f(\langle S_1, \mathrm{Peter}\rangle) & & & & & & & \\
\quad = \Pi_{\mathrm{EXISTS?}}(\sigma^*_{\mathrm{STATE}=S_1,\mathrm{EMP}=\mathrm{PETER}}(\mathrm{emp\_rel})) & & & & & & & \\
 & (\mathrm{STATE} & \mathrm{EMP} & \mathrm{EXISTS?} & \mathrm{MGR} & \mathrm{DEPT} & \mathrm{SAL}) \\
\quad = \Pi_{\mathrm{EXISTS?}} & S_1 & \mathrm{Peter} & 0 & \perp & \perp & \perp \\
\quad = 0. & & & & & &
\end{array}$$

Thus, Peter is not an employee in S_1.

Example 2. "Is Liz an employee in state S_3?"

$$\begin{array}{lllllll}
f(\langle S_3, \mathrm{Liz}\rangle) & & & & & & \\
\quad = \Pi_{\mathrm{EXISTS?}}(\sigma^*_{\mathrm{STATE}=S_3,\mathrm{EMP}=\mathrm{Liz}}(\mathrm{emp\_rel})) & & & & & & \\
 & (\mathrm{STATE} & \mathrm{EMP} & \mathrm{EXISTS?} & \mathrm{MGR} & \mathrm{DEPT} & \mathrm{SAL}) \\
\quad = \Pi_{\mathrm{EXISTS?}} & S_3 & \mathrm{Liz} & 1 & E_2 & D_2 & 50 \\
\quad = 1. & & & & & &
\end{array}$$

Thus, Liz is an employee in S_3.

Example 3. "Is entity 50 an employee in state S_3?"

$$\begin{aligned}
&f(\langle S_3, 50\rangle)\\
&\quad = \Pi_{\text{EXISTS?}}(\sigma^*_{\text{STATE}=S_3,\text{EMP}=50}(\text{emp\_rel}))\\
&\quad = \Pi_{\text{EXISTS?}}\begin{pmatrix}\text{STATE} & \text{EMP} & \text{EXISTS?} & \text{MGR} & \text{DEPT} & \text{SAL}\\ & & \varnothing & & & \end{pmatrix}\\
&\quad = 0.
\end{aligned}$$

Thus, 50 is not an employee in S_3.

6.4.4 *Interpretation of RECs.* Unlike the case of EECs, in which a single historical entity relation r over a scheme R represented all of the information about the existence of entities of a given sort in the database, in the case of RECs there may be any number of historical relationship relations of a given arity that must together be considered to determine which n-ary relationships exist. Our definition of the interpretation of the constants REL-n, therefore, must be given in terms of the entire database and not just of a single relation.

Let n-rels $= \{r_1, \ldots, r_k\}$ be the set of all the n-ary historical relationship relations in the database, that is, all relations in the database over schemes R_i of the form $R_i(\underline{\text{STATE}}\ \underline{K_{i_1}} \cdots \underline{K_{i_n}}\ \text{EXISTS?}\ A_{i_1} \cdots A_{i_m})$. Since these k relations are all defined over the same logical domains for the set of attributes $\{\text{STATE}, K_{i_1}, \cdots, K_{i_n}, \text{EXISTS?}\}$, we can conceptually take the union of these k projections considered as relations over these logical domains. (Notice that $\text{LD}(K_{i_j}) = E$ for all K_{i_j}.) In order to do this we define a new relation r over the scheme $R(\underline{\text{STATE}}\ \underline{E_1} \cdots \underline{E_n}\ \text{EXISTS?})$, where r is the union of these k relations over these "common" attributes:

$$r = \bigcup_{i=1}^{k} \Pi_{\text{STATE},K_{i_1},\ldots,K_{i_n},\text{EXISTS?}}(r_i).$$

Then the interpretation of the constant REL-n induced by the database, $F_{\text{hdb}}(\text{REL-}n)$, is that function f in $\langle s, \langle e^n, t\rangle\rangle$ whose value for any state $s \in S$ and n-tuple $\langle x_1, x_2, \ldots, x_n\rangle \in E^n$ is given as:

$$f(s, \langle x_1, x_2, \ldots, x_n\rangle) = \Pi_{\text{EXISTS?}}(\sigma^*_{\text{STATE}=s,\langle K_1,\ldots,K_n\rangle=\langle x_1,\ldots,x_n\rangle}(r)),$$

which is completely analogous to our definition of the interpretation of the EECs, or 1-ary relationships.

For example, in our department-store database the only binary relationship is the one between DEPTs and ITEMs. We evaluate f for various 2-tuples in various states.

Example 4. "Is there a relationship between the Hardware Department and Item Glove in state S_3?"

$$\begin{aligned}
&f(\langle S_3, \text{Hardware}, \text{Glove}\rangle)\\
&\quad = \Pi_{\text{EXISTS?}}(\sigma^*_{\text{STATE}=S_3,\text{DEPT}=\text{Hardware},\text{ITEM}=\text{Glove}}(\text{sales\_rel}))\\
&\quad = \Pi_{\text{EXISTS?}}\begin{pmatrix}\text{STATE} & \text{DEPT} & \text{ITEM} & \text{EXISTS?} & \text{VOL}\\ S_3 & \text{Hardware} & \text{Glove} & 0 & \perp\end{pmatrix}\\
&\quad = 0
\end{aligned}$$

Thus, the relationship Hardware-Glove does not exist in S_3.

Example 5. "Is there a relationship between the Toy Department and Item Game in state S_1?"

$$f(\langle S_1, \text{Toy, Game}\rangle)$$
$$= \Pi_{\text{EXISTS?}}\,(\sigma^*_{\text{STATE} = S_1,\ \text{DEPT} = \text{Toy},\ \text{ITEM} = \text{Game}}(\text{sales\_rel}))$$
$$= \Pi_{\text{EXISTS?}} \begin{array}{ccccc} (\text{STATE} & \text{DEPT} & \text{ITEM} & \text{EXISTS?} & \text{VOL}) \\ S_1 & \text{Toy} & \text{Game} & 1 & 6 \end{array}$$
$$= 1.$$

Thus, the relationship Toy-Game does exist in S_3.

Example 6. "Is there a relationship between the Toy Department and Peter in State S_1?"

$$f(\langle S_1, \text{Toy, Peter}\rangle)$$
$$= \Pi_{\text{EXISTS?}}(\sigma^*_{\text{STATE} = S_1,\ \text{DEPT} = \text{Toy},\ \text{ITEM} = \text{Peter}}\ (\text{sales\_rel}))$$
$$= \Pi_{\text{EXISTS?}} \begin{array}{ccccc} (\text{STATE} & \text{DEPT} & \text{ITEM} & \text{EXISTS?} & \text{VOL}) \\ & & \varnothing & & \end{array}$$
$$= 0.$$

Thus the relationship Toy-Peter does not exist in S_3.

In order to define the interpretation of the remaining two kinds of nonlogical constants that we have defined, we again need a preliminary definition, in this case to handle the role-attribute ICs.

Definition. Let r be a historical relation on scheme $R(\underline{\text{STATE}}\ \underline{K_1} \cdots \underline{K_n}$ EXISTS? $A_1 \cdots A_m)$, and let X be a $\langle K_1 \cdots K_n\rangle$-value, that is, $X \in E^n$. Then the *A_1-IC associated with the object X in r on R*, $F1_{A_i,X,r,R}$, is that function of type $\langle s, e\rangle$ whose interpretation induced by r is given as follows:

$$F1_{A_i,X,r,R}(s) = \Pi_{A_i}(\sigma^*_{\text{STATE} = s,\ \langle K_1 \cdots K_n\rangle = X}(r)).$$

Definition. The *set of A_i-ICs associated with the object X in r on R in state s*, $F2_{A_i,X,r,R}$, is a function of type $\langle s, \langle\langle s, e\rangle, t\rangle\rangle$, whose interpretation induced by r is given as follows:

$$F2_{A_i,X,r,R}(s) = \left|\begin{array}{ll} \{F1_{A_i,X,r,R}\} & \text{if} \quad \Pi_{\text{EXISTS?}}(\sigma^*_{\text{STATE} = s,\ \langle K_1 \cdots K_n\rangle = X}(r)) \neq 0 \\ \varnothing & \text{otherwise} \end{array}\right|.$$

In other words, in any state s we associate an object X with its role-attribute ICs only if the object X exists in state s, otherwise it is not associated with any ICs. (Note that in any state the set given by F_2 is either a singleton set—containing one IC—or the empty set.) This definition enables us to simplify the types of many of our constants (as compared to Montague's treatment in PTQ [35]), while at the same time avoiding assigning a role to any IC associated with an object that is nonexistent is a given state.

6.4.5 *Interpretation of RCs.* Let r be a historical relation on scheme $R(\underline{\text{STATE}}\ \underline{K_1} \cdots \underline{K_n}$ EXISTS? $A_1 \cdots A_m)$. Then the interpretation induced by r of the RC A_i' is simply the union of all of the sets of A_i-ICs associated with any objects X. In other words, $F_{\text{hdb}}(A_i')$ is that function f in $\langle s, \langle\langle s, e\rangle, t\rangle\rangle$ whose value for any state $s \in S$ is given as follows.

$$f(s) = \bigcup_{X \text{ in } E^n} F_{2_{\ldots}}(s).$$

For example, SAL′ for any state s denotes the set of all ICs which are the salary-selecting functions of any employee.

6.4.6 *Interpretation of ACs.* As in the case of the REL-n's, for any given n we use a single nonlogical constant to represent information about all objects of arity n, information that may be located in an arbitrary number of database relations. An AC AS-n represents the association between any object of arity n and each of its role ICs. We must therefore define the interpretation of these constants in terms of the entire database and not just of a single relation. We would like to take all of the functions given by F_2, that is, the set of all of the ICs associated with any object X, and *merge* them all together to yield a single function which, for any object X, gives all of the role ICs associated with X. In order to do this, we need to make this notion of merging precise.

Definition. We say that a relation r on $R = \langle A, K\rangle$ is *defined* for the object $x \in K$ if $x \in \Pi_K(r)$.

As before we let n-rels $= \{r_1, \ldots, r_k\}$ be the set of all relations r_i in the database over schemes R_i of the form R_i ($\underline{\text{STATE}}\ \underline{K}_{1_i} \cdots \underline{K}_{n_i}$ EXISTS? $A_{1_i} \cdots A_{m_i}$). By the historical entity-relationship constraint (2), an entity X can belong to only one entity set, and by constraint (6) only one relationship can exist for any set of entity sets. Together, these constraints mean that any n-ary object $\langle x_1, x_2, \ldots, x_n\rangle$ is defined by at most one relation in hdb, that is

$$\Pi_{\langle K_{1_i}, \ldots, K_{n_i}}(r_i) \cap \Pi_{\langle K_{1_j} \ldots K_{n_j}\rangle}(r_j) = \varnothing,$$

for any two distinct relations r_i, $r_j \in n$-rels. From this, it follows that for any $X \in E^n$ and any role attribute A, the function $F_{1_{A_i, X, r_i, R_i}}$ is defined for at most one $r_i \in n$-rels; this is thus also the case for the function $F_{2_{A_i, X, r, R}}$.

Then the interpretation of AS-A_i induced by the database is that function f in $\langle e^n, \langle\langle s, e\rangle, t\rangle\rangle$ whose value for any $X \in E^n$ is given as follows.

$$f(X) = \left| \begin{array}{ll} \bigcup_{s \in S} F_{2_{A_i, X, r, R}}(s) & \text{if for some } r \text{ in } n\text{-rels the object } X \text{ is defined in } r \\ \varnothing & \text{otherwise} \end{array} \right|$$

In other words, the interpretation of AS-A_i gives, for any n-ary object X, the set of all ICs in the role A_i associated with X in any state.

After this more formal presentation of the logical model induced by an HDB instance, it will be informative to take a look at each of the elements in the historical database in turn to see how it is reflected in the model.

6.5 Informal Discussion of IL$_s$ and HDB

6.5.1 *Domains and Values.* In the definition of the HDB, the set *UD* consists of the names of all of the individuals that may possibly be referenced in any stage of the database history. The database itself can be viewed as a collection of sentences in an implied logic, and we have just presented a translation from this language into IL_s. The domains correspond in the following way. The set of constants of type e in IL_s, C_e, is defined to be $\{d : d \in UD\}$. Correspondingly, the set E of individuals in the model for IL_s is defined to be $\{\underline{d} : d \in UD\} \cup \{\perp\}$. Moreover we have specified the interpretation of these constants in the obvious way:

$$F_{\text{hdb}}(d) = \underline{d}.$$

6.5.2 *Attributes.* As we have seen, the HDB model identifies three different kinds of attributes: the distinguished attributes STATE and EXISTS?, attributes that are keys whose values are rigid designators of entities, and role attributes which are unconstrained functions (ICs) which in any state give some property of either an entity or a relationship. Montague describes this distinction between constant and unconstrained ICs in this manner: "'Ordinary' common nouns (for example, *horse*) will denote sets of constant individual concepts (for example, the set of constant functions on worlds and moments having horses as their values; from an intuitive viewpoint, this is no different from the set of horses). It would be unacceptable to impose this condition on such 'extraordinary' common nouns as *price* or *temperature*; the individual concepts in their extensions would in the most natural cases be functions whose values vary with their temporal arguments" [35]. We have made the same claim here in the HDB realm; in particular, we have argued that key attributes (such as EMP) and role attributes (such as SAL) are to be identified with Montague's "ordinary" and "extraordinary" common nouns, respectively.

It is, of course, the attribute STATE which bears the burden of providing the temporal semantics for the HDB model. We believe that it is best to define the model in terms of a very general temporal semantics and allow the user to specify (via meaning postulates) further properties of this parameter. We have described here our step-function continuity assumption as a means of interpolating the partial function given by the historical database. The attribute EXISTS? enables objects to come in and out of focus at will as objects of interest to the enterprise. When an object is of interest, EXISTS? has the value 1 and all of the role attributes for that object are defined; otherwise, EXISTS? is 0 and it has no attributes (all are $\perp$).

6.5.3 *Tuples.* A tuple in the HDB model, as in the entity-relationship model, is viewed as a collection of facts about a single object, an entity or a relationship. In either case it has seemed more natural to us to view the association between an object and its attributes as essentially binary. The theory could easily have treated n-ary tuples as n-ary associations among the various ICs involved. With the choice of semantic primitives that we have made, a tuple in a historical relation representing an object of arity n with m role attributes is reflected in the logic IL_s by a simple sentence composed of three parts:

1. n entity existence terms and, if $n > 1$, an additional relationship existence term;
2. m terms identifying the sorts of the m role attributes;
3. m terms associating the n-ary object with each of its m attributes.

For example, the first tuple in dept-rel is completely represented in IL_s by the following formula in IL_s (assume that the variable x is of type $\langle s, e\rangle$).

$$\exists\, x[\mathrm{DEPT}'_*(S_1, \mathrm{Toy}) \wedge \mathrm{FLOOR}'(S_1, x) \wedge \mathrm{AS\ FLOOR}(\mathrm{Toy}, x) \wedge x(S_1) = F_1]$$

The first tuple in sales-rel is completely represented in IL_s by the following.

$$\exists\, x[\mathrm{DEPT}'_*(S_1, \mathrm{Toy}) \wedge \mathrm{ITEM}'_*(S_1, \mathrm{Ball}) \wedge \mathrm{VOL}'(S_1, x) \wedge \text{AS-VOL}(\mathrm{Toy}, \mathrm{Ball}, x) \wedge x(S_1) = 3]$$

6.5.4 *Data Dependencies and Constraints.* The inclusion of an explicit time component in the HDB model allows us to express the semantics of a wide class of database constraints in the same language, something not possible in a first-order logic without some extra apparatus. We divide these database constraints into two categories and make the following definitions.

Definition. An *extensional database constraint* is a constraint on individual valid states of the database. It can be said to hold (or not hold) simply on the basis of the extension of the database with respect to a single state.

Definition. An *intensional database constraint* is a constraint which defines valid state progressions in the database. It can be said to hold (or not hold) only by examining at least two states of the HDB.

Current theoretical relational database research has been primarily concerned (without itself using the term) with extensional constraints, such as FDs or MVDs. The relationship between the FDs and MVDs of the relational model and axioms expressed as formulas in a first-order logic is one which is well understood (see, e.g., [36, 37]). The FD EMP $\rightarrow$ SAL, for example, is an abbreviation for the first-order formula

$$\forall x\, \forall y\, \forall z\, [\mathrm{EMP}(x) \wedge \mathrm{SAL}(y) \wedge \mathrm{SAL}(z) \wedge \text{AS-SAL}(x, y) \wedge \text{AS-SAL}(x, z) \rightarrow y = z]$$

in the domain relational calculus (i.e., with variables having individuals as their domain), or for the formula

$$\forall t_1\, \forall t_2[t_1(\mathrm{EMP}) = t_2(\mathrm{EMP}) \rightarrow t_1(\mathrm{SAL}) = t_2(\mathrm{SAL})]$$

in the tuple relational calculus (i.e., with variables having tuples as their domain). Ullman [48] discusses these two calculi and demonstrates their equivalence. An intensional logic allows us to express more fully and easily the intent of these FDs. We can specify explicitly that they must hold over all states of the database. Moreover, we can make the more explicit statement that there is only one function (IC) that picks out a given attribute (e.g., the SALary) of any object (e.g., EMP) that has that attribute:

$$\forall x\, \forall y\, \forall z\, \forall i\, [\mathrm{EMP}'(i, x) \wedge \mathrm{SAL}'(i, y) \wedge \mathrm{SAL}'(i, z) \wedge \text{AS-SAL}(x, y) \wedge \text{AS-SAL}(x, z) \rightarrow y = z].$$

Here we have quantified over all states of the database with the state variable i (type s), and over the ICs x, y, and z, we have equated not merely the value (extension) of the two SALaries, but the SALary-ICs (functions) themselves. (We note in passing that the comparable axiom in IL_s using "tuple" variables would require a different approach than we have taken. We would have to have tuple variables of the appropriate type, since IL_s is a typed logic.) Similar intensional axioms for MVDs would constrain the acceptable models for our HDB.

Intensional constraints have not received much attention in the database literature. Where they have been examined (e.g., by Smith and Smith [46], Nicolas and Yazdanian [38], and Casanova and Bernstein [7] as "dynamic constraints" or constraints upon update operations), they have been considered as different in kind from extensional (or "static") constraints. In this paper we have shown how IL_s, as a higher order language with a temporal dimension, allows us to consider different types of objects (e.g., states, individuals, ICs, and other arbitrarily defined functions) and to make statements about any of these objects with the full power of quantified logic and lambda calculus. We can thus express both types of constraints in IL_s in the same natural way, that is, as axioms about objects (of the appropriate type), without having to invent a new technique for expressing the dynamic constraints.

Consider the following kind of constraint that might hold in an enterprise keeping a relation on EMP-REL.

No employee can ever be given a cut in pay.

This is an intensional constraint. It constrains the kind of function that can serve as a SAL-IC for any EMPloyee, in particular to those functions from states to dollar values that have everywhere a nonnegative derivative. It is not expressible as a first-order database axiom because it does not refer simply to the extension of the SALary function in any one state, but rather to the entire function considered as an intensional object, namely, an IC. In IL_s this constraint is expressible as:

$$\forall i_1 \forall u \forall x [\text{EXP}'_*(i_1, u) \wedge \text{SAL}'(i_1, x) \wedge \text{AS-SAL}(u, x) \\ \rightarrow \forall i_2[i_1 < i_2 \rightarrow x(i_1) \leq x(i_2)]].$$

Where u is a variable over individuals. This ability to consider both intensional and extensional constraints as essentially the same kind of constraint, and to express them in the same language, is a good example of the power that an intensional logic has to provide a unified theory of database semantics. In the section on queries which follows, we give examples of the "definition" in IL_s of English words such as "rehire" (an EMPloyee), "raise" (a SALary), and "transfer" (a DEPT assignment), definitions which use the same concept of explicit quantification over states of the HDB.

6.5.5 *Queries.* As with database constraints, the inclusion of the state component in the historical database model allows us to consider a much broader class of database queries in a consistent manner. We are similarly motivated, therefore, to make the following distinction. An *extensional database query* is a query whose evaluation depends only on the values in the database with respect to a single index or state. An *intensional database query* is a query whose evaluation depends on the intensions of at least one attribute, that is, on the function from states to individuals (ICs) that represents that attribute.

It should be apparent that extensional queries are precisely those that static databases have been concerned with handling, and moreover that these queries are handled just as well by a historical database. We note, however, that since the HDB contains, as it were, many static databases indexed by state, it is possible to ask the same extensional queries with respect to different states and thus to get potentially different answers. For example, the answer to "What is Peter's salary?" with respect to state S_2 is "30K," but with respect to state S_3 what appears to be the same query of the same database yields the equally correct (but different) answer "35K." Thus we see that in order to utilize the power of the HDB, extensional queries must be more fully specified to indicate the state at which evaluation is to be performed. In [11], this process is explained more fully, and the concept of a variable *now*, whose interpretation is always the latest state of the HDB, is discussed.

It is the class of intensional queries in which we are more interested because these queries utilize the full power of the HDB and show it to be a much closer model of the real world than a one-dimensional static database. We suggest that within the context of an HDB we have the potential to answer all of the queries which were mentioned at the beginning of Section 3. We repeat them here.

"Has John's salary risen?"
"When was Peter rehired?"
"Did Rachel work for the toy department last year?"
"Has John ever earned the same salary as Peter?"
"Will the average salary in the linen department surpass $30,000 within the next five years?"

How, for instance, might we handle the query "Has John's salary risen?" Let us assume a mechanism for translating this query into the following formula in IL_s

$$\exists\, x[\mathrm{SAL}'(\mathrm{now}, x) \wedge \mathrm{EMP}'_*(\mathrm{now}, \mathrm{John}) \wedge \text{AS-SAL}(\mathrm{John}, x) \wedge \mathrm{RISE}'(\mathrm{now}, x)]$$

In order to evaluate this query, we need some mechanism for providing a meaning to the predicate RISE′. There are two ways that we could do this: either by providing the denotation of RISE′ via a direct translation from the database, analogous to the way we defined our primitives (such as SAL′), or by providing its denotation indirectly, essentially making RISE′ a predicate whose meaning is derived from the denotations of the basic predicates induced by the database. This is the course we shall take, as the former method is impractical—it would have to be updated with each database update. Before we can provide any definition, we must, of course, decide upon an appropriate meaning for the English word "rise." We suggest the following: RISE′ is true of a SALary IC at a given state i if and only if there is a preceding interval of time culminating in state i during which the SAL-IC has an everywhere nonnegative derivative (or, equivalently, is monotonically nondecreasing). Of course we could quibble about this definition for a while, but that is not the point. The point is that given any such well-defined semantics for the word, we could express its meaning in IL_s. The suggested definition translates into the IL_s meaning postulate:

$$\forall\, x\, \forall\, i\, [\mathrm{RISE}'(i, x) \leftrightarrow [\mathrm{SAL}'(i, x) \wedge \exists\, i_1\, \forall\, i_2\, \forall\, i_3 \\ [i_1 \le i_2 < i_3 \le i \rightarrow x(i_2) \le x(i_3)]]].$$

We hasten to point out that there is nothing sacred in this definition about the attribute SAL. In the context of other attributes that in English might meaning-fully be said to "rise," (e.g., the BALance of a bank account, the BATting-AVErage of a baseball player), the above meaning postulate could easily be generalized.

Given this MP, we evaluate the predicate RISE′(i, x) as follows. From emp_rel we see that the SAL-IC associated with John is an IC whose value for the three known states is as follows.

$$\begin{bmatrix} S_1 \rightarrow \bot \\ S_2 \rightarrow 30 \\ S_3 \rightarrow 35 \end{bmatrix}$$

The value for all other states is $\bot$. Let us call this function S_J. Then RISE′(i, S_J) evaluated for $i = S_3$ is true (pick S_2 as the i_1 which the MP asserts must exist).

As another example, we could define the English verb "rehire" as follows.

$$\forall u \, \forall i \, [\text{REHIRE}(i, x) \leftrightarrow [\text{EXP}'_*(i, u) \wedge \exists i_1 \exists i_2$$
$$[i_1 < i_2 < i \wedge \text{EMP}'_*(i_1, u) \wedge \neg \, \text{EMP}'_*(i_2, u)]]].$$

That is, it is true at state i that the individual u has been rehired if u is an EMPloyee at time i, and at some earlier time i_1 was also an EMP, while at some third time i_2 between i and i_1, u was not an EMP.

7. SUMMARY AND FUTURE WORK

In this paper we have espoused the overall philosophy that formal logic has made and can continue to make important contributions to the understanding and specification of the semantics of databases. The choice of the logic IL$_s$ has been motivated in this paper by the fact that it incorporates a temporal semantics that formalizes the concept of a historical database. In [11], this choice is also motivated from the perspective of providing a formal definition of an English query language as a Montague grammar (MG).

Specifically, we have shown how the relational database model can be easily extended to incorporate the concept of *historical relations* and, indeed, an entire *historical database*, and we have shown how IL$_s$ can provide a semantic theory for this database concept. We have presented both an informal discussion of an HDB as a cube composed of a time-ordered sequence of flat, static relations, and a formal description of the relationship between an HDB and the logic IL$_s$ and its model theory. Finally, we have given examples of the power of the historical database to model real-world semantics more closely than existing database models. Two such examples were emphasized: the ability to express the semantics of intensional and extensional database constraints within the same theory, and the ability to process intensional and extensional queries.

We believe that the HDB concept is exciting precisely because it suggests the possibility of formalizing a wide variety of database-semantic issues "under one roof," namely, within the precise model-theoretic semantics of IL$_s$. We mention a number of these issues here.

Montague's English fragment PTQ [35] is provided with a formal semantics indirectly, by means of rules for translating expressions in the fragment into IL, for which a direct model-theoretic semantics is given. In [11], we present a technique for describing an English query fragment for relational database querying in English, which draws upon Montague's work and work done subsequently by other researchers in logic and linguistic theory within the framework of MG, especially Bennett [1, 2], Kartunnen [25], and Dowty [15]. In this paper, we provide a semantics for English questions which takes advantage of the simplification of real-world semantics inherent in a database, yet which is powerful enough to interpret a useful class of database queries correctly.

The first question that this paper will suggest to many readers is that of implementation; database theories are almost inevitably, and quite properly, judged by their practicality. Obviously the picture of each historical relation as a fully specified cube is an idealization. Even if all of the information in the cube were known, a direct implementation would be highly redundant. Furthermore, there may be situations in which the complete history of some attributes may be unknown or uninteresting to the enterprise. Questions of how to implement these relations efficiently both for storage and for retrieval, and of how to handle a mixture of static and historical relations within a single database, are among the many interesting implementation questions that remain to be studied.

Another area of interest, suggested by our work in defining the translation of English questions into IL_s, is the possibility of interpreting English statements as database commands. For example, we could interpret the statement "John earns 30K," when made by an authorized user, as a command to record this as a fact in the database with the time-stamp taken from the system clock. As with questions, intensional logic gives us a framework for providing a formal semantics for an appropriate fragment of English to serve as a DML to perform such database maintenance operations as insertion and deletion. Consideration would have to be given to the semantics of error-correction types of maintenance, that is, the sorts of commands which mean not that a given fact once true about the world no longer obtains, but rather that a previous specification of that "fact" was in error. How do such changes, ignored in this paper, affect the model-theoretic semantics? Since an update in general represents only partial information about a state, can we make certain assumptions that will help to further specify that state? (For example, if Peter's SALary is respecified, can we assume that his DEPT remains the same?)

We have incorporated the work presented in this paper into the relational database model, constrained by the view of data semantics presented by the entity-relationship model. The question of how to extend other database models, such as the hierarchical [24], network [12], and functional [45] models, to include a temporal semantics is another area for future study. Even within the relational model, the question of other semantic restrictions on the kinds of relations that make sense, within the context of a formalized temporal semantics, is still wide open for future study.

The idea of using a database to model hypothetical situations as potential futures from a given present situation, and thus providing the ability to answer queries about the implications of such "possible worlds," is another expansion of the HDB concept that appears to offer promising applications. The query suggested earlier in this paper, "Will the average salary in the linen department surpass $30,000 within the next five years?" is the sort of question that we believe could be handled by such an organization. Salary raises built into union contracts, *cost-of-living increases, projections in costs based on the expected inflation rate,* and so on, are the sorts of applications that a historical database ought to be able to model. Stonebraker and Keller [47] provide an examination of some of these possibilities from a different perspective.

In the simple model we have presented here, EXISTence is synonymous with belonging to an entity set, and we have not allowed an entity to be of more than one sort. We have begun investigating an extension to this model that would allow entities to fill different (and even multiple) roles at various times, as long as they still EXISTed as entities in some relation. For example, we could model people with a relation on scheme PERSON_REL(NAME STATE EXISTS? GENDER ⋯), and then have relations on schemes such as BORROWER_REL(NAME STATE IS_BORROWER? ACCT_# ⋯) and DEPOSITOR_REL(NAME STATE IS_DEPOSITOR? ACCT_BAL ⋯). People could fill the roles of depositor and/or borrower in any state at will, indicated by the Boolean-valued IS_⟨ROLE⟩? attribute, provided they were said to EXIST in that state in the PERSON relation. Meaning postulates could assert the IS-A hierarchy (BORROWER IS-A PERSON, etc.), and with what appear at this point to be minor changes in our scheme for encoding a database into a logical model the present HDB approach seems to work, and to offer interesting insights into the semantics of this sort of database model.

Another important area for future work is the nature of the time coordinate in the HDB model, and the kinds of constraints that particular applications may wish to make upon the general treatment we have defined. Allowing more sophisticated continuity assumptions, different assumptions for different attributes, modifying the continuity principle, conceiving of time not as moments but as partitioned into intervals, and so on, are among the many issues relating to the temporal semantics that remain to be addressed.

Finally, we note that the last few years have seen a number of researchers, among them Schmid and Swenson [43], Hammer and McLeod [21], Maier and Warren [31], and Biller and Neuhold [3], discuss the need for more powerful database models or languages in order to specify a database semantics that more closely models the real world. We agree entirely with this overall goal, but view with some apprehension the proliferation of so-called semantic data definition languages (SDDLs) that are not provided with a formal semantics. While we would not say that IL_s is the only solution for a clear database semantics, we do believe strongly that an intensional logic such as IL_s can serve as a much-needed *lingua franca* in which to compare these higher level semantic models and languages, and even to provide a basis for constructing proofs that demonstrate their equivalence or differences.

An analogous situation is occurring in the field of artificial intelligence, which is witnessing the same proliferation of knowledge representation languages (KRLs): frames [34], KRL [4], PROLOG [27], and RLL [20], to name only a few. Considerable discussion and often heated arguments have ensued over which language is better. Hayes [22] expressed much the same sentiment that we have presented here, arguing that logic can serve as a universal tool for clarity and comparison.

Naturally, until the semantics of natural languages is more completely understood, artificial languages such as these SDDLs, which are clearly more useroriented than the equally artificial language IL_s, are the appropriate kind of vehicle for users to express their database semantics. But we believe that unless these languages are provided with a formal model-theoretic semantics, there will be no basis for making informed judgments about the expressive power of these languages as a whole, or about the accuracy (or even the precise meaning) of particular statements in these languages.

ACKNOWLEDGMENTS

The authors would like to thank Albert Croker, Paul K. Harter, David Maier, and Ed Sciore for many helpful discussions that contributed to clarifying our ideas and their presentation herein, and the anonymous referees for suggesting many improvements to a previous version of this paper.

REFERENCES

1. BENNETT, M.R. Some extensions of a Montague fragment of English. Ph.d dissertation, University of California at Los Angeles, 1974.
2. BENNETT, M.R. Questions in Montague grammar. Indiana University Linguistics Club, Bloomington, Indiana, 1979.
3. BILLER, H., AND NEUHOLD, E.J. Semantics of data bases: The semantics of data models. *Inf. Syst. 3*, 1 (1978), 11-36.
4. BOBROW, D.G., AND WINOGRAD, T. An overview of KRL—A knowledge representation language. *Cognitive Science 1*, 1 (1977), 3-46.
5. BUBENKO, J.A., JR. The temporal dimension in information modelling. In *Architecture and Models in Data Base Management Systems*, G.M. Nijssen (Ed.), North Holland, Amsterdam, 1977.
6. CARNAP, R. *Meaning and Necessity*. University of Chicago Press, Chicago, 1947.
7. CASANOVA, M.A., AND BERNSTEIN, P.A. The logic of a relational data manipulation language. In *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages* (San Antonio, Texas, January 29-31, 1979), ACM, New York, pp. 101-109.
8. CHANG, C.L. DEDUCE 2: Further investigations of deduction in relational data bases. In *Logic and Data Bases*, H. Gallaire and J. Minker (Eds.), Plenum Press, New York, 1978.
9. CHEN, P.P.-S. The entity-relationship model—Toward a unified view of data. *ACM Trans. Database Syst. 1*, 1 (March 1976), 9-36.
10. CHURCH, A.L. *The Calculi of Lambda-Conversion*. Princeton University Press, Princeton, N.J., 1941.
11. CLIFFORD, J. A logical framework for the temporal semantics and natural-language querying of historical databases. Ph.d dissertation, S.U.N.Y. at Stony Brook, 1982.
12. CODASYL DATA BASE TASK GROUP. CODASYL data base task group report. ACM, New York, 1971.
13. CODD, E.F. A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (June 1970), 377-387.
14. DOWTY, D.R., WALL, R.E., AND PETERS, S. *Introduction to Montague Semantics*. D. Reidel, Dordrecht, Germany, 1981.
15. DOWTY, D.R. *Word Meaning and Montague Grammar*. D. Reidel, Dordrecht, Germany, 1979.
16. FREGE, G. On sense and reference. In *Translations from the Philosophical Writings of Gottlob Frege*, P. Geach and M. Black (Eds.), Basil Blackwell Publisher, 1952.
17. GALLAIRE, H., AND MINKER, J. *Logic and Data Bases*. Plenum Press, New York, 1978.
18. GALLIN, D. *Intensional and Higher-Order Modal Logic*. North-Holland, Amsterdam, 1975.
19. GOLDSTEIN, B.S. Constraints on null values in relational databases. Tech. Rep. 80/015, Dep. Computer Science, State University of New York at Stony Brook, 1981.
20. GREINER, R., AND LENAT, D.B. A representation language language. In *Proc. 1st Annual National Conference on Artificial Intelligence*, Stanford Ca. American Ass. for Artificial Intelligence, Menlo Park, Ca.
21. HAMMER, M., AND MCLEOD, D. The semantic data model: A modelling mechanism for data

base applications. In *Proc. ACM SIGMOD International Conference on Management of Data* (Austin, Texas, May 31–June 2, 1978), ACM, New York, pp. 26–36.

22. HAYES, P.J. In defence of logic. In *Proc. 5th International Joint Conference on Artif. Intell.*, Cambridge.
23. HOBBS, J.R., AND ROSENSCHEIN, S.J. Making computational sense of Montague's intensional logic. *Artif. Intell 9*, 3 (1977), 287–306.
24. IBM. *IMS/360—Application Description Manual.* GH-20-0765, IBM, White Plains, New York.
25. KARTUNNEN, L. Syntax and semantics of questions. *Linguistics and Philosophy,* 1 (1977), 3–44.
26. KLOPPROGGE, M.R. TERM: An approach to include the time dimension in the entity-relationship model. In *Proc. 2nd International Conference on Entity-Relationship Approach,* Washington, D.C.
27. KOWALSKI, R. *Logic for Problem Solving.* Elsevier, North-Holland, New York, 1979.
28. LAINE, H., MAANAVILJA, O., AND PELTONA, E. Grammatical data base model. *Inf. Syst. 4*, 4 (1979).
29. MAIER, D. *The Theory of Relational Databases.* Computer Science Press, Potomac, Md., 1983.
30. MAIER, D., AND WARREN, D.S. A theory of computed relations. Tech. Rep. 80/012, Dep. Computer Science, S.U.N.Y. at Stony Brook, 1980.
31. MAIER, D., AND WARREN, D.S. Specifying connections for a universal relation scheme database. In *Proc. ACM SIGMOD International Conference on Management of Data* (Orlando, Florida, June 2–4, 1982), ACM, New York, pp. 1–7.
32. McCARTHY, J., ABRAHAMS, P.W., EDWARDS, D.J., HART, T. P., AND LEVIN, M.I. *LISP 1.5 Programmer's Manual.* MIT Press, Cambridge, 1962.
33. MINKER, J. An experimental relational data base system based on logic. In *Logic and Data Bases,* H. Gallaire and J. Minker (Eds.), Plenum Press, New York, 1978.
34. MINSKY, M. A framework for representation knowledge. In *The Psychology of Computer Vision,* P.H. Winston (Ed.), New York, 1975.
35. MONTAGUE, R. The proper treatment of quantification in ordinary English. In *Approaches to Natural Language,* K.J.J. Hintikka (Ed.), Dordrecht, Germany, 1973.
36. NICOLAS, J.M. First order logic formalization for functional, multivalued, and mutual dependencies. In *Proc. ACM SIGMOD International Conference on Management of Data* (Austin, Texas, May 31–June 2, 1978), ACM, New York, pp. 41–46.
37. NICOLAS, J.M., AND GALLAIRE, H. Data base: theory vs. interpretation. In *Logic and Data Bases,* H. Gallaire and J. Minker (Eds.), Plenum Press, New York, 1978.
38. NICOLAS, J.M., AND YAZDANIAN, K. Integrity checking in deductive data bases. In *Logic and Data Bases,* H. Gallaire and J. Minker (Eds.), Plenum Press, New York, 1978.
39. QUINE, W.V.O. *From a Logical Point of View.* Harper and Row, New York, 1953.
40. QUINE, W.V.O. *Word and Object.* MIT Press, Cambridge, Mass., 1960.
41. REITER, R. On closed world data bases. In *Logic and Data Bases,* H. Gallaire and J. Minker (Eds.), Plenum Press, New York, 1978.
42. RESCHER, N., AND URQUHART, A. *Temporal Logic.* Springer-Verlag, New York, 1971.
43. SCHMID, H.A., AND SWENSON, J.R. On the semantics of the relational data model. In *Proc. ACM SIGMOD International Conference on Management of Data* (San Jose, Calif., May 14–16, 1975), ACM, New York, pp. 211–223.
44. SERNADAS, A. Temporal aspects of logical procedure definition. *Inf. Syst. 5* (1980), 167–187.
45. SHIPMAN, D.W. The functional data model and the data language DAPLEX. *ACM Trans. Database Syst. 6*, 1 (March 1981), 140–173.
46. SMITH, J.M., AND SMITH, D.C.P. Database abstractions: aggregation and generalization. *ACM Trans. Database Syst 2*, 2 (June 1977), 105–133.
47. STONEBRAKER, M., AND KELLER, K. Embedding expert knowledge and hypothetical data bases into a data base system. In *Proc. ACM SIGMOD International Conference on Management of Data* (Santa Monica, Calif., May 14–16, 1980), ACM, New York, pp. 58–66.
48. ULLMAN, J.D. *Principles of Database Systems.* Computer Science Press, Potomac, Md., 1980.

Received June 1981; revised June 1982; accepted October 1982