TOWARDS AN ALGEBRA OF HISTORICAL RELATIONAL DATABASES

James Clifford

December 1984

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

Working Paper Series

CRIS #86

#84-91(CR)

Presented at ACM SIGMOD 1985, Austin, Texas.

Table of Contents

Towards An Algebra of Historical Relational Databases

Abstract. In search of the appropriate semantics for the inclusion of structures and operations that will meet the needs of a wide class of users interested in a database system supporting temporal views of their data, the paper includes a discussion of many problems that must be addressed. Salient features of the author's Historical Relational Database Model (HRDBM) are presented, and some subtle nuances that time brings to the development of an historical relational algebra are illustrated. Along the way, a number of observations and guidelines are presented that may help guide the search for an historically relationally complete database model and query languages.

KEYWORDS: historical databases, relational data model, relational algebra, query languages, relational completeness, historical relational completeness.

1. Introduction

In a previous paper [Clifford 83] we presented the Historical Database Model (HDBM), a theory of the semantics of an extended relational database model having time as a fundamental organizing principle. The present paper continues to explore such an HDBM, but from an operational perspective, in contrast to the denotational semantic view of the previous work. Whereas we previously defined a logical model theory for an HDB, we are here interested in defining a relational algebra for HDBs. This has proven more difficult than we imagined, not because of technical difficulties, but because of subtle semantic nuances caused by the introduction of time into the model. We thought that it might prove more informative, not to present a formal historical relational algebra in full detail, but rather to explore some of these issues, as they arise in our work and in the work of others in the area, and discuss notions relating to the correctness and completeness of such an algebra. By proceeding in this fashion we may open the door to more discussion of the desired properties and functionality of an HDBM before, perhaps prematurely, becoming bound to a fixed system definition.

There are a number of goals that we believe can provide some perspective in our search for the "right" model:

1. It should, if at all possible, be a consistent extension of the traditional relational model. Thus, flat relations and the familiar relational algebra should be treated as a special case of the historical relational model.

2. A corollary to this point, but one that bears emphasizing, is

that the proposed historical relational algebra be, in fact, an algebra. This is clearly one of the basic properties of the relational model, and one which we must be maintained.

3. Less formalizable, but no less important, is the goal of semantic completeness; the model should be so natural as to accord with people's intuitive views of time and information over time, and so powerful as to allow the extraction of the temporal information that it contains. In this second sense it should be able to serve as a standard for defining the notion of "historical relational completeness," comparable to Codd's notion of "relational completeness."

4. It must adequately address the issue of succinctness, or minimality, with respect to the temporal dimension, i.e., the structures seen by the user should contain only information at points in time when users record a change, yet the operations should allow the automatic inference of values at other time points "covered" by the database.

5. Given the pervasiveness of the three-dimensional spatial metaphor for historical databases (the database as "cubes"), the model should if possible maintain this view at the external user's level.

This part of the paper is organized as follows. In Section 2 we discuss briefly some related work in the area. Section 3 presents salient features of the model that we have developed so far, along with a discussion of the reasons for some of the decisions we have made. Section 4 then discusses many of the subtleties inherent in a temporally-oriented database, in the context of developing extensions to the basic relational operations (project, select, and join) and two new operations, "time-slice" and "when." Finally in Section 5 we discuss directions for future research. A running example throughout the text is a film-lover's database (Appendix, with apologies for errors, omissions and fabrications) significantly more rich in temporal information than many examples that stand behind some model proposals.

In discussing the issue of time and databases, we not infrequently encounter a "so what?" attitude: isn't it obvious that if you need to, you time-stamp your data. Two points can be made in response to this attitude. First, a point we have made before, none of the three great data models provides any mechanism at all for organizing temporally-oriented data. It is the purpose of this paper to present the second point, namely, that time is something so taken for granted that its exact nature is highly elusive. Thus, while it is not technically difficult to come up with a consistent model having various algebraic operations defined, intuitively it is far from obvious which operations are appropriate, meaningful, and correct. As Augustine of Hippo long ago observed: "What, then, is time? If no one asks me, I know; but, if I want to explain it to a questioner, I do not know." (Confessiones XI, XIV).

2. Previous Work

Although there are a number of researchers active in the area of temporal database models ([Bolour et al. 82] and [Ariav 83a] respectively survey the literature and delineate the major research areas) to date there has been no really comprehensive treatment of an operational view of these databases. There have, however, been some attempts in this direction. In the Time Relational Model (TRM) of [Ben-Zvi 82] a temporal dimension is added to ordinary relations, in a manner similar to the HDBM. Then an algebra for TRM is defined which, although internally consistent, is extremely limited. Essentially, all the operators on time-relations (3-dimensional) specify a "Time-View"

which "extracts a regular (two-dimensional) relation out of the Time-Relation." After this "one-shot" reference to the temporal dimension, the user must then work on this "view" within the ordinary relational model. A true time-relational algebra, which would stay within the the space of time relations, is not defined.

In [Ariav 83b] an extension to the query language SQL was defined, in an effort to incorporate temporal elements in a way that would not penalize users not interested in access to the historical data. Recent work in [Snodgrass 84] reports on a similar extension to the query language QUEL. One problem with this work is that its model contains two different types of objects, event relations and interval relations. This unduly complicates the model, particularly the definitions of the query language. Neither of these papers addresses the formulation of an historical algebra, so that many operational issues that the algebra forces you to consider are not treated. An unpublished manuscript [Ariav 84] looks at the issue of an algebra of historical relations, but because of certain assumptions built into their structure, have run into a number of problems. For instance, they have been unable to define a join that is symmetric (where A|X| B = B |X| A). [Lum 84] addresses the issue of the underlying data structures to support relational operations, proposing time-stamping of tuples and keeping current tuples in a relation. History tuples "belonging" to a current tuple are chained together in reverse time order. Dadan, Lum, and Weiner continue to explore implementation strategies for this model [Dadan 84].

In all of these studies, relations are viewed as 3-dimensional cubes, the third dimension being the time. The 3-dimensional structure is converted to relational tables (2-dimensional) by time-stamping the tuples. [Clifford 82a], however, was the first to suggest a different view, one closer to the view of the intensional logic which provided the original impetus for the model in [Clifford 82b] and [Clifford 83]. In this view time is incorporated into the relational model, not at the level of the tuples, but at the level of the attributes. In attempting to define an operational view of relational HDBs, the author has become even more convinced of the correctness of this treatment of time in historical relational databases.

3. The HDB model

3.1. "What, then, is time?"

Perhaps the best place to start in an enterprise aimed at adding time into a database model is in an examination of this new object, time. What kinds of objects are we going to allow users to treat as members of the set of times, and what properties will this set have? These are basic questions to any historical database model. About one property there has been little dissension: the set Time is a linear order, i.e., for any two times t_1 and t_2, either t_1 equals t_2, t_1 is-less-than t_2, or t_2 is-less-than t_1. Indeed this ordering is perhaps time's essential property. As to the members of the set, there has been less agreement. In [Clifford 83] it was treated as dense, essentially isomorphic to the set of reals. For two reasons, we now prefer to treat time as discrete,

and isomorphic to the natural numbers. First, it is clear that any recording instrument must have at best a finite sampling quantum, and second, any practical domain (or language) that we might define for time attributes in an HDB would have at most a countably infinite set of names for time moments or time intervals. Thus while it may be philosophically or theoretically interesting to consider a continuum of moments of time, from a practical standpoint the natural numbers seem a more useful candidate for modelling the properties of database time.

A few other general properties of the set Time will emerge as we discuss some of the operations, but in general we adopt the position that the model should say as little as possible about Time except for properties that are essential to its proper use. The specific elements of Time are best left for the user to define. Others, for example [Anderson 81], have adopted a somewhat different view, and have explored various additional properties of calendar systems, etc.

3.2. Where does time fit into the model?

Given some structure for the set Time, which for the moment we will take to be <T,BEFORE>, where T is some countable set and BEFORE a linear order on T, the question of how best to fit this structure into the relational model naturally arises. In most of the work to date, (e.g., [Klopprogge 81], [Clifford 83], [Snodgrass 84], [Ariav 83b], [Ben-Zvi 82]) some form of tuple time-stamping has been adopted. In attempting to define an operational view of HDBs we have become convinced of the correctness of the view we first proposed in in [Clifford 82a], which treats time as a component of the attributes rather than the tuples.

Consider the following relation EREL, where each tuple is time-stamped, and consider how we might define the "projection" operator for such an organization.

EREL (EMP STATE SAL DEPT)

Peter	1.1.80	30K	Shoe
Peter	4.6.80	32K	Shoe
Margi	6.3.78	30K	Shoe
Margi	1.1.79	31K	Shoe
Margi	4.6.80	32K	Shoe
Jack	4.7.79	30K	Linen
Jack	4.12.80	30K	Shoe

Two possibilities readily suggest themselves. For example, consider the projection of this relation onto the SAL attribute. If we simply project out this column, we get the two-dimensional relation on the left; if we want to remain within the space of historical relations, and project both the STATE attribute (by fiat) and the specified attribute (SAL), we get the historical relation on the right:

(SAL)
30K
32K
31K

(STATE	SAL)
1.1.80	30K
4.6.80	32K
6.3.78	30K
1.1.79	31K
4.7.79	30K
4.12.80	30K

The first method is clearly worse; we have no idea when these amounts were earned, if they are all current salaries (no), if they are related in some way (yes, but how?), etc. The second technique is considerably better. More information is present (although one of the two duplicate tuples <4.6.80,32K> was eliminated in making the result a set), but we still are not given any relationships among the tuples. We have lost

what, in intensional logical terms, are called the "individual concepts" (ICs) -- the salary "of" Margi, the salary "of" Peter, etc. These ICs are functions from times to dollar amounts, and projecting in this fashion loses the functions, or the "of-ness" of salaries. Non-key attributes are always related in this "of" way to the "object" given by the key value. The "pure" relational model did not have to deal with this, essentially semantic, point, although the RM/T extension [Codd 79], and other semantic extensions to the RM have had to consider it (e.g., [Sciore79 79], or [HammerMcleod78 78].) It appears that a proper treatment of HDBs must do likewise. (Note that a projection onto DEPT might seem to avoid this issue; after all, {<Shoe>,<Linen>} appears to be a reasonable result. But again, we have lost the "of-ness" of the use of the DEPT attribute in EREL; if we want information about departments, we should probably be examining a different relation, one "about" departments.

There are other compelling reasons for considering STATE as a component of the attributes, including the following:

1. different attributes may be measurable/recordable at different rates (days, months, seconds,...);

2. some attributes are inherently not time-varying (e.g., BIOLOGICAL-GENDER), and should not be encumbered with a tuple's time stamp;

3. attributes vary over time in different ways (how many is open question), e.g. continuous functions (e.g., TEMPERATURE), aggregates over an interval (e.g., SALES-VOLUME), and step functions (e.g., MANAGER);

4. when a change occurs, it is generally to the value of an

individual attribute, not to the values of all of the attributes in a tuple.

There are at least two important ramifications of this view. The first is that relations in our model are no longer in First Normal Form, since the domain for time-varying attributes is non-simple. (However, the structure of domains is precise, highly constrained, and exploited by the algebraic operations.) The second is that with attributes differing along various time-related dimensions, it becomes necessary to define some underlying "basic" view of time for the database. This is necessary in order for the enterprise to be modelled with a consistent view of time; it becomes crucial when we consider, as in the join operation, the interaction between relations.

4. Overview of the model

In the model that we have developed, an historical database consists of a collection of historical relations, each one defined over some interval of time (typically, but not necessarily, beginning at some point t and continuing to the present, or NOW). (The examples in the Appendix will be referred to as we present the model and issues of time.)

The set Time is a set of times, $\{t_1, t_2, \ldots\}$ that is at most countably infinite, with a linear order BEFORE, i.e. BEFORE $= \{ \langle t_i, t_j \mid t_i \text{ is before } t_j \rangle \}$. This set serves as the underlying domain of times for the entire database.

In order to provide a full treatment of time, we have found it necessary to distinguish between three different kinds of attributes:

1. Constant attributes (CA), such as BLOOD_TYPE in DIRECTORS, which are time invariant and hence have simple domains.

2. Time-varying attributes (TVA), such as STUDIO in DIRECTORS, which can potentially vary over time; the domains of these attributes are functions from Time to some simple domain.

3. Temporal attributes (TA), such as YEAR in FILMS, whose domain is Time.

These are called the temporal type of an attribute. The notion of CAs is of course related to the notion of the relation key: all attributes in a relation key must be CAs; the other non-key attributes can be of any temporal type. Also, it is important to note that the temporal type of an attribute is dependent upon the particular relation in which it appears (for instance, STUDIO in DIRECTORS is a TVA, but in STUDIOS it is a CA). We also point out that the inclusion of TAs (as in the relation FILMS) provides a solution to the problems addressed, e.g. in the use of two different types of relations, event relations and interval relations, in [Snodgrass 84]. We believe that this greatly simplifies the model and its operations.

In modelling objects and their properties over time, an essential property to consider is the lifespan of an object, i.e., the period of time during which the object and its properties are being modelled in the system. Every historical database system must address this fundamental issue ([Klopprogge 81], [Clifford 83]). In our model we

assume that each relation has an associated lifespan which is a complete segment [START,END] of the set TIME; END is typically the present moment, NOW. As in [Clifford 83] (the Comprehension Principle), we assume that a base relation R has complete information about the "objects" that R models over its lifespan. Since objects can freely move into and out of our scope of interest (employees are hired, fired, rehired, etc.), we must also provide for modelling the lifespan's of individual objects. While there are several possible solutions to this problem, for the moment we are inclined to utilize a "does not exist" null value. Moreover, there is strong reason to believe that a "value unknown" null will be needed, since it is highly likely that in historical databases incomplete information, especially about the past, will abound. Accordingly our model uses three different null values:

1. $NULL_1$: value of TVA becomes unknown at this point in time
2. $NULL_2$: value of TVA becomes non-existent at this point in time
3. $NULL_3$: value of TVA is unknown at any points in time

The two-dimensional relational model provides two basic operations for reducing relations along each dimension: select along the "object" (or tuple) dimension, and project along the attribute dimension. The third basic operation is the join, for combining two relations. It should come as no surprise, therefore, that in extending this model to three-dimensional relations, we will need, in addition to these operations, a reducing operation, which we call time slice, for the new third dimension. Most of the fundamental problems in building an

historical model can be illustrated by examining potential definitions for these operations.

4.1. Projection

On first glance, projection(π) appears easy to extend; after all, we have not really altered the attribute dimension of our model. So, for instance,

$\pi_{NAME,STUDIO}$ (DIRECTORS) yields the relation:

(NAME	STUDIO)
Sternberg	1924 --> MGM
	1926 --> Paramount
	1935 --> Columbia
	1938 --> MGM
	1952 --> $NULL_1$
Cukor	1930 --> Paramount
	1932 --> RKO
	1939 --> MGM
	1950 --> $NULL_1$
Hitchcock	1927 --> Brit. Intl.
	1934 --> Gaumont
	1938 --> Gainsborough
	1939 --> RKO
	1951 --> Warner Br.
	1954 --> Paramount
	1959 --> MGM
	1960 --> Paramount
	1962 --> Universal
	1964 --> $NULL_1$

However, consider the following projection, which discards all of the TVAs in its operand:

$\pi_{NAME,BLOOD\_TYPE}$ (DIRECTORS) yields the relation:

(NAME	BLOOD_TYPE)
Sternberg	A
Cukor	O

Hitchcock AB

Although this looks satisfactory, we must consider the question whether it is any longer an historical relation. If it is not, than we have abandoned our goal of defining an algebra, and if it is, then we will have to be especially careful in defining our other operations. For example, what would be the result of taking a TIME-SLICE of this relation at the time 1934? or at the time NOW? This issue, of nailing down our definition of the structures in an historical relational model, will be addressed again when we consider some other operations.

4.2. Selection

Since we have expanded our notion of domains to include both simple domains, and structured domains (functions from TIME to a simple domain), we can expect that the definition of select (σ), which reduces along the value dimension, will be significantly affected. In fact, although most people have felt that JOIN was the difficult operation, most of the problems that arise in defining an historical relational model can be illustrated in considering σ, which will be easier since we can look at only one relation. (Since σ can simulate a JOIN, it is no surprise that both operations address similar problems.) We will consider seven examples.

Select Example 1: Select on CA

$\sigma_{(NAME = Hepburn)}$ (STARS)

(NAME	DIRECTOR	BLOOD_TYPE)
Hepburn	1932 --> Cukor 1938 --> Hawks	A

1940 --> Cukor
1953 --> NULL

Select Example 2: Select on TA

σ(YEAR = 1935) (FILMS)

(FILM	STUDIO	YEAR)
The Devil is a Woman	Paramount	1935
Sylvia Scarlett	RKO	1935

Since both CAs and TAs are simple domains, they require no change to the standard definition of σ. TVAs, however, present special problems. There possible values can be given by the user as the selection criteria: the value of a TVA at a specified time (Exs. 3, 4 & 5), the value of a TVA for some time (Ex. 6), or (rarely), the value of the TVA for the object's entire lifespan (Ex. 7).

Select Example 3: Select on TVA,
Member of Function Given (1)

σ(STUDIO(1937) = MGM) (LAWYERS)

(LAWYER	STUDIO	SALARY)
Howell	1924 --> MGM	1924 --> 30K
	1930 --> Paramount	1925 --> 35K
	1937 --> MGM	1937 --> 40K
	1940 --> $NULL_1$	1940 --> $NULL_1$

In this example there are no difficulties; Howell is the only lawyer working for MGM in 1937, and both of his TVAs are explicitly defined for this time.

Select Example 4: Select on TVA,
Member of Function Given (2)

σ(STUDIO(1925) = MGM) (LAWYERS)

(LAWYER	STUDIO	SALARY)

Howell	1924 --> MGM	1924 --> 30K
	1930 --> Paramount	1925 --> 35K
	1937 --> MGM	1937 --> 40K
	1940 --> $NULL_1$	1940 --> $NULL_1$

This example highlights the need for an interpolation function for TVAs, called a Continuity Assumption in [Clifford 83]. Users must be able to query the database at will with respect to time points or periods, and yet the database cannot possibly store values for every attribute at every point in time. Thus, each attribute must have an associated interpolation function, so that the database system can reconstruct an entire time series over the lifespan of each object from the partial specification stored. As discussed in [Clifford 83], a very common interpolation function interprets a partial specification as a step function.

Select Example 5: Select on TVA,
Members of Function Given (3)

σ(STUDIO([1925,1940]) = MGM) (LAWYERS)

(LAWYER	STUDIO	SALARY)
	Ø	

In this example a range of times is specified; only those lawyers who worked for MGM throughout the interval [1925,1940] are requested, and in this instance none fit the bill.

Select Example 6: Select on TVA,
Value from Range Given

σ(STUDIO = Warner Br.) (LAWYERS)

(LAWYER	STUDIO	SALARY)
Rosen	1912 --> Universal	1945 --> 70K

	1923 --> Warner Br.	1953 --> $NULL_1$
	1930 --> $NULL_1$	
	1945 --> RKO	
	1953 --> $NULL_1$	
McManus	1923 --> Warner Br.	1923 --> 35K
	1930 --> $NULL_1$	1926 --> 40K
		1930 --> $NULL_1$

In this example the user can ask for all tuples that have the value of Warner Br. for any point in time in their STUDIO function; it mirrors an existential quantifier over times.

Select Example 7: Select on TVA,
Entire Function Given

σ (STUDIO = 1923 --> Warner Br.
1930 --> $NULL_1$) (LAWYERS)

(LAWYER	STUDIO	SALARY)
McManus	1923 --> Warner Br.	1923 --> 35K
	1930 --> $NULL_1$	1926 --> 40K
		1930 --> $NULL_1$

For completeness, a σ completely analogous to the traditional σ can be defined, where the user gives the entire value for a TVA and asks for all tuples having this value. This flavor would be extremely rare, and syntactically quite messy. Note that it is the only flavor involving TVAs which could be defined without reference to the function structure of the TVA domain.

4.3. Time Slice

In a sense Time Slice (τ) is a kind of σ, in which a value from the domain of a TVA is given. However, it is more general in that it allows the selection across all of the attributes in the relation. It is for this reason that it deserves treatment as an independent operator across a third dimension. Again, some examples will illustrate the problems in defining an appropriate semantics.

TIME SLICE Example 1: A Point in Time (1)

$\tau_{(1938)}$ (DIRECTORS)

(NAME	STUDIO	BLOOD_TYPE)
Sternberg	1938 --> MGM	A
Cukor	1938 --> MGM	O
Hitchcock	1938 --> Gainsborough	AB

As pointed out earlier in the discussion of Projection, there is a problem with reducing historical relations to flat ones. The Continuity Assumption and Comprehension Principle were defined to allow us to simulate three-dimensional relations, information complete over some time interval, out of much simpler stored relations; we must not fall into the trap of defining operators that subvert this end. Specifically, this relation is less information-bearing than the base relation from which it is derived; we must not allow operations to apply the interpolation function of the base relation and incorrectly infer, for example, that Hitchcock worked for Gainsborough in 1939. This issue has been overlooked in any of the system proposals to date. Our current solution is to attach a lifespan to each relation, and to have the

operands of the algebraic operations be relations and their lifespans. The lifespan of a relation is simply an interval $[t_1,t_2]$ over which, as in the Comprehension Principle, the relation is assumed to be completely defined. Thus the result of an operation is always a relation with a (possibly new) lifespan. Interpolation functions can be applied only over the lifespan of a relation; in this case the lifespan of the result is [1938,1938], and no problem need arise.

TIME SLICE Example 2: A Point in Time (2)

$\tau(1927)$ (DIRECTORS)

(NAME	STUDIO	BLOOD_TYPE)
Sternberg	1927 --> Paramount	A
Cukor	1927 --> $NULL_1$	O
Hitchcock	1927 --> Brit. Intl.	AB

TIME SLICE Example 3: An Interval of Time

$\tau([1923,1935])$ (LAWYERS)

(LAWYER	STUDIO	SALARY)
Howell	1924 --> MGM	1924 --> 30K
	1930 --> Paramount	1925 --> 35K
Rosen	1923 --> Warner Br.	$NULL_3$
	1930 --> $NULL_1$	
McManus	1923 --> Warner Br.	1923 --> 35K
	1930 --> $NULL_1$	1926 --> 40K
		1930 --> $NULL_1$

This example highlights the need for understanding the difference between the lifespan of a tuple and of a relation. the lifespan of this result relation is, of course, the interval of the time slice,

[1923,1935]; however, the lifespan of, for instance, the tuple with key Howell, is [1924,1935]. Notice also that not all attributes need have values in a tuple (e.g., SALARY in Rosen tuple) for the result to be meaningful -- hence the system must be able to generate null values in a result.

TIME SLICE Example 4:
Interpolation vs Non-interpolation

$\tau_{(1925)}$ (STUDIOS)

(STUDIO	HEAD	NUM_FILMS)
MGM	1925 --> Mayer	1925 --> 10
Paramount	1925 --> Schulberg	1925 --> 12
Warner Br.	1925 --> J. Warner	$NULL_3$
Universal	1925 --> Laemmle	$NULL_2$

A point that is also overlooked in existing models is that there are different ways of modelling data over time, and these need to be reflected in different types, or properties, of TVAs. In this example, the attribute HEAD is interpolatable, and uses a simple step-function interpolation. However, the attribute NUM_FILMS is inherently non-interpolatable. From the value of NUM_FILMS at a given time point, we cannot infer anything about its value at any other time point. Note also that there has to be a built-in semantics of the different nulls (e.g., τ of $NULL_3$ should probably give $NULL_3$, but $NULL_2$ is generated for the case of non-interpolatability).

TIME SLICE Example 5:
Disaggregatable vs Non-disaggregatable Attrs.

$\tau_{(7/12/25)}$ (STUDIOS)

(STUDIO	HEAD	NUM_FILMS)

MGM	7/12/25 --> Mayer	$NULL_2$
Paramount	7/12/25 --> Schulberg	$NULL_2$
Warner Br.	7/12/25 --> J. Warner	$NULL_2$
Universal	7/12/25 --> Laemmle	$NULL_2$

This final example illustrates the need for flexibility in the user interface, and the problems this engenders. Users should be allowed freedom to refer to time in ways that are natural to do so (in this instance, referring to a specific day), and the system should be able to (a) translate between different time representations and (b) know when values associated with larger intervals of time (years, e.g.) can correctly be associated with smaller intervals of time (days, e.g.) This requires an additional piece of information associated with an attribute, namely whether or not it is disaggregatable. In this example, the attribute HEAD can be disaggregated, but the attribute NUMFILMS clearly cannot; hence the result as shown.

4.4. JOIN

Most of the problems in understanding the meaning of historical JOINs are not essentially different from the problems we encountered with SELECT and TIME-SLICE, so we will not dwell on them here. However, we point out that in joining R1 with R2 the following cases need to be considered:

1. R1 and R2 have no attributes in common (Cartesian Product should result).
2. shared attributes are both CAs.
3. shared attributes are both TVAs.
4. shared attributes are TV and C.

5. shared attributes are TV and T.

One additional issue that arises with the JOIN is the interval over which the join is to be performed. Given that the lifespan of R_1 is $I_1 = [t_1,t_2]$ and of R_2, $I_2 = [t_3,t_4]$, there are two possible cases to consider for the lifespan of the result: either the intersection or the union of I_1 and I_2. Since both of these can have a reasonable interpretation, it seems appropriate to define both a UNION-JOIN and an INTERSECTION-JOIN.

4.5. WHEN

Finally, an additional time-related operator called WHEN (Ω), is introduced to provide a mechanism for naming time values not simply with constants (like 1983) but with expressions (like WHEN A = v in r). This unary operator on relations, unlike the other relational operators, yields as a result a set of times rather than a relation. It is used to form temporal expressions which can serve as components of a τ or σ operation.

In the simplest case the result of Ω is a connected interval, as in the first example which yields the times when Cukor headed Paramount:

WHEN Example 1: Interval Result

$\Omega_{(\text{STUDIO=Paramount, Head=Cukor})}\ (\text{STUDIOS})$

$= [1919,1925]$

Note that with this operator a query such as "Who were the lawyers at Warner Br. when Cukor headed Paramount" is easily expressed:

π_{LAWYER} (σ(Studio=Warner Br.)
(τ(Ω(STUDIO=Paramount, Head=Cukor) (STUDIOS))
(LAWYERS)))

In general, however, the result of Ω may be a set of disconnected intervals, as in the query for the times when Hitchcock directed for Paramount:

WHEN Example 2: Disconnected Intervals Result

Ω(STUDIO=Paramount, Name=Hitchcock) (DIRECTORS)
= {[1954,1959],[1960,1962]}

This implies that τ and σ must be able to deal with temporal arguments which are sets of intervals; for example, τ must be able to time slice a relation at disjoint intervals, producing null values for the unselected time points.

5. Summary

We have presented a discussion of issues and problems related to the subtle interactions of time with the other components of the relational database model. These issues must be addressed by any system that hopes to provide a complete range of temporal structures and operations, or, in short, that merits being called an historical database model. We have presented the basic properties of a model that we are building to support a temporal view of data, and hope to spur further research into some of the problems we have addressed.

Acknowledgements

I would like to acknowledge the insights into many of these problems that have been generated in numerous discussions with my colleague Gadi Ariav and our student Joseph Shiftan.

Appendix

STUDIOS (STUDIO	HEAD	NUM_FILMS)
MGM	1924 --> Mayer	1924 --> 6
	1948 --> Schary	1925 --> 10
	1956 --> $NULL_2$	1970 --> 15
	1970 --> Aubrey	
	1974 --> $NULL_1$	
Paramount	1919 --> Cukor	1919 --> 2
	1925 --> Schulberg	1925 --> 12
	1935 --> $NULL_2$	1936 --> 10
RKO	1945 --> Schary	1945 --> 10
	1948 --> Hughes	1946 --> 11
	1957 --> $NULL_2$	1947 --> 12
Warner Br.	1923 --> J. Warner	
	1969 --> Ashley	$NULL_3$
	1972 --> $NULL_1$	
Universal	1912 --> Laemmle	1930 --> 6
	1936 --> Blumberg	1937 --> 9
	1946 --> Spitz	1965 --> 11
	1952 --> Rackmil	
	1955 --> Hunter	
	1965 --> Wasserman	

STARS (NAME	DIRECTOR	BLOOD_TYPE)
Dietrich	1930 --> Sternberg	O
	1937 --> Lubitsch	
	1948 --> Wilder	
	1950 --> Hitchcock	
	1957 --> Wilder	
	1958 --> Welles	
	1959 --> $NULL_1$	
Grant	1932 --> Sternberg	AB
	1935 --> Cukor	
	1938 --> Hawks	
	1940 --> Cukor	
	1941 --> Hitchcock	
	1943 --> NULL	
	1946 --> Hitchcock	
	1960 --> NULL	
Hepburn	1932 --> Cukor	A
	1938 --> Hawks	
	1940 --> Cukor	
	1953 --> NULL	

DIRECTORS

(NAME	STUDIO	BLOOD_TYPE)
Sternberg	1924 --> MGM	A
	1926 --> Paramount	
	1935 --> Columbia	
	1938 --> MGM	
	1952 --> $NULL_1$	
Cukor	1930 --> Paramount	O
	1932 --> RKO	
	1938 --> MGM	
	1950 --> $NULL_1$	
Hitchcock	1927 --> Brit. Intl.	AB
	1934 --> Gaumont	
	1938 --> Gainsborough	
	1939 --> RKO	
	1951 --> Warner Br.	
	1954 --> Paramount	
	1959 --> MGM	
	1960 --> Paramount	
	1962 --> Universal	
	1964 --> $NULL_1$	

FILMS

(FILM	STUDIO	YEAR)
The Blue Angel	Paramount	1930
Touch of Evil	Universal	1958
Angel	Paramount	1937
Stage Fright	Warner Br.	1950
Witness for the Prosecution	Un. Artists	1957
A Foreign Affair	Paramount	1948
Dishonored	Paramount	1931
Shanghai Express	Paramount	1932
Blonde Venus	Paramount	1932
The Scarlet Empress	Paramount	1934
The Devil is a Woman	Paramount	1935
Suspicion	RKO	1941
North By Northwest	MGM	1959
Notorious	RKO	1946
Sylvia Scarlett	RKO	1935
Bringing Up Baby	RKO	1938
The Philadelphia Story	MGM	1940
A Bill of Divorcement	RKO	1932
Little Women	RKO	1933
Adam's Rib	MGM	1949

LAWYERS (LAWYER	STUDIO	SALARY)
Howell	1924 --> MGM	1924 --> 30K
	1930 --> Paramount	1925 --> 35K
	1937 --> MGM	1937 --> 40K
	1940 --> $NULL_1$	1940 --> $NULL_1$
Rosen	1912 --> Universal	1945 --> 70K
	1923 --> Warner Br.	1953 --> $NULL_1$
	1930 --> $NULL_1$	
	1945 --> RKO	
	1953 --> $NULL_1$	
McManus	1923 --> Warner Br.	1923 --> 35K
	1930 --> $NULL_1$	1926 --> 40K
		1930 --> $NULL_1$

References

[Anderson 81] Anderson, T.L.
Database Semantic of Time.
PhD thesis, Computer Science Dept., U. of Washington, 1981.
(Unpublished).

[Ariav 83a] Ariav, G., Clifford, J., and Jarke, M.
Time and Databases.
In ACM-SIGMOD International Conference on Management of Data, pages 243-245. May, 1983.

[Ariav 83b] Ariav, G.
Preserving the Time Dimension in Information Systems.
Technical Report DS-WP 83-12-06, Decision Sciences Dept., Univ. of Penn., December, 1983.
(Ph.D. Thesis).

[Ariav 84] Ariav, G., Beller, A., and Morgan, H.L.
A Temporal Model.
Technical Report DS-WP 82-12-05, Decision Sciences Dept., Univ. of Penn., December, 1984.

[Ben-Zvi 82] Ben-Zvi, J.
The Time Relational Model.
PhD thesis, Dept. of Computer Science, University of California, Los Angeles, 1982.
(Unpublished).

[Bolour et al. 82]
Bolour, A., Anderson, T.L., Deketser, L.J., Wong, H.K.T.
The Role of Time in Information Processing: A Survey.
ACM SIGMOD Record 12(3):28-48, April, 1982.

[Clifford 82a] Clifford, J.
A Model for Historical Databases.
In Proceedings of Logical Bases for Data Bases.
Toulouse, France, December, 1982.

[Clifford 82b] Clifford, J.
A Logical Framework for the Temporal Semantics and Natural-Language Querying of Historical Databases.
PhD thesis, Dept. of Computer Science, SUNY at Stony Brook, December, 1982.
(Unpublished).

[Clifford 83] Clifford, J., and Warren D.S.
Formal Semantics for Time in Databases.
ACM Trans. on Database Systems 6(2):214-254, June, 1983.

[Codd 79] Codd, E.F.
Extending the Database Relational Model to Capture More Meaning.
ACM Trans. on Database Syst. 4(4):397-434, December, 1979.

[Dadan 84] Dadan, P., Lum, V.., and Werner, H.D.
Integration of Time Versions in Relational Database Systems.
In Proc. of The Tenth International Conference on Very Large Data Bases, pages 509-521. 1984.

[HammerMcleod78 78]
Hammer, Michael,and Mcleod, Dennis.
The Semantic Data Model: A Modelling Mechanism for Data Base Applications.
In Proceedings of the ACM SIGMOD Conference. Austin, 1978.

[Klopprogge 81]
Klopprogge, M.R.
Term: An Approach to Include the Time Dimension in the Entity-Relationship Model.
In P.P.S. Chen (editor), Entity-Relationship Approach to Information Modeling and Analysis, pages 477-512. ER Institute, 1981.

[Lum 84] Lum, V., et al.
Designing DBMS Support for the Temporal Dimension.
In Proceedings of the ACM SIGMOD Conference, pages 115-126. Boston, June, 1984.

[Sciore79 79] Sciore, Edward.

Improving Semantic Specification in a Relational Database.

In Proceedings of the ACM SIGMOD Conference. Boston, 1979.

[Snodgrass 84] Snodgrass, R.

The Temporal Query Language TQuel.

In Proceedings of the 3rd ACM SIGMOD Symp. on Principles of Database Systems, pages 204-212. Waterloo, Ontario, Canada, April, 1984.