A MODEL FOR HISTORICAL DATABASES

James Clifford

November 1982

Center for Research on Information Systems
Computer Applications and Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #47

GBA #82-76(CR)

Presented at Conference "Logical Bases for Databases," Toulouse, France, December 1982.

This material is based upon work supported by the National Science Foundation under grant #IST-8010834.

Categories and Subject Descriptions: H.2.1 [Database Management]: logical design - data models.

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

ABSTRACT

It is becoming increasingly apparent that we are on the verge of several new technologies that will offer virtually unlimited secondary storage at affordable prices. Database applications can be expected to take advantage of this expanded storage capacity, and a particularly promising area in this regard is the use of so-called "non-deletion" or "historical" databases. It is therefore appropriate to begin exploring formal models for these historical databases -- models that are intrinsically oriented toward the storage of data over the course of time, and that provide a formal semantics for the interaction between time and the other stored data items.

We present such a model, the Historical Database Model (HDBM), and define its semantics in terms of an underlying logical model. For this purpose we use the language IL-s and its model theory, a simplified version of Richard Montague's higher-order lambda calculus with intensions. The HDBM is defined as an extension of the relational database model, incorporating a distinguished STATE attribute that "time-stamps" the facts recorded in the database. Intuitively such a database can be viewed as a set of three-dimensional relations in the ordinary sense. The formal semantics is defined in terms of objects (the values of keys), which are identified with non-varying or constant entities, and the properties of these objects (the values of non-key attributes), which are identified with individual concepts in the intensional model. Two possible encodings of the database into the logical model are presented and discussed.

It is a widely accepted view that first-order logic provides a formalization of the semantics of the relational database model that has helped to clarify many of the issues in relational database theory. We argue that the richer logic IL-s, with its built-in notion of "denotation with respect to a moment of time" and with its capability for naming higher-order objects, is an appropriate vehicle for providing an analogous formal theory of the semantics of an HDB. Finally, we briefly discuss our work using IL-s as a target language for interpreting a natural-language query fragment which we have defined as a Montague Grammar, and point to some interesting topics for further research in the general area of time and databases.

I. Introduction

It is becoming increasingly apparent that we are on the verge of several new technologies that will offer virtually unlimited secondary storage at affordable prices. Optical disks written by laser will soon be available commercially (see [Marshall 1981], e.g.), and researchers are beginning to investigate new storage strategies appropriate to these devices (see [Maier 1981]). Database applications will undoubtedly want to take advantage of this expanded storage capacity, and a particularly promising area in this regard is the use of so-called "non-deletion" or "historical" databases. It is therefore appropriate to begin exploring formal models for these historical databases -- models that are intrinsically oriented toward the storage of data over the course of time, and that provide a formal semantics for the interaction between time and the other stored data items.

In [Clifford & Warren 1981] we presented such a model, the Historical Database Model (HDBM), and defined its semantics in terms of an underlying logical model. For this purpose we used the language IL-s and its model theory, a simplified version of Richard Montague's higher-order lambda calculus with intensions. The HDBM was defined as an extension of the relational database model, incorporating a distinguished STATE attribute that "time-stamps" the facts recorded in the database. Intuitively such a database can be viewed as a set of three-dimensional relations in the ordinary sense. The formal semantics is

defined in terms of objects (the values of keys), which are identified with non-varying or constant entities, and the properties of these objects (the values of non-key attributes), which are identified with individual concepts in the intensional model. In this paper we present an overview of the HDBM and discuss two possible encodings of an historical database into the logical model of IL-s.

It is a widely accepted view that first-order logic provides a formalization of the semantics of the relational database model that has helped to clarify many of the issues in relational database theory. We argue that the richer logic IL-s, with its built-in notion of "denotation with respect to a moment of time" and with its capability for naming higher-order objects, is an appropriate vehicle for providing an analogous formal theory of the semantics of an HDB. Finally, we briefly discuss our work using IL-s as a target language for interpreting a natural-language query fragment which we have defined as a Montague Grammar, and point to some interesting topics for further research in the general area of time and databases.

The paper is organized as follows. We begin in Section II by presenting certain necessary definitions and comments on our notation. Then in Section III we motivate the need for a temporal component to a database model and discuss two intuitive views of time's impact on the rest of the model. Section IV discusses the relationship between our intuitive view of time

and the logical formalism of IL-s, and outlines two posisble encodings of an historical database into an IL-s model. Finally we present a brief discussion of our work in natural-language database querying using a Montague Grammar approach, and discuss some of areas for further research.

II. Definitions and Notation.

A. Relational Database

This section introduces some of the standard definitions from the relational database model (mostly from [Maier]), 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 $A = \{A1, A2, \ldots, An\}$ and a finite set of key attributes $K = \{K1, K2, \ldots, Km\}$, where $K \subset A$. We generally underline the key attributes and write such a relation scheme as $R(\underline{A1}\ \underline{A2}\ \ldots\ \underline{Am}\ Am+1\ \ldots\ An)$; in this case it is to be understood that $A = \{A1, \ldots, An\}$ and $K = \{A1, \ldots, Am\}$. We will occasionally refer to such an R as an n-ary relation scheme. The attributes $Am+1, \ldots, An$ are referred to as role attributes.

To say that $K = \{K1, K2, \ldots, Km\}$ 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, $t\text{-}1(K) \neq t\text{-}2(K)$, and no proper subset of K has this property.

The values for the attributes come from a set D of domains, $D = \{D1, D2, \ldots, Dk\}$, each Di any non-empty set. We let UD denote the union of these domains, i.e. $UD = D1 \cup D2 \cup \ldots \cup Dk$.

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 ---> D which maps each attribute onto its corresponding domain, i.e., DOM(Ai) is the domain of the attribute Ai.

Finally, we say that a relation r on relation scheme $R = \langle A,K \rangle$ is a finite set of mappings $\{t\text{-}1, t\text{-}2, \ldots, t\text{-}n\}$ where each t-i is a function from A to UD such that $t\text{-}i(A\text{-}j) \in DOM(A\text{-}j)$ for all $t\text{-}i \in r$ and all $A\text{-}j \in A$. The constraint that $K = \{A1, \ldots, Am\}$ is a key of scheme R means that any valid relation r on R has the property that for any distinct tuples t-i and t-j in r, $t\text{-}1(K) \neq t\text{-}2(K)$, and no proper subset of K has this property.

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

We assume that the reader is familiar with the usual relational operations, project, select and join given relations r on scheme R and s on scheme S:

$\Pi_X(r)$, is the relation r' on X, such that $r = \{t(X) : t \in r\}$;

$\Delta_{Ai=a}(r)$, is the relation r' on scheme R, such that $r' = \{t : t \in r \text{ and } t(A\text{-}i) = a\}$;

$r \bowtie s$, is the relation q on $Q = R \cup S$, such that

$q = \{t :$ there are tuples $t\text{-}r \in r$ and $t\text{-}s \in s$

s.t. $t\text{-}r = t(R)$ and $t\text{-}s = t(S)$ $\}$.

If r is a relation on R and X and Y are subsets of R, then r satisfies the functional dependency (FD) X ---> Y if for any value x, $\Pi_Y(\Delta_{X=x}(r))$ has at most one tuple.

B. Entity-Relationship Semantics

We have adopted the entity-relationship view of data semantics [Chen 1976] as applied to the relational model for two main reasons. First, the constraints that the entity-relationship model makes upon the database view of an enterprise seem "natural." Second, entities and relationships are very closely analogous to kinds of objects 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, the similarity in some of their concepts strengthens their intuitive appeal. The following definitions relate the entity-relationship model to the relational database model; in [Clifford & Warren 1981] we present a set of historical entity-relationship constraints that expand the model to include a conception of time.

An entity relation is a relation r on a scheme R of the form (K1 A1 ... An) where K1 is the key and any k-value for K1 uniquely determines the values for each of the other attributes. (This essentially means that each entity relation is in BCNF; see [Ullman 1980] for a discussion.) Intuitively, a K1-value k uniquely identifies some entity of interest to the database, and each Ai-value associated with k gives one of k's attributes. 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 (K1 ...Kn A1 ... Am) where {K1, ..., Kn} is the key and determines the values of the other attributes. Intuitively, a <K1, ..., Kn>-value <k1, ..., kn> represents an n-ary relationship among the n entities k1, ..., kn, and each Ai-value associated with <k1, ..., kn> gives an attribute of that relationship.

C. Intensional Logic

In this section we introduce enough concepts of intensional logic as will make the rest of this paper intelligible; for more information we recommend [Dowty 1981] as an excellent introduction before attempting Montague's extremely terse presentation [Montague 1973].

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 plus 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 [Church 1941].) (Readers familiar with the programming language LISP [McCarthy 1962] are familiar with the general concepts of lambda-abstraction. 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.

After that brief summary, we proceed with the following 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 <a,b> $\in$ T.

The interpretation function for the language assigns 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 of type s (for states) states or points of reference; and to expressions of type <a,b> 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, for the examples herein use only a portion of the language. Instead we stress the following points:

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 functional operator ¬ (not), and truth-functional connectives /\ (and), \/ (or), --> (material implication), <--> (mutual material implication), as well as = (equality) and the ordering relation symbol < ("prior to").

3. IL-s contains the universal and existential quantifiers, ∀ and ∃, respectively.

In addition to the usual rules of formation are the following:

1. if A is an expression of type <a,b> and B an expression of type a, then A(B) is an expression of type b, and denotes the result of applying the function denoted by A to the object denoted by B as argument.

2. if x is a variable of type a, and B an expression of type b, then λ x B is an expression of type <a,b>, and denotes a particular function from objects of type a to objects of type b.

It is the model theory of IL-s that is of most interest to us here. A model M for the language IL-s is an ordered 4-tuple M = <E,S,<,F> where:

1. E is a non-empty 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 a function which assigns to each constant c-a ∈ 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:

$D\text{-}e = E$
$D\text{-}t = \{0,1\}$
$D\text{-}s = S$
$D\text{-}\langle a,b\rangle = D\text{-}b^{D\text{-}a}$, i.e. the set of all functions from D-a to D-b

The following simple example should help to breathe a little life into these definitions. Assume a language with the following constants:

> Peter, Liz, Elsie, and THE_BOSS of type <s,e>,
> 77, 78, 79, 80, and 81 of type s, and
> EMP of type <s,<e,t>>,

and 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 the interpretation function F makes the obvious assignments to the state constants, and interprets the other constants as follows:

F(Peter) =		F(Liz) =	
	1977 --> Peter		1977 --> Liz
	1978 --> Peter		1978 --> Liz
	1979 --> Peter		1979 --> Liz
	1980 --> Peter		1980 --> Liz
	1981 --> Peter		1981 --> Liz

F(Elsie) =	
	1977 --> Elsie
	1978 --> Elsie
	1979 --> Elsie
	1980 --> Elsie
	1981 --> Elsie

These functions, from states to individuals, are called 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 Si they pick out the same individual. Compare how F interprets the constant THE_BOSS:

F(THE_BOSS) =

1977 --> Liz
1978 --> Peter
1979 --> Peter
1980 --> Peter
1981 --> Elsie

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

F(EMP) =

1977 --> {Liz}
1978 --> {Peter, Liz}
1979 --> {Peter, Liz}
1980 --> {Peter}
1981 --> {Elsie}

Such a function is called a property of individuals.

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 <s,<e,t>> and 78 is of type s, this expression is well-formed and is of type <e,t>. Its interpretation is given by applying the function which is the interpretation of EMP to the interpretation of 78, viz. 1978:

1977 --> {Liz} 1978 --> {Peter, Liz} 1979 --> {Peter, Liz} 1980 --> {Peter} 1981 --> {Elsie}	(1978) = {Peter, Liz}

Thus we see that the interpretation rules give the expected meaning to EMP(78), viz. Peter and Liz are the EMPloyees in 1978. Consider now the expression EMP(78)(Elsie), of type t. The denotation of this expression is "computed" by "applying the set" {Peter, Liz} (considered as a characteristic function) to the argument Elsie to obtain the value 0 (False); i.e., Elsie is not an EMPloyee in 1978.

Finally, we consider an example that makes explicit reference to time, the formula which translates the sentence "Elsie was the boss":

$$\exists i[\ [i < \underline{now}] \wedge \text{THE\_BOSS}(i)(\text{Elsie})]$$

If we assume that now (of type s) is interpreted as 1981, this formula is True just in case at some time i prior to 1981 Elsie was "the boss." 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.

III. Motivation and Informal Semantics.

The relational database model proposed in [Codd 1970] views a database as a collection of "time-varying relations of assorted degrees" [Chang 1978]. However the model itself incorporates neither the concept of time nor any theory of temporal semantics. We believe that the concept of time is of

interest in real-world databases, and shall present a technique for incorporating a semantics of time into a database model. Although we use the relational model for this purpose, it is not an essential ingredient in the work discussed.

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, an 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 is 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 an overview of such an organization, a discussion of the usefulness of the historical database concept for modelling the real world (or some "possible world") more closely than with a static database, and a discussion of the semantics of the database model in terms of the model theory of IL-s. For a good overview of the issues involved in incorporating a temporal dimension in databases we recommend [Bubenko 1977].

In order to motivate both the incorporation of time into a database model, and the particular formalism that we propose, we turn to a particular database application to provide concrete examples. Consider a static database with a relation emp_rel on scheme EMP_REL(EMP MGR SAL DEPT). 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 $\pi_{SAL}(\sigma_{EMP=John}(emp\_rel))$, and in a first-order language as something like $\{z \mid \exists x \exists y\ emp\_rel(John,x,y,z)\}$ where x, y, and z are individual variables and John is an individual constant. To answer such a query, a Data Manipulation Language (DML) simply accesses the relation instance emp_rel or EMP_rel, such as the one in Figure 1. In recent database literature (e.g. [Minker 1978], [Reiter 1978], [Chang 1978]) such a relation instance has been termed the extension of the relation scheme EMP_rel, a term borrowed from logic.

The relational model, and its interpretation as an applied first-order logic, is well-suited to queries of this sort. However, one can easily imagine other sorts of queries that casual users might want to ask about the employees in this company, e.g.:

"Has John's salary risen?"

"When was Peter re-hired?"

"Did Rachel work for the toy department last year?"

"Has John ever earned the same as Peter?"

Such time-dependent questions are not handled by the theory of any of the "three great data models;" indeed they have only recently begun to receive attention within the database literature ([Bubenko 1977], [Nicolas & Yazdanian 1979],

[Casanova & Bernstein 1979], [Laine et al. 1979], [Sernadas 1980], [Kloprogge 1981], and [Ariav 1982] are among the recent papers that discuss from various points of view the need for a temporal semantics for databases). How can we incorporate a temporal component into these data models so that (a) the model provides a built-in, uniform treatment of the phenomenon of time and (b) this treatment accords with our intution about how time interacts with the entities and relationships that we perceive?

The success of first-order logic at providing a formalization of the semantics of the relational model suggests that it might prove equally successful here. Why not simply take as our universe a set containing, in addition to all of the entities of interest to the database enterprise, all of the times as well? This approach in effect would lead us either to a sorted first-order logic, or equivalently to the introduction of predicates (such as ENTITY and TIME) to distinguish between these two kinds of objects. We believe, however, that such an approach obscures the fundamental "differentness" of time. Time is not just another sort of entity; it is rather a point of reference with respect to which all other denoting expressions receive an interpretation. In other words, without intending to beg the question by the use of "loaded" terminology, time is as different in type from an entity such as a person, as a person is of a fundamentally different type from the truth value "False." And so while it is perhaps technically feasible (see, e.g., [Bolour 1981]) to provide an account of the semantics of an historical relational database by encoding time into the

universe of objects, we strongly suggest that the use of a temporal logic, with a built-in and fundamental distinction between time and entities, is more appropriate.

There are, of course, many choices for such a temporal logic (see [Rescher & Urquhart 1971], e.g.). In [Clifford 1982] we present arguments for our choice of the Intensional Logic (IL) framework of Richard Montague, arguments having to do with its application as a target language for a natural-language front-end query processor.

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 25K, it clearly makes no sense to ask whether 25K "rises." To answer this question more data is needed than the current extension of John's salary: the values of John's salary for some other point(s) of time are needed. The expression $\{z \mid \exists x \exists y\ emp\_rel(John,x,y,z)\}$, then, has two very different meanings in these two queries. 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, 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. We suggest that such things as SALaries, be identified, not with individual entities ($'s), but with entities ($'s) in the role of an EMPloyee's salary. The salary of an individual employee is not another individual, but rather an individual concept, whose extension depends upon the state at which it is evaluated. Such an object, whose extension is dependent upon the state, is said to be "intensional." (The terms extension and intension are given formal definitions in intensional logic. They should not be confused with their usage in some database papers where the term "intension," e.g. in [Reiter 1978], is used to refer to axioms which constrain the set of possible models for the database.) It is helpful to think of these ICs as filling a role; at any moment of time the role of so-and-so's salary might be filled by an appropriate dollar amount.

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?"

Suppose that an enterprise wishes to maintain an historical database, containing the entity relation scheme EMP_rel, and further suppose that at three different points in time the static relation instances emp_rel-1, emp_rel-2, and emp_rel-3 in (Figure 3) represented the real world situation. Then this historical information can easily be encoded into a single, historical relation by extending the relation scheme to include an attribute, call it STATE, to "time-stamp" each tuple, and then merging the three resulting relations. However, there is an additional problem that we must solve: some EMPloyees are not represented in every state. For example, John is not an EMPloyee in state S3, and therefore there is no tuple for John in this new relation. Given the query "What is John's salary in S3?" 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 S3. In other words, the question of who "exists" in the modelled enterprise in any given state becomes of paramount importance to a successful database modelling of that enterprise. In order to handle this problem we introduced the concept of a completed relation, and an additional, Boolean-valued attribute EXISTS?.

A completed relation has a tuple in each state for every entity that is an EMPloyee in <u>any</u> state in the entire database. In this way the database "follows" objects and their attributes throughout all of the states of the database. To accomplish this we must determine all of the objects (key values) that are represented in any relation instance. We introduce the notions of the <u>Active Key Domain (AKD)</u> of a relation instance as the set of all key values (entities or relationships) in the relation instance, and for a set of relation instances, the <u>Complete Active Key Domain (CAKD)</u> as the union of the AKD's of each instance in the set. Thus, for example, the CAKD of the set of relation instances in Figure 3 is {John, Mike, Elsie, Liz, Rachel, Peter, Sharon, Beth}: all of the EMP entities "known" to the database.

We then extend (conceptually) each relation instance so that it has a tuple for each entity in CAKD, the set of all "possible" EMPloyees that are "actual" in some state, using the attribute EXISTS? as follows. If the entity k is an actual entity in state S-i, then in the expanded relation the tuple t-k representing k will have t-k(EXISTS?) = 1 and all the remaining attributes will retain their appropriate values. On the other hand if k is not an actual entity in state S-i, the tuple t-k will have t-k(EXISTS?) = 0, but the distinguished value "<u>!</u>" for every other attribute other than STATE. <u>!</u> indicates the <u>inapplicability</u> of this information for this entity, i.e., that no individual fills that role for that entity. The completed relations for the emp-rel instances is shown in Figure 4; we

could also consider the union of these three relations as a single, historical employee relation.

These completed relation instances provide 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 proposed the picture of an 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. The three-dimensional cube representation of the completed relation, such that the i-th plane of the cube is the completed relation corresponding to state S-i, is shown in Figure 5. In this view time is seen as adding a third dimension to the normal flat-table view of relations, and we can visually follow the changes in the facts about each EMPloyee through a three-dimensional row of the cube.

An alternative view of the effect of time, one that accords more closely with the view provided by the logical model, is proposed in Figure 6. Here we see that a key value such as an EMPLoyee can be modelled by a constant individual, while the attributes of an EMPLoyee must be modelled by individual concepts whose value (extension) is a function of the state. In [Clifford FC] we discuss how this view is more appropriate to providing a proper treatment of historical realtions with a modified relational algebra that accords distinguished status to the STATE attribute.

Several general categories of assumptions were identified in [Clifford & Warren 1981] that had to be made with respect to the temporal component of an historical database if a complete account of the entities and relationships was to be given. These assumptions have the same flavor as the Closed World Assumption of Reiter [1978] but are expanded to incorporate the temporal dimension.

The simplistic three-dimensional view of snapshot relations is obviously inadequate in the face of the generally accepted notion of dense time. Two additional assumptions, which we call the Comprehension Principle and the Continuity Assumption, enable us to view an HDB as modelling 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 role attributes in these states.

Consider again the historical entity relation scheme EMP_REL(STATE EMP DEPT MGR SAL), and an instance defined for the sequence of states <S1, S2, ..., S7>. The first assumption about any such relation is that it is intended to model EMPloyee entities over the entire closed interval of time [S1,S7]. Since under the most reasonable views of time this interval is considered to be dense, the best that any finite relation can do is provide a simulation of this infinite set of moments of time. It does this by means of a sequence of snapshots, in this case taken at each moment in the sequence <S1, ..., S7>. Because this 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 an historical database defined over a sequence of states <S1, S2, ..., Sn> should be considered as modelling an enterprise completely over the entire closed interval [S1,Sn]. 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 [S1,Sn]. Moreover, for any state S not in the interval [S1,Sn], as far as the database is "concerned" no entities or relationships exist, and the value of all ICs is $\underline{|}$.

One area for further research would be the relaxation of the second part of this principle, perhaps with the introduction of a many-valued logic. Future work might also incorporate other so-called "null-values" (in addition to "$\underline{|}$") as a formal null-value semantics is developed (see [Goldstein 1981], e.g.).

The other assumption we must make concerns the interpretation of the database for those moments of time in the interval [S1,S7] which are not included in the sequence <S1, S2, ..., S7>. The database "samples" the values of the ICs of interest for only some finite subset of states in [S1,Sn], yet we want to be able to consider that the database implicitly defines each IC as a total function from states to individuals.

Definition.

Any assumption which extends a database mapping from a finite set of moments {S1, S2, ..., Sn} (ordered as in the sequence <S1, S2, ..., Sn>) into a set of individuals, into a mapping from all moments in the closed, dense interval [S1,Sn] into that set of individuals, we call a Continuity Assumption.

A number of different possiblities exist for interpolating these role functions in the database, but by far the most common one is to assume, as in Figures 7a and 7b, that a step function is the intended interpretation. Certainly for all role attributes that record non-numeric data (e.g., MGR, DEPT), and for many that record numeric data (e.g., SAL), this is the only appropriate choice. Under the Step-Function Continuity Assumption the value of an IC for any state s within the database is given by the value of the function recorded in the database at the greatest state s' less than or equal to s. It is assumed that the HDB initially records information about an object X when it becomes of interest to the enterprise, say at state s-i, and 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 role ICs has changed value, or when it ceases to be an object of interest to the enterprise (EXISTS? becomes 0.)

This discussion of the HDBM has been informal and mostly intuitive. Four points are perhaps appropriate at this point. First, the technique of time-stamping each tuple is an obvious and fairly simple idea, and many databases have kept information such as salary histories in a similar way. The STATE and

EXISTS? attributes of the HDBM, however, are distinguished attributes that are an intrinsic part of the model, and not ordinary attributes under the user's direct control. An explicit temporal semantics can thus be incorporated directly within the framework of the relational model, provided that the model (including the DML) is extended to include a special treatment for these attributes. Second, although we have presented a very general notion of time, it is assumed that through the technique of Meaning Postulates ([Carnap 1947], [Montague 1973]), or axioms that constrain the set of allowable models, the user of a real HDB could make certain modifications (for example, defining appropriate intervals of interest) to the general temporal semantics of the model. Third, although our examples have used a single entity relation, the extension to relationship relations and to an entire database of historical relations is developed in [Clifford & Warren 1981]. Finally we point out that this model of an historical database is theoretical; no remarks in this paper should be construed as referring to implementation strategies. Obviously a direct implementation (for instance of "completed relations") would be extremely cost-prohibitive for any real database. We are currently in the process of developing a number of different implementation strategies and algorithms for an HDBM that eliminate the redundancies of the formal model.

IV. Historical Databases and IL-s

In this section we present an overview of the semantic interpretation of the HDBM informally outlined in the previous section. The formal details presented in [Clifford & Warren 1981] were based upon what has been called the Universal Role Assumption (URA) (see [Maier & Warren 1982] and [Ullman 1982]), that there is a unique relationship between any two database attributes. We present a brief outline of that treatment here, using the sample database in Figure 8 (based on the department-store database in [Chang 1978]) as a running example, and briefly discuss another encoding scheme that makes no such assumption.

Under the URA it is possible to ignore the names given to relations and allow the attributes to identify the relationships that hold in the modelled enterprise. It is therefore possible to encode an HDB into a logical model using five different classes of functions denoted by non-logical constants in IL-s. The information in an HDB is organized in the form of historical entity and historical relationship relations. We encode this information in the logical model by a set of functions which are defined implicitly by the database. We will discuss this encoding in terms of the names (non-logical constants) of these functions in the language IL-s, and briefly discuss an alternative encoding.

For each HDB we 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.

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.

The domain of the distinguished attribute STATE is some set of states S in our logical model. 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 <s,t> 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, viz. C-<s,t> . 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 sorts of constants.

The set of entities (e.g. EMPloyees) in any state is given by the denotation of the corresponding entity constant (of type <s,<e,t>>) for that entity set. For example, EMP-* denotes, for any state s, the set of employees in state s.

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. For example, encoding the binary historical relationship relation SALES_rel reqquires the constant REL-2 of type <s,<e,<e,t>>>. 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. All n-ary relationships can be combined into a single function (REL-n) since we assume the entity sets of the participants uniquely determine the relationship.

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. An IC fills a role only in those states in which its associated object exists (or, equivalently, in which its value is not $\underline{\iota}$.) 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 <s,<<s,e>,t>>. DEPT', for example, denotes in any state the set of DEPT-ICs (i.e., department-of-some-employee roles) that exist in that state.

n-ary objects (entities or relationships) are bound <u>permanently</u> (i.e., not as a function of the state) to all of their role ICs by the denotation of the non-indexical constant AS-n. Thus, e.g., each EMPloyee is bound to three ICs which, in those states in which the employee exists, are its SAL-, MGR-, and DEPT- picking-out functions. The constant AS-1 of type <e,<<s,e>,t>> in the logic represents the association between

each entity (object of arity 1) and its role IC's; AS-2 of type <e,<e,<<s,e>,t>>> represents the association between each binary relationship and its role IC's, etc.

Any given HDB scheme thus determines a set C-HDB of constants in IL-s from among these six categories of non-logical constants. (These constants are uniquely determined except for the constants of type <s,t>, for which many choices can be made.) In the case of the department-store database, the following set of constants is determined:

C-e is the set of domain value constants,

C-s is the set of state constants,

C-<s,t> is some set of state-set constants,

C-<s,<e,t>> = {EMP-*, ITEM-*} is the set of entity constants

C-<s,<e,<e,t>>> = {REL-2} is the set of relationship constants

C-<s,<<s,e>,t>> = {MGR',DEPT',SAL',TYPE',VOL'} is the set of role constants,

and

C-<e,<<s,e>,t>> = {AS-1} $\cup$ C-<e,<e,<<s,e>,t>>> = {AS-2} is the set of

association constants.

In [Clifford & Warren 1981] we presented formal definitions of an HDB scheme and an instance hdb on this scheme, and showed how the interpretation of the constants determined by a given historical database scheme HDB is induced by an instance hdb over that scheme. An alternative encoding that does not rely on the URA would treat each relation separately and define a

corresponding non-logical constant in IL-s of the appropriate type and interpretation. For example, the relation emp-rel could be encoded into a function denoted by the constant EMP-REL of type <e,<<s,e>,<<s,e>,<s,e>>>>. Such a function would represent the association between an entity (the employee) and its three IC-s (manager, department, and salary) (depicted in Figure 6).

We conclude by taking a brief look at three of the elements of the relational model to see how their correlates in the HDBM have been affected by the temporal semantics with which they have been provided.

Attributes.

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." [Montague 1973]. We have made the same claim here in the HDB realm; in particular we have argued that key attributes (like EMP) and role attributes (like SAL) are to be identified with Montague's "ordinary" and "extraordinary" common nouns, respectively.

The attribute STATE 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 a Step-Function Continuity Assumption as a means of interpolating the partial function given by the historical database. The attribute EXISTS? models the changing focus of interest of objects to an 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 $\perp$.)

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

define an extensional database constraint as a constraint on individual valid states of the database, and an intensional database constraint as a constraint which defines vaild state progressions in the database. An intensional constraint can only be said to hold (or not to hold) only by examining at least two states of the HDB.

Current theoretical relational database research has been primarily concerned (without using the term) only 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., [Nicolas 1978] and [Nicolas and Gallaire 1978].) The FD EMP --> SAL, e.g., is an abbreviation for the first-order formula:

$$\forall x \forall y \forall z [EMP(x) \wedge SAL(y) \wedge SAL(z) \wedge AS\text{-}1(x,y) \wedge AS\text{-}1(x,z) \rightarrow y = z]$$

in the domain relational calculus (i.e., with variables having individuals as their domain). An intensional logic allows us to easily express more fully the full 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:

$$\forall x \forall y \forall z \forall i [EMP(i,x) \wedge SAL(i,y) \wedge SAL(i,z) \wedge AS\text{-}1(x,y) \wedge AS\text{-}1(x,z) \rightarrow y = z]$$

Here we have quantified over all states of the database with the

state variable i (type s), and have equated, not merely the value (extension) of the two SALaries, but the SALary-ICs (functions) themselves.

Intensional constraints have not received much attention in the database literature. Where they have been examined (e.g. by Smith and Smith [1977], Nicolas and Yazdanian [1978], and Casanova and Bernstein [1979]) as "dynamic constraints" or constraints upon update operations), they have been considered as different in kind from extensional (or "static") constraints. 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, i.e., as axioms about objects (of the appropriate type), without having to invent a new technique for expressing the dynamic constraints. For example, the constraint:

No employee can ever be given a cut in pay.

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 non-negative derivative. It is not clearly 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, viz. an IC. In IL-s this constraint is

expressible as:

$$\forall\ i\text{-}1\ \forall\ u\ \forall\ x\ [\text{EMP-}*(i\text{-}1,u)\ /\backslash\ \text{SAL}'(i\text{-}1,x)\ /\backslash\ \text{AS-}1(u,x)\ \text{-->}\ \forall\ i\text{-}2\ [\ i\text{-}1 < i\text{-}2\ \text{-->}\ x(i\text{-}1) <= x(i\text{-}2)]\ ]$$

This ability to consider both intensional and extensional constraints as essentially the same kind of constraints, and to express them in the same language, is a good example of the power of IL-s to provide a unified theory of database semantics.

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 to define an extensional database query as a query whose evaluation depends only on the values in the database with respect to a single index or state, and an intensional database query as a query whose evaluation depends on the intensions of at least one attribute, i.e on the function from states to individuals (ICs) that represents that attribute.

Extensional queries are precisely those that static databases have been concerned with handling; these queries are handled just as well by an historical database. However, 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 S2 yields the answer "30K," but

with respect to state S3 what appears to be the same query of the same database yields the equally correct (but different) answer "35K." 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 [Clifford 1982] 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.

Intensional 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. Within the context of an HDB we have the potential to answer all of the queries which were mentioned at the beginning of Section III. Assume a mechanism for translating the query "Has John's salary risen?" into the following formula in IL-s:

$$\exists x\,[\mathrm{SAL}'(\mathrm{now},x) \wedge \mathrm{EMP}\text{-}{}^{*}(\mathrm{now},\mathrm{John}) \wedge \mathrm{AS}\text{-}1(\mathrm{John},x) \wedge \mathrm{RISE}'(\mathrm{now},x)]$$

To evaluate this query, we need to provide a meaning to the predicate RISE'. 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 iff there is a preceding interval of time culminating in state i during which the SAL-IC is monotonically non-decreasing. 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:

$$\forall x \forall i [RISE'(i,x) \leftrightarrow [SAL'(i,x) \wedge \exists i\text{-}1 \forall i\text{-}2 \forall i\text{-}3 [i\text{-}1 \leq i\text{-}2 < i\text{-}3 \leq i \rightarrow x(i\text{-}2) \leq x(i\text{-}3)]]]$$

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:

S1 -->	⊥
S2 -->	30
S3 -->	35

and whose value for all other states is ⊥. Let us call this function SJ. Then RISE'(i,SJ) evaluated for i = S3 is true (pick S2 as the i-1 which the MP asserts must exist).

V. Summary and Future Work.

Under the general assumption that formal logic has made and can continue to make important contributions to the understanding and specification of the semantics of databases, we have tried to show how the temporal semantics of the logic IL-s can be used to formalizes the concept of an historical database. In [Clifford 1982] the choice of IL-s is also motivated from the perspective of providing a formal definition of an English Query Language as a Montague Grammar (MG). Figure 9 depicts the overall scope of that work, which defined a formal relationship between an historical database and an IL-s model, and between queries expressed in a formally defined fragment of English and their interpretation in the same IL-s model. Here we have presented both an informal discussion of an HDB as a

cube composed of a time-ordered sequence of flat, static relations, and an overview of the relationship between an HDB and the logic IL-s and its model theory. We have also 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.

The HDBM suggests the possibility of formalizing a wide variety of database-semantic issues "under one roof," viz. within the precise model-theoretic semantics of IL-s. In addition to the natural-langauge querying potential discussed in [Clifford 1982], we mention a number of other issues here.

The first question that this paper will suggest to many readers will be that of implementation. Even if all of the information in the 3-dimensional 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 timestamp taken from the system clock. As with questions, intensional logic gives us a framework for providing a formal semantics (or pragmatics -- see [Clifford 1982]) 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 interpretation of error-correction types of maintenance, i.e., the sort of commands which mean, not that a given once-true fact about the world no longer obtains, but rather that a previous specification of that "fact" was in error. Also, 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 (e.g., if Peter's SALary is re-specified, 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 [IMS360], network [CODASYL 1971], and functional [Shipman 1981] models to include a temporal semantics is another area for future study. 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 provide 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. A query like "Will the average salary in the linen department surpass 30K within the next 5 years?" is the sort of question that we envisage could be handled by such an organization. Salary raises built into union contracts, cost-of-living increases, are the sorts of applications that an historical database ought to be able to model. How, for example, might an historical database be incorporated within the context of a decision support system ([Ginzberg 1982]). Stonebraker and Keller [1980] provide an examination of some of these possibilities from a different perspective.

In the simple model we have presented here, EXISTence is synonomous 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 <u>some</u> relation. For example, we could model people with a relation on scheme

PERSON_REL(<u>NAME STATE</u> EXISTS? GENDER ...)

and then have relations or schemes like

BORROWER_REL(NAME STATE IS_BORROWER? ACCT#...)

and

DEPOSITOR_REL(NAME STATE IS_DEPOSITOR? ACCTBAL...).

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.

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, need further study. 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, etc., 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 [1976], Hammer and McLeod [1978], and Biller and Neuhold [1978], 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 these Semantic Data Definition Languages (SDDLs) that are not provided with a formal semantics. We believe that the use of more powerful logics, like IL-s, can clarify many of the issues involved in these higher-level semantic models and languages, even to the point of providing a basis for constructing proofs that demonstrate their equivalence or differences.

References

Ariav, Gad and Howard L. Morgan (1982). "MDM: Embedding the Time Dimension in Information Systems," TR #82-03-01, Dept. of Decision Sciences, The Wharton School, Univ. of Pennsylvania, Philadelphia.

Biller, Horst and Erich J. Neuhold (1978). "Semantics of Data Bases: The Semantics of Data Models," Information Systems 3,1, Pergamon Press, Great Britain, 11-36.

Bolour, Azad and Luc J. Dekeyser (1981). "Time Relations and Event Relations," Technical Report, Section of Medical Information Science, Univ. of California, San Francisco.

Bubenko, Janis A. Jr. (1977). "The Temporal Dimension in Information Modelling," in Architecture and Models in Data Base Management Systems, ed. G. M. Nijssen, North Holland, Amsterdam.

Carnap, Rudolf (1947). Meaning and Necessity, University of Chicago Press, Chicago.

Casanova, Marco A. and Philip A. Bernstein (1979). "The Logic of a Relational Data Manipulation Language," Proc. 6th ACM Symp. on Prog. Lang.

Chang, C.L. (1978). "DEDUCE 2: Further Investigations of Deduction in Relational Data Bases," in Logic and Data Bases, ed. Gallaire and Minker.

Chen, Peter Pin-Shan (1976). "The Entity-Relationship Model -- Toward a Unified View of Data," ACM Trans. on Database Systems 1,1, 9-36.

Church, Alonzo (1941). "The Calculi of Lambda-Conversion," Princeton University Press, Princeton.

Clifford, James and David S. Warren (1981). "Formal Semantics for Time in Databases," TR #81/025, Dept. of Computer Science, SUNY at Stony Brook, Stony Brook (submitted for publication).

Clifford, James (1982). "A Logical Framework for the Temporal

Semantics and Natural-Language Querying of Historical Databases," Ph.D. dissertation, Dept. of Computer Science, SUNY at Stony Brook, Stony Brook.

Clifford, James (forthcoming). "An Algebra for Historical Relational Databases," Working Paper, Dept. of Computer Applications and Information Systems, Graduate School of Business Administration, New York University, New York.

CODASYL Data Base Task Group (1971). "CODASYL Data Base Task Group Report," ACM, New York.

Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks," Commun. ACM 13,6 377-387.

Dowty, David R., Robert E. Wall and Stanley Peters (1981). Introduction to Montague Semantics, Reidel Publ. Co., Dordrecht.

Gallaire, Herve and Jack Minker (1978). Logic and Data Bases, Plenum Press, New York.

Ginzberg, Michael J., Walter Reitman, and Edward A. Stohr (1982). Decision Support Systems, North Holland, Amsterdam.

Goldstein, Billie S. (1981). "Constraints on Null Values in Relational Databases," TR #80/015, Dept. of Computer Science, SUNY at Stony Brook, Stony Brook.

Hammer, Michael and Dennis Mcleod (1978). "The Semantic Data Model: A Modelling Mechanism for Data Base Applications," Proc. ACM SIGMOD 1978, Austin.

IMS/360 - Application Description Manual, IBM White Plains, New York, GH-20-0765.

Klopprogge, Manfred R. (1981). "TERM: An Approach to Include the Time Dimension in the Entity-Relationship Approach," Proc. 2nd Intl. Conf. on Entity-Relationship Approach, Washington DC.

Laine, Harri, Olavi Maanavilja, and Eero Peltola (1979). "Grammatical Data Base Model" Information Systems 4,4 Great Britain.

Maier, David. The Theory of Relational Databases, to appear.

Maier, David and David S. Warren (1980). "A Theory of Computed Relations," TR 80/012, Dept. of Computer Science, SUNY at Stony Brook, Stony Brook.

Maier, David (1981). "Using Write-Once Memory for Database Storage," Technical Report, Dept. of Computer Science, SUNY at Stony Brook, Stony Brook.

Maier, David and David S. Warren (1982). "Specifying Connections for a Universal Relation Scheme Database," Proc. ACM SIGMOD 1982, Orlando, 1-7.

Marshall, M. (1981). "Optical Disks Excite Industry," Electronics, May 5, 1981.

McCarty, J. et al. (1962). LISP 1.5 Programmer s Manual, MIT Press, Cambridge.

Minker, Jack (1978). "An Experimental Relational Data Base System Based on Logic," in Logic and Data Bases, ed. Gallaire and Minker.

Montague, Richard (1973). "The Proper Treatment of Quantification in English," in Approaches to Natural Language, ed. K.J.J. Hintikka et al., Dodrecht, 221-242.

Nicolas, J.M. (1978). "First Order Logic Formalization for Functional, Multivalued, and Mutual Dependencies," Proc. ACM SIGMOD 1978, Austin, 40-46.

Nicolas, J.M. and H. Gallaire (1978). "Data Base: Theory vs. Interpretation," in Logic and Data Bases, ed. Gallaire and Minker.

Nicolas, J.M. and K. Yazdanian (1978). "Integriy Checking in Deductive Data Bases,", in Logic and Data Bases, ed. Gallaire and Minker.

Reiter, Raymond (1978). "On Closed World Data Bases," in Logic and Data Bases, ed. Gallaire and Minker.

Rescher, Nicholas and Alistair Urquhart (1971). Temporal Logic, Spinger Verlag, New York.

Schmid, Hans Albrecht and J. Richard Swenson (1976). "On the Semantics of the Relational Data Model," Proc. ACM SIGMOD 1976, 9-36.

Sernadas, Amilcar (1980). "Temporal Aspects of Logical Procedure Definition," Information Systems 5, 167-187, Great Britain.

Shipman, David W. (1981). "The Functional Data Model and the Data Language DAPLEX," ACM Trans. on Database Systems 2,2, 105-133.

Smith, J.M. and D.C.P. Smith (1977). "Database Abstractions: Aggregation and Generalization," ACM Trans. on Database Systems, 2,2, 105-133.

Stonebraker, M. and K. Keller (1980). "Embedding Expert Knowledge and Hypothetical Data Bases into a Data Base System," Proc. ACM SIGMOD 1980, Santa Monica.

Ullman, Jeffrey D. (1980). Principles of Database Systems, Computer Science Press, Potoman, MD.

Ullman, Jeffrey D. (1982). "The U.R. Strikes Back" Proc. ACM Symposium on Principles of Database Systems 1982, Los Angeles, 10-23.

EMPREL

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

relation emprel

Figure 1

1977 --> John
1978 --> John
1979 --> John
1980 --> John
1981 --> John

(a) Intension of "John"

1977 --> Linen
1978 --> Linen
1979 --> Linen
1980 --> Toy
1981 --> Toy

(b) Intension of "Department-of-John"

1977 --> 25K
1978 --> 25K
1979 --> 27K
1980 --> 27K
1981 --> 30K

(c) Intension of "Salary-of John"

Figure 2

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

relation emp-rel-1

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

relation emp-rel-2

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

relation emp-rel-3

Figure 3

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S1	John	1	John	Linen	23K
S1	Mike	1	Mike	Linen	17K
S1	Elsie	1	Elsie	Toy	26K
S1	Liz	1	Liz	Hardware	30K
S1	Rachel	1	Rachel	Hardware	29K
S1	Peter	1	Peter	Hardware	29K
S1	Sharon	0	⊥	⊥	⊥
S1	Beth	0	⊥	⊥	⊥

relation emp-rel-1'

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S2	John	1	John	Linen	25K
S2	Mike	1	Mike	Toy	20K
S2	Elsie	1	Elsie	Toy	27K
S2	Rachel	1	Rachel	Hardware	28K
S2	Sharon	1	Sharon	Hardware	25K
S2	Peter	0	⊥	⊥	⊥
S2	Beth	0	⊥	⊥	⊥
S2	Liz	0	⊥	⊥	⊥

relation emp-rel-2'

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S3	Beth	1	Beth	Linen	23K
S3	Elsie	1	Elsie	Toy	27K
S3	Rachel	1	Rachel	Hardware	28K
S3	Sharon	1	Sharon	Hardware	25K
S3	Peter	1	Peter	Hardware	33K
S3	John	0	⊥	⊥	⊥
S3	Liz	0	⊥	⊥	⊥
S3	Mike	0	⊥	⊥	⊥

relation emp-rel-3'

Figure 4

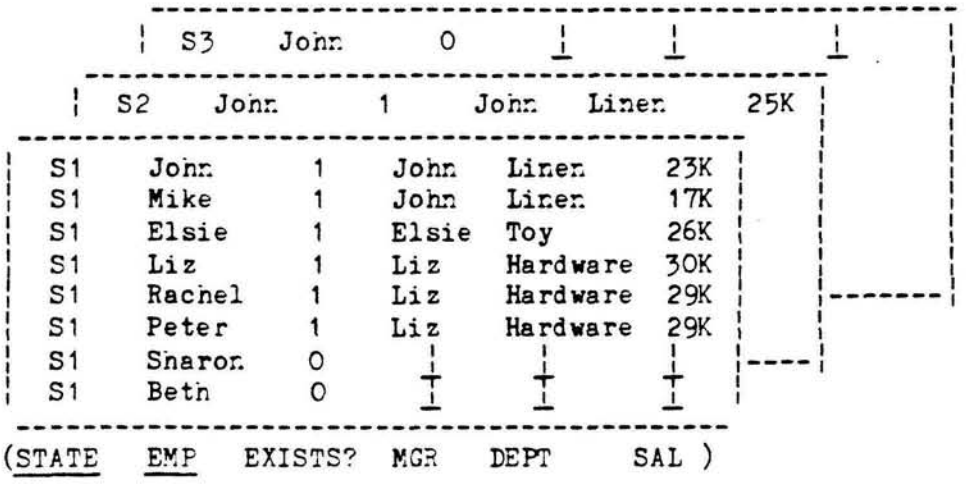


Figure 5

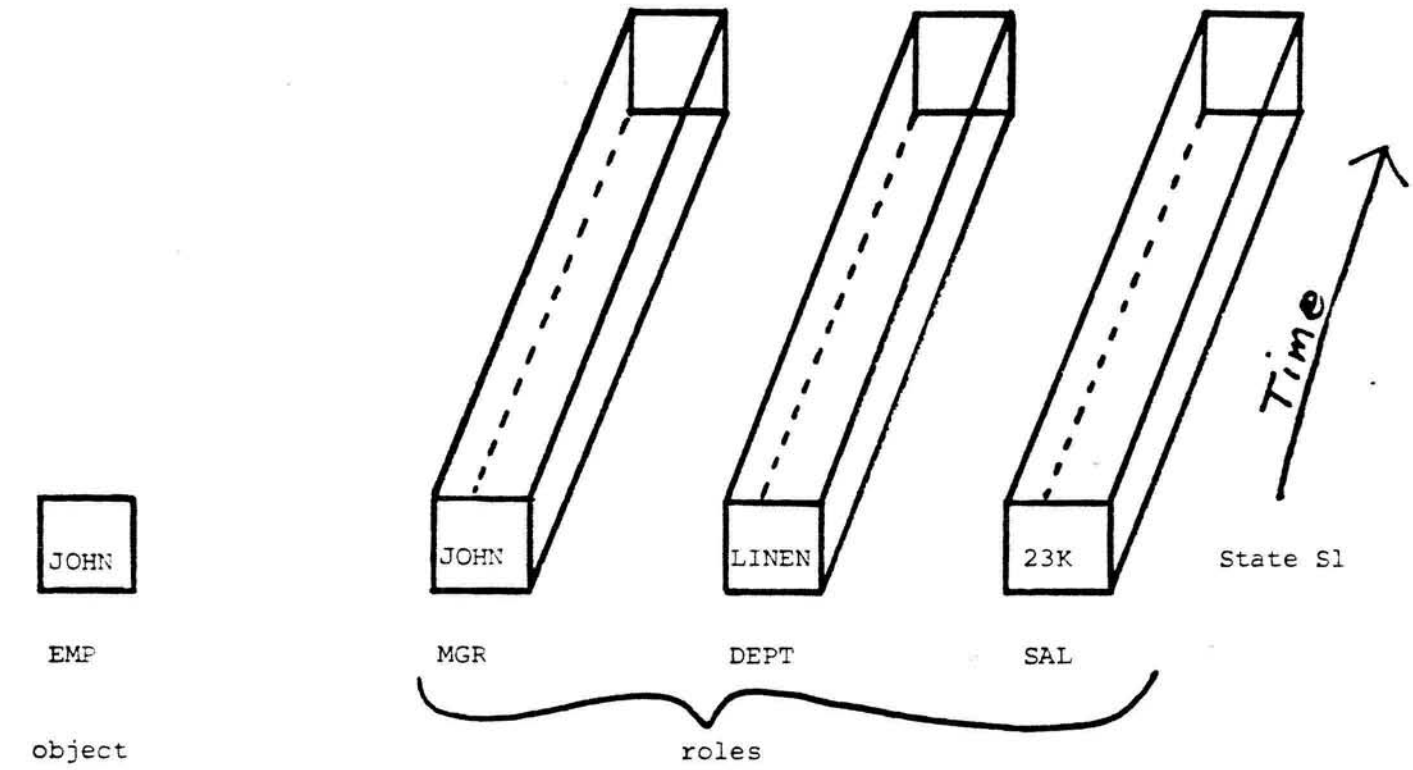
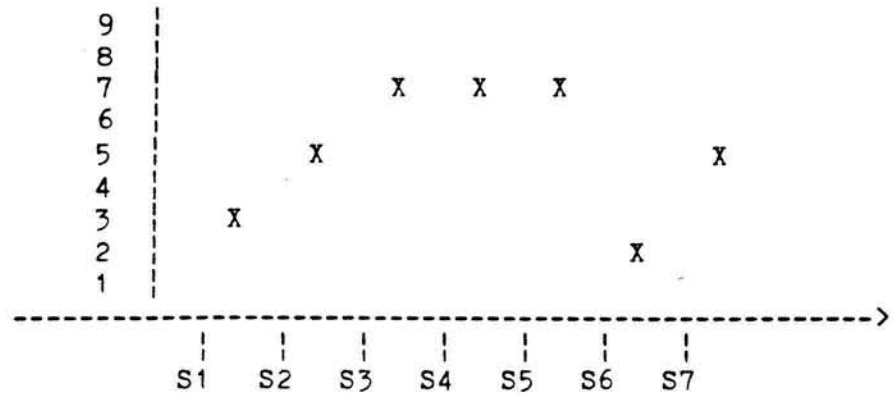
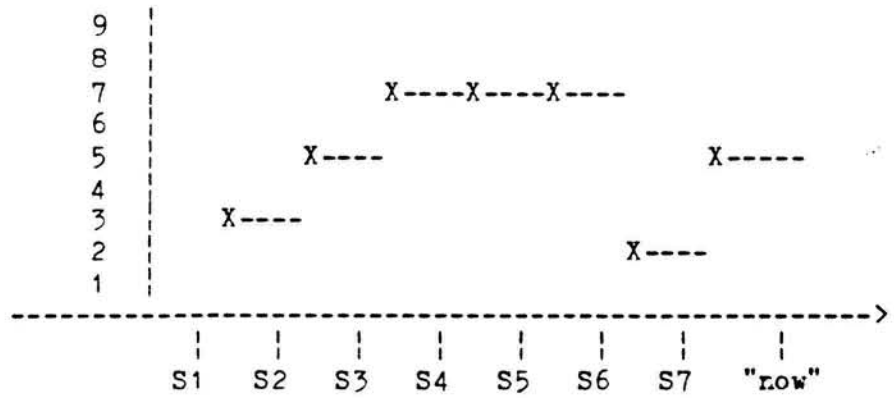


Figure 6



(a)



(b)

Figure 7

EMP-REL

STATE	EMP	EXISTS?	MGR	DEPT	SAL
S1	Peter	0	⊥	⊥	⊥
S1	Liz	0	⊥	⊥	⊥
S1	Elsie	1	Elsie	Toy	50
S2	Peter	1	Elsie	Hardware	30
S2	Liz	1	Elsie	Toy	35
S2	Elsie	1	Elsie	Toy	50
S3	Peter	1	Liz	Linen	35
S3	Liz	1	Liz	Hardware	50
S3	Elsie	0	⊥	⊥	⊥

relation emp-rel

ITEM-REL

STATE	ITEM	EXISTS?	TYPE
S1	Ball	1	5
S1	Game	1	6
S1	Glove	1	7
S2	Ball	1	0
S2	Game	1	6
S2	Glove	1	5
S3	Ball	1	10
S3	Game	0	⊥
S3	Glove	0	⊥

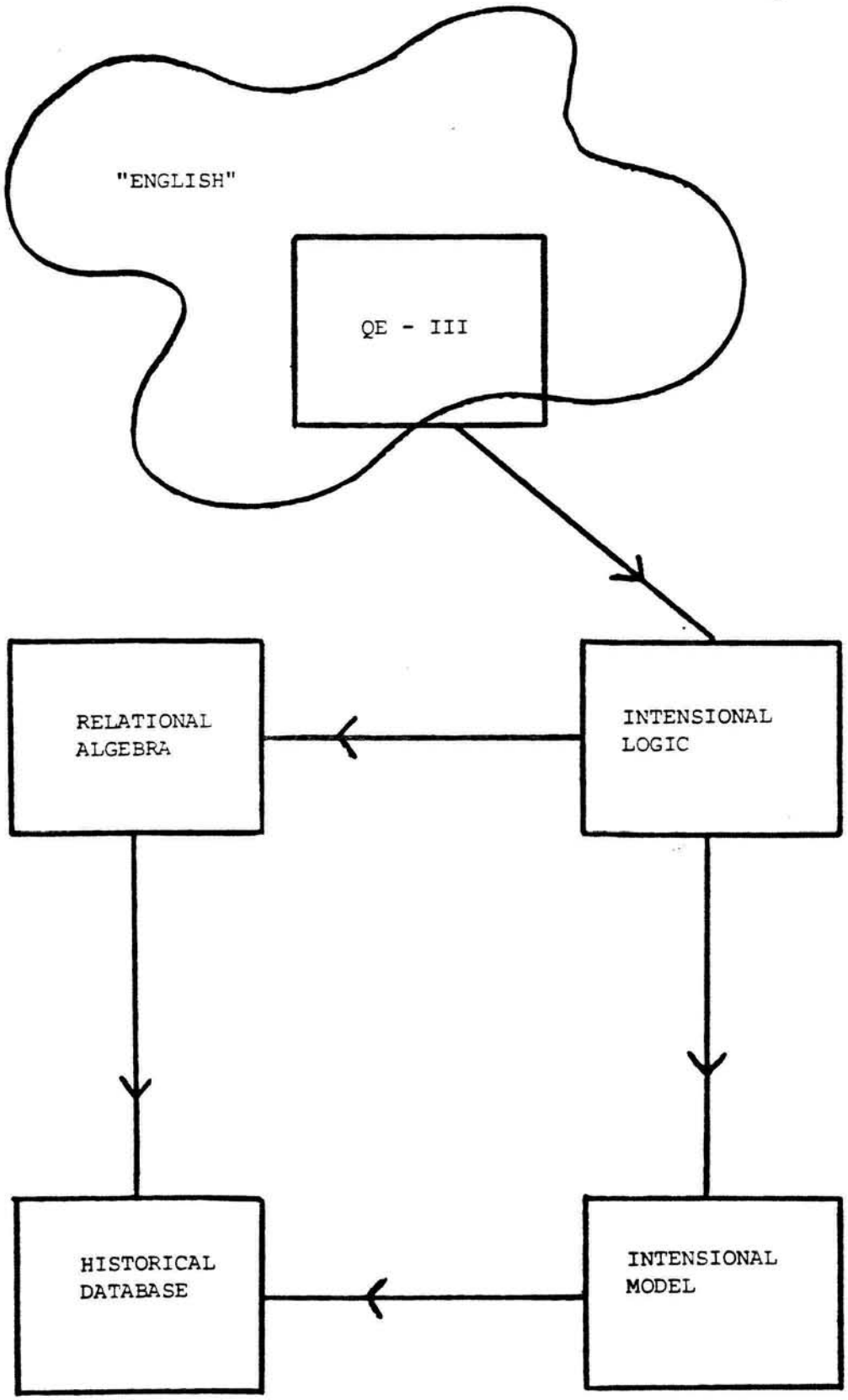
relation item-rel

SALES-REL

STATE	DEPT	ITEM	EXISTS?	VOL
S1	Toy	Ball	1	3
S1	Toy	Game	1	6
S1	Hardware	Glove	1	9
S1	Linen	Glove	0	⊥
S2	Toy	Ball	1	3
S2	Toy	Game	1	6
S2	Hardware	Glove	1	9
S2	Linen	Glove	1	2
S3	Toy	Ball	1	4
S3	Toy	Game	1	6
S3	Hardware	Glove	0	⊥
S3	Linen	Glove	0	⊥

relation sales-rel

Figure 8



OVERALL FRAMEWORK

Figure 9