Models in Temporal Knowledge Representation and Temporal DBMS

by

Renato Barrera and Khaled K. AI-Taha

National Center for Geographic Information & Analysis
and
Department of Surveying Engineering
University of Maine
Orono, ME 04469

RENATO@MEECAN1.bitnet
KALTAHA@MECAN1.bitnet

National Center for Geographic Information & Analysis/NCGIA

Report 90-8

Preface

Geographic information systems will in the future contain data that has a spatial as well as a temporal reference. The National Center for Geographic Information and Analysis has selected the problem of temporal data as the focus of one of its research initiatives.

Preliminary work for this research initiative started by developing an overview of the existing literature, gathering the few papers written on temporal GIS topics, and then expanding to Artificial Intelligence (AI) and Database literature. We assume that at least some temporal aspects of a GIS with temporal data will be similar to the problems of temporal data in other applications and the results of work done in other areas applicable to GIS.

Specifically, we expect the Al research to help us understand the problems of temporal reasoning (how do we draw conclusions from time related facts) and the conceptual representation of temporal data. Al systems typically address problems with a small number of facts or rules. GIS will contain very large amounts of data (especially if one considers the currently discussed databases to monitor "global change") and we, therefore, also need to consider the work done in the database community, where problems of structuring and accessing large collections of temporal data were studied.

In this report, we present the results we found in the published literature. Of course, we may have overlooked some relevant papers and would be glad if they were brought to our attention. We present it here as "materials" that may help others. We have not yet established the connection between the results found and their applicability to the temporal GIS situation. The examples used here are, with very few exception, not GIS related. Nevertheless, the material should be useful for advanced researchers trying to deal with temporal GIS problems.

Andrew U. Frank

Abstract

The concept of time arises in various fields of interest, such as philosophy, psychology, linguistics, artificial intelligence (AI), and database management systems (DBMS). For the wide range of areas concerned about time, the framework of a research discipline emerged, which is often called *temporal reasoning.* Temporal information is essential to most applications for many reasons: future prediction and planning, observation of rules that cause changes in the world, and analysis and explanation of the present state of the world. Models for temporal reasoning implement either change-based approach or a time-based approach. We briefly describe some change-based models and their limitations; we also describe in more detail three time-based models, namely, the interval -based, the point-based, and the mixed models. We also present a survey of the literature in temporal database management systems DBMS from the early 1970's to 1988 and discuss data models, temporal attributes, temporal query languages, and physical concepts in DBMS. Finally, we compare the temporal reasoning models with the DBMS models and present the major differences in the two fields of interest.

Table of Contents

Chapter 1: Introduction

1. 1 Time in an Information System

When we first learned the German language, the teacher gave us the general and ancient Roman's rule for how to order the pieces of information presented in one sentence, which says: "Te, Ca, Mo, Lo," which means: "Tempus, Causus, Modus, Locus." i.e., a complete and descriptive sentence will answer the questions in the following order: **When? Why? How?** And **Where?** This means that an audience, or similarly, a user of an information system, would want to know the time of an incident first, showing that temporal information is not only an important aspect to the user, but also often the most meaningful.

Temporal information is essential to most applications for many reasons: to predict the future behavior of a system, to make a plan based on our knowledge about the past, to observe the rules that cause the changes in the system of interest, and to analyze data based on previous observations and explain why things became the way they are now [Shoham and Goyal 19881.

Information systems have to consider temporal data to answer queries that could not be answered otherwise, such as: "Which of our products lasts for a period, twice as long as its warranty time, before it will break?" "When is part 7 is expected to be replaced?" "Which of our employees have more than 5 years experience and earned less than $2,000 a month in 1989T or "Which town had the greatest increase in population last year?" "Which factory is expected to be displaced next year?" "Which city, in 1989, had more than 50,000 inhabitants and still had only one shopping mall?" Or. "Find the hours where the temperature at Orono had its highest reading each day for the last two weeks" Information systems also have to deal in one way or another with outdated data [Allen 19831. The simplest approach would be to delete the old data. However, in this approach, there will be no way to retrieve
earlier-deleted data. Therefore, researchers have developed temporal data models and temporal logic to maintain knowledge over time.

1.2 Fields of Interest for Temporal Reasoning

The concept of time and temporal data arises in various fields of interest, such as philosophy, psychology, linguistics, artificial intelligence (AI), and database management systems (DBMS) [Allen 19831. The queries presented in the previous section show the various areas where temporal information is needed, including industry, marketing, and medicine. For the wide range of areas concerned about time, the framework of a research discipline emerged is often called *temporal reasoning.*

In the last two decades, many research about time has been carried out, especially in AI and DBMS. There are commonalties and differences in the approaches applied in each of the two areas, and we will dedicate most of this report in presenting and explaining them.

1.3 Temporal Reasoning and Temporal Databases

In the following sub-sections, we will present and compare the temporal aspects and the temporal approaches for temporal knowledge representations (KRS) as applied in artificial intelligence and database management systems (DBMS). We will also point out the major differences between the two areas.

1.3.1 Temporal Knowledge Representations (KRS)

In many applications of artificial intelligence, one is concerned with reasoning about time by assertions based on incomplete knowledge stored in a database. Since changes are time dependent, time and change are strongly related. Because of the relation between time and change, one could store the changes in a database as an approach to reason about time.
Therefore, one can distinguish two different approaches to temporal reasoning:

• a change-based approach, and

• a time-based approach.

Models applying the change-based approach such as the *situation calculus* [McCarthy and Hayes 19691 and the *dynamic logic* [Pratt 19761 concentrates on recording changes or facts valid at a certain point in time. The time domain itself is not part of the model. While the changebased approach is easier to visualize and apply, it has many limitations, such as the disability to handle intervals, overlapping actions, and so on.

In a time-based approach, one considers the time as a separate dimension, similar to a one-dimensional space. The different models applying the time-based approach can follow either a

- Point-based approach: The point-based approach considers a dense, complete, unbounded, and real time line (see chapter two for details). Points are used as its basic temporal objects. Each time interval consists of an ordered pair of points [McDermott 19811.

- Interval-based approach: In the interval-based approach, a linear and discrete time is applied. It considers intervals as primitives [Allen 19831.

1.3.2 Temporal Database Management Systems (DBMS)

In DBMS, one considers four points of view:

- Data Models: Existing data models (e.g., relational, EntityRelationship, etc.) have been designed for a world in which time is used for one purpose only-that of ordering transactions. The enrichment of one such model may dictate major amendments.

- Languages: Temporal languages may differ in the representation of time: the semantics (valid vs. transaction time), and the allowed type and expressiveness of the temporal constructs.

- Logical Design of Databases: Classical design methods consider information as composed of static entities and ignore the dynamic behavior of the data while *temporal methods* for logical design provide a unified approach that covers both static and dynamic considerations.

- Physical Issues: Two physical issues were discussed in the literature: the storage of temporal databases and the optimization of temporal queries.

The historic part stored in the database, being large, is preserved in a low-cost ROM device while the current versions reside in more conventional storage media.

The optimization issue is concerned with re-parsing queries into more convenient forms.

1.3.3 The Differences Between HRS and DBMS

Some differences and similarities between a temporal knowledge representation (KRS) and a database management system (DBMS) follow:

- Temporal KRS are designed for *generality* in the representation *of knowledge* while temporal DBMS's are designed for storage *efficiency,* retrieval, and maintenance *of data;* and for supplying the user with the best means to handle well-structured information.

- KRS usually consider a single perspective and uncertainty *of* time. A temporal DBMS may include several perspectives of time and require both a linear model of time and exact dates.

- Information in a KRS is represented by propositions. A DBMS renders information by providing a temporal extension to the data models (e.g. relational, Entity-Relationship) used in a classical DBMS.

- KRS may use first order logic, reified sentences, or modal temporal logic while DBMS applies first order logic only.

1.4 Temporal GIS's

In recent years, temporal reasoning is getting more attention in GIS. Yet a lot of research must still be done before temporal GIS's become available. In the Temporal GIS Workshop at the University of Maine, Orono (Orono, ME, Oct 12-13, 1990), researchers have concluded that the temporal GIS is a wide and a rather complicated issue. There is still a gap between existing technology (temporal databases and temporal query languages) and GIS needs. This gap has to be narrowed down from both ends: the GIS and the database systems. For DBMS's, one has to come out with clearly defined requirements to design the suitable system and data structure that can handle the problem. This is often difficult to do in most GIS applications. Although for some GIS applications, such as cadastral systems and town planning, it is possible to design such requirements, in other applications, such as environmental global-change, it is complicated. In some GIS applications it is not even clear right at the start what results are expected, because of the lack in temporal-data representations and tools for analysis.

Gail Langran has presented a wide range of GIS applications and fields of interest for temporal data, such as forest resource management, urban and regional management, research and development, electronic navigation charts, transportation, and the like, [Langran 89]. Initiative 10 of the National Center for Geographic Information and Analysis (NCGIA) focuses on temporal relations in GIS.

Other researchers are contributing to this topic as well. Armstrong studied time in spatial databases [Armstrong 881. Worboys has shown the role of modal logics in a GIS [Worboys 90a; Worboys 90b]. Hunter and Williamson have presented a method of storing and processing temporal geographic data by the addition of time-encoding attributes to data elements as required. They have also created a historical digital cadastral database to demonstrate this method [Hunter and Williamson 19901. The first work on the field of temporal GIS of note is the work of Bassoglu and Morrison in 1977, who constructed a US Historical County Boundary Data File to hold changes that took place on county boundaries in several states for the last 200 years. Another database that implements time encoding is the Norwegian Socioeconomic Database, which handles administrative boundaries for the period 1770-1980 [Henrichen 19861.

This report is structured as follows: Chapter 2 presents the models and issues on temporal reasoning from the artificial intelligence point of view. It discusses the change-based vs. the time based approach, and describes in detail the interval-based, point-based, and mixed approach. Chapter 3 first discusses the temporal DBMS which include time models and perspectives, temporal attributes and objects, and the data models. Second, it will describe the temporal query languages which include calculus-based and algebra-based approaches. Finally, the chapter will also present the time-based design of the databases and their physical concepts. In chapter 4 we will present some concluding remarks and introduce our recommendations.

Chapter 2: Various Models for Temporal Reasoning

Temporal reasoning is a research area that provides a uniform framework to disjoint fields of research, and is an essential part of most tasks considered as intelligent behavior. It is so prevalent that it is often not recognized explicitly [Allen and Kautz 1985; Shoham and Goyal 19881. The necessity for studying temporal reasoning and representing temporal knowledge arises in a wide range of disciplines. including computer science, geographic information systems, philosophy, psychology, and linguistics. In computer science, it is a crucial topic in information systems, program verification, artificial intelligence, and other areas involving process modeling [Allen 19831.

The importance of temporal reasoning arises when we deal with outdated data since information systems must address this problem. Deleting that data ends the possibility of accessing any old information. The deletion of historic data is absurd in many applications, such as maintaining medical records, real estate, and the like. Examples of aspects for the importance of time in various applications are as follows:

- Medical diagnosis systems ask for the time at which a virus infected the blood system.
- Device troubleshooting systems look at how long it takes a capacitor to saturate.
- Determining senior title rights in real estate. Time is an aspect which will determine senior title rights that can resolve the ownership of disputed areas [Onsrud 19851.

Shoham and Goyal in have pointed out four main tasks appearing in artificial intelligence that need reasoning about time [Shoham and Goyal 19881:

- **Prediction**: to forecast the state of the world at some future time.
- **Explanation**: to produce a description of the world at some earlier time.
- **Planning**: to produce a sequence of actions that would lead to a wanted future world.
- **Learning new rules**: to produce a set of rules that govern the change which account for the observed regularities in the world.

The literature is full of contradictory theories and disputes on the nature and representation of time [Allen and Hayes 1985]. Many researchers on this topic have found it difficult to become aware of and familiar with the works of others. Bolour et al., who presented a survey of 69 papers on time and summarized most of them as a first step towards letting researchers know each other's work, with an attempt leading to an increased communication of ideas on information processing and time [Bolour et al. 19821.

2.1 Difficulties in Temporal Reasoning

While time seems obvious in people's everyday coping with the world [Allen and Hayes 19851, the implementation of temporal data into information systems becomes complex. The difficulty stems from several factors:

• **The Imprecision of some expressions** in the English language to represent actions involving time,

•• *Now*: can mean this moment, today, this year, or this decade, etc.
•• *Today:* can mean this day, at present.

• **Variety of interpretations**: a *year is* perceived as a small amount of time (or a point) by a historian while a second seems like a century to a computer engineer.

• **Variety of actions**: actions are different in their

•• *duration:* There can be *instant actions*, i.e., indivisible actions that have instantaneous effect, and there can also be *actions that hold over a period of time*.

•• *sequence*: actions can obey several assertions according to their relative occurrence in time. For example, two actions may meet, overlap, or disjoint, etc.

•• *effect*: actions can

- have an instantaneous effect: e.g., turn on the light;
- have a delayed effect: e.g., the car alarm goes off 10 seconds after one opens its door;
- undergo a natural death: e.g., the car alarm will stop after 60 seconds;
- cause a periodic effect: e.g., circle around the table; or
- have no effect: e.g., I was standing for 15 minutes.

2.2 Issues in Temporal Knowledge Representation

A model for temporal knowledge representation is based on a certain temporal logic. To construct a temporal logic, one should consider the following issues and define an approach to implement [Shoham and Goyal 19881 [Allen 19831. Some of these issues include the following:

• Conceptual issues:

•• The use of time *points vs. time intervals:* In English, we can refer to time as points or intervals:

We found the letter at twelve noon (a point in time).
We found the letter yesterday (a time interval).

•• *Assertions over points or intervals:* do we allow interpretations over time points or over time intervals? Are intervals of duration zero allowed?

•• The possibility of *inheritance:* Whether the truth of a proposition over one interval constrains the truth over other intervals, e.g., does a property holds over subintervals of a non-point interval when it holds over the non-point interval itself? For example, consider the following two sentences:

The color of my house was yellow during 1988.

I stayed in the USA during 1988 for my studies.

While in the first sentence the color of the house is valid over any subinterval of 1988, in the second sentence, the property *staying* may not hold over subintervals of 1988 since I made several trips outside the USA.

• Structural issues:

These issues are concerned with the mathematical objects used for representing time intervals:

•• *Precedence:* Time can be considered linear, circular, and can be totally or partially ordered.

•• *Discrete vs. dense:* Time is discrete if the number of points in any interval is finite and it is dense if between every pair of points there is another point.

•• *Complete vs. incomplete: A* structure is complete if for every series of points that is bounded from above by another point, there exists a point that is the least upper bound of the series. Otherwise, the structure is incomplete.

•• *Bounded vs. unbounded:* a set of time points [S) is unbounded in the future if every point has a later point, and unbounded in the past if every point has an earlier point. The set is bounded in the future or in the past if a point has no later point or earlier point respectively.

• Logical issues

There are three primary options for the type of logic used in temporal reasoning:

- *Classical First-Order Logic:* In this option, time can be simply included as an argument of a predicate. For example, if M is an interpretation then
  `M V Color (House 17, Red,` T_1, T_2`)`
  Which means that over the interval < T_1, T_2>, *House 17* had color red in that interpretation (M).

- *Reified Sentences:* In this representation, one separates the temporal component from the atemporal component of an assertion. For example, if M is an interpretation, the expression

  `M V True (`T_1, T_2`, Color(House 17, Red))`

  associates the proposition type Color(House 17, Red) with an interval from time T_1 to T_2.

- *Modal Temporal Logic:* The temporality of a formula enters in its semantics instead of its syntax. In Philosophy, modal temporal logics in which time points correspond to possible worlds, the so-called tense logics, have been prevalent. For example, in the red house example we would have

  `M, T V Color(House 17, Red)`

2.3 Change-based Models

Since time and change are strongly related, obviously one could express time only by expressing change. However, this approach has conceptual limitations which can be overcome only by introducing a temporal dimension to the model. Therefore, models for temporal representation can be subdivided according to their approach into either a changebased or a time-based approach [Shoham and Goyal 19881.

In the next sections, the change-based approach is briefly mentioned only for historical reasons; later, we list its limitations. In section 2.4 we will discuss the time-based approach in more detail.

The change-based approach concentrates on recording changes or facts valid at a certain time point. The time domain itself is not part of the models. Prototypes of this approach are *situation calculus* and *dynamic* logic.

• Situation calculus

Situation calculus views the world as a set of states or situations, each of which is a snapshot of the world. Until an action happens, the last state of the world remains unchanged [McCarthy and Hayes 19811.

• Dynamic Logic

Dynamic logic, introduced by Pratt in 1969, is a framework for reasoning about programs based on modal logic. The idea is to integrate programs into an assertion language by allowing them to be modal operators [Pratt 19761. For more details about dynamic logic and its use for temporal GIS the reader is referred to [Worboys 1990a; Worboys 1990b].

The basic change-indicators in the change-based models are *actions* in situation calculus and *programs* in dynamic logic.

• Limitations of the Change-based Approach

The change-based approach has several limitations [Shoham and Goyal 19881. These can be either conceptual or general in their nature:

Conceptual limitations:

- Instantaneous actions: The actions cannot have any duration (e.g., "the system needs 10 seconds to boot up" is not a valid action).
- Instantaneous and immediate effects: The delayed effects (e.g., 10 seconds after the door is open, the car alarm will go off) and natural death (the alarm will stop after 30 seconds) cannot be expressed.
- Concurrent or overlapping actions: Simultaneous actions cannot be expressed in the change-based formalism.
- Continuous processes: The continuous processes are not expressible in change-based systems (e.g., turning on the tap resulted in the level of water growing steadily, until the cup overflowed).

General limitations:

- The qualification problem: In the context of predicting the future, this the problem is in making sound predictions about the future without taking into account everything in the past. For example, when an action happens, its effect is predictable only if nothing goes wrong (e.g., starting a car will result that its motor turns over if the batteries are good, the tank is full, and so on).

- The ramification problem: This problem arises when the results of an actions are very complex. For example, if one drives a car then, as a result, the car and its tires, engine, doors, etc. are, with the car itself, in the new place.

- The frame problem: This problem arises because of the complexity of representing things that remain unchanged because of an action. For example, by moving the car to a new location, its color, size, and windows do not necessarily change.

2.4 Time-Based Approach

The time-based approach recognizes only one real change: the passage of time, which is a constant change unaffected by anything else [Shoham and Goyal 1988). At various points on a time structure, there are true or false assertions. To implement a time-based approach, a temporal logic must be built. In addition, certain decisions have to be made. Assertions can be interpreted over time points, time intervals, or both.

We will discuss three different models of the time-based approach: An interval-based, a point-based, and a mixed model. The interval-based model allows statements to be interpreted only over time intervals [Allen 19841 [Allen and Hayes 19851. In the point-based model there are two kinds of statements: (1) those interpreted over time points are facts and (2) others referring to intervals are events [Dean and McDermott 1987; Harniak and McDermott 1985]. The mixed model interprets assertions over intervals and allow interpretations over of zero-duration intervals as a substitute for time points [Shoham and Goyal 19881.

In the following sections, we summarize briefly the three different models of temporal reasoning:

- the interval-based approach [Allen 19831 [Allen 1984],

- the point-based approach [McDermott 19811 [Harniak and McDermott 19851 [Dean and McDermott 1987],

- and the time-based approach [Shoham and Goyal 19881.

2.4.1 Interval-based Model

The interval-based temporal representation discussed in this section is mainly based on Allen's formalism [Allen 19831 [Allen 19841 [Allen and Kautz 19851. From the perspective of artificial intelligence, Allen introduced a powerful temporal representation that takes the notion of a temporal interval as primitive. The representation describes a method of representing the relationships between temporal intervals in a hierarchical manner using constraint propagation techniques. The representation is designed explicitly to deal with the problem that much of our temporal knowledge is relative, and cannot be described by a date.

The interval-based model for temporal reasoning is based on a set of relationships which hold between time intervals. There are a total of thirteen ways in which an ordered pair of intervals can be related (see Figure 1 bellow).

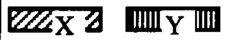
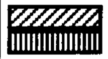
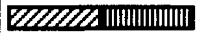
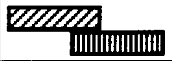
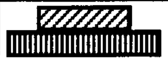
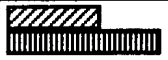
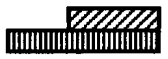
Relation	Symbol	Inverse symbol	Pictorial image
X **before** Y	<	>	
X **equal** Y	=		
X **meets** Y	m	mi	
X **overlaps** Y	o	oi	
X **during** Y	d	di	
X **starts** Y	s	si	
X **finishes** Y	f	fi	

Figure 1: The thirteen (seven plus their inverse) possible relationships between two 1-dimensional intervals [Allen and Kautz 1985].

With these relationships one can express any holding relationship between two intervals. The following are examples of predicates that clarify the interpretation of the figure above [Allen 19841:

before (t_1, t_2): interval t_1 is before interval t_2, and they do not overlap in anyway.
equal (t_1, t_2): t_1 and t_2 are the same interval.
meets (t_1, t_2): interval t_1 is before t_2 with no interval between them. i.e., t_1 ends where t_2 starts.
overlaps (t_1, t_2): interval t_1 starts before t_2, and they overlap.
during (t_1, t_2): interval t_1 starts after and finishes before t_2 .
starts (t_1, t_2): interval t_1 shares the same beginning as t_2, but ends before t_2 ends.
finishes (t_1, t_2): interval t_1 shares the same end as t_2, but begins after t_2 begins.

The relationships between intervals will represent the arcs between the nodes in a network with individual intervals as its nodes. For arcs with uncertainty, all-possible relationships are used. The relation and its network representation are best demonstrated in figure 2.

Relation	Network
1. i during j	N_i --(d) $\rightarrow N_j$
2. i during j or i before j or j during i	N_i --(< d di) $\rightarrow N_j$
3. i before j or j before i or i meets j or j meets i	N_i --(< > m mi) $\rightarrow N_j$

Figure 2. Representing knowledge of temporal relations in a network [Allen 1983].

When a new relation is entered in a network, all consequent relationships among the influenced nodes are computed. This is done by computing the transitive closure of the temporal relations [Allen 19831.
For instance

if i is during j and j is before k then i must be before k

The new facts are added to the network, and so on.

For any three intervals A. B, and C, with the relations rl and r2 between AB and BC respectively, there is a transitivity table which presents the possible relationships r3 between A and C inferred as a function of the two relations rl and r2. The transitive table is shown in figure 3.

In the following example, we show how a newly added temporal relationship to the network will improve our temporal knowledge:

Example [Allen 1983]: Consider the three relations S, R, and L:

S overlaps or meets L:	S(o m)L
S is before, meets, is met by, or after R:	S(< m mi >)R

S(o m)L $\rightarrow$ L(oi mi)S

Br2C / Ar1B	<	>	d	di	o	oi	m	mi	s	si	f	fi
<	<	no info	< o d m s	<	<	< o d m s	<	< o d m	<	<	< o m d s	<
>	no in fo	>	> oi d mi	>	> oi d mi	>	> oi d mi	>	> oi mi d f	>	>	>
d	<	>	d	no info	< o d m	> oi d mi	< m	> mi	d	> oi mi d f	d	< o m d s
di	< o di m fi	> oi di mi	o oi d di	di	o di	oi di	o oi m	oi di mi	di fi o	di	di si oi	di
o	<	> oi di mi	o d	< o di m	< o m	o oi d di	<	oi di	o	di fi o	d s o	< o m
oi	< o di m fi	>	oi d	> oi di mi	o oi d di =	> oi mi	o di	>	oi d f	oi > m	oi	oi di si
m	<	> oi di mi	o d m	< m	<	o d	<	d di =	m	m	d s o	<
mi	< o di m fi	>	oi d mi	> mi	oi d	>	d di =	>	d f oi	>	mi	mi
s	<	>	d	< o m di fi	< o m	oi d f	<	mi	s	s si =	d	< m o
si	< o m di fi	>	oi d f	di	o di fi	oi	o di fi	mi	s si =	si	oi	di
f	<	>	d	> oi fi di si	o d s	> oi mi	m	>	d	> oi mi	f	f fi =
fi	<	> oi mi di si	o d s	di	o	oi di si	m	si oi di	o	di	f fi =	fi

Figure 3: The Transitivity Table [Allen 1983]. It generates the relation r3 between intervals A and C as a function of the given relations r1 between intervals A and B, and r2 between intervals B and C.

To get the relation between L and R, one tests all-possible combinations generated from the transitivity table in figure 3 (given: L(oi mi)S and S(< m mi >)R produces the following possibilities):

L(oi)S	and S(<)R	→	L(< o m di fi)R
L(oi)S	and S(m)R	→	L(o di fi)R
L(oi)S	and S(mi)R	→	L(>)R
L(oi)S	and S(>)R	→	L(>)R
L(mi)S	and S(<)R	→	L(< o m di fi)R
L(mi)S	and S(m)R	→	L(s si =)R
L(mi)S	and S(mi)R	→	L(>)R
L(mi)S	and S(>)R	→	L(>)R

From the above relations we infer: L(< > o oi m di s si fi =)R.
If we add a **new** fact that : L(o s d)R and intersect the two relationships above we conclude that L(o s)R.

Knowing this new fact, we can carry on and improve our knowledge about relationships in the network such that:
given : S(o m)L and L(o s)R produces the following possibilities:

S(o)L	and L(o)R	→	S(< o m)R
S(o)L	and L(s)R	→	S(o)R
S(m)L	and L(o)R	→	S(<)R
S(m)L	and L(s)R	→	S(m)R

From the above relations we infer: S(< o m)R
intersecting S(< o m)R with S(< m mi >)R we conclude: S(< m)R

if there were other nodes such that D(d)S we can infer as well:
D(<)R
D(< o m d s)R

2.4.2 Point-Based Approach

This approach uses points as its basic temporal objects on which an interval has an ordered pair of points [McDermott 19811. As an example illustrating this approach, we briefly describe a temporal-logic-based approach, referred to as *temporal imagery*. The framework is called *shallow temporal reasoning*, because it is broken into several steps, each of which is assumed to require only a small allowance of computational resources to carry out. Shallow temporal reasoning has four steps:

• generating a set of candidate hypothesis,
• selecting one hypothesis from the candidates,
• using the selected hypotheses as a basis for prediction, and
• responding to unforeseen consequences noticed during prediction.

The approach implements time maps. Time maps and the temporal database are briefly described in the following paragraphs.

Time Maps

A time map is a graph whose nodes refer to points or instances of time corresponding to the beginning and ending of events [Dean and McDermott 19871. The points provide a frame of reference for reasoning about the temporal relationships between events. Its edges correspond to constraints, i.e., point-to-point conditions. Edges are labeled with an upper and a lower bound of time.

For more than one consistent constraint between any two points, then

The *maximum* of the *lower bounds* ≤ the *least* of the *upper bounds*
Or if U_i and L_i represent upper and lower bounds for a constraint i

then

$$L_i \leq U_i \qquad , \forall \quad i=1...n$$

A relation between any two points on the map is represented by a path between them. An example of a time map is shown in figure 4.

The relation between p_1 and p_2 is given by the following paths:

c_1	= [5,9]
c_2 c_3	= [6,8]
c_2 c_4	= [-1,11]

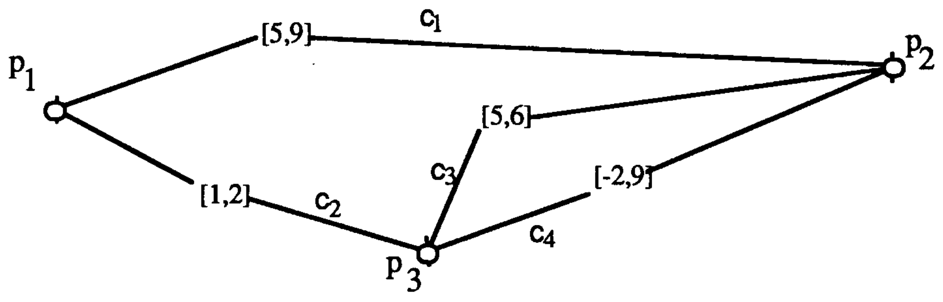


Figure 4. Relating pairs of points with constraints [Dean and McDermott 1987].

The values in brackets (e.g., [5,9]) denote the lower and upper bound of the constraint between points p_1 and p_2 represented in the first

relation. The interested path of the three listed above is the one (depends on the application) with either or both:

•• the greatest lower bound or
•• the least upper bound.

In the above example, the second path c_2- c_3 provides both the highest lower and the lowest higher bound, i.e.. [6,8].

Each interval consists of a *begin* and an *end* point such that the beginning should precede or coincide with the ending. These intervals are called *tokens. A* type is denoted by a formula like (status x is on). Some definitions and terminology used in [Dean and McDermott 19871 follow:

• *Time token* (or token): an interval together with a type.

• *Fact types:* a timelessly true fact.

• *Persistences:* intervals during which a fact type is true (tokens corresponding to facts).

• *Apparent contradiction: two* tokens with contradictory types (e.g., (status x on) and (status x offl) and ordered such that one begins before the other and such that the earlier could persist longer than the beginning of the later.

• *Clipping:* to resolve the apparent contradiction between tokens forcing the end of the earlier to precede the beginning of the later.

The time map management system (TAM) is not simply a simulator, but a means of viewing time "from the side." One reasons with a time map by making modifications to the database and then examining the repercussions of those modifications.

Time maps do not need to contain complete information. In order to be useful, it is only important to adopt a consistent strategy for interpreting the information they contain. The interpretation strategy built into the operation of the TMM is quite simple:

- Events occur as early as they can; the lower bound on the duration of an event token is most indicative of its start time.

- The facts persist as long as possible; the duration of a persistence is best estimated by the upper bound on the distance separating its begin and end points.

- A fact P is true throughout an interval just in case there exists a fact token of type P that
  •• begins before or is coincident with the interval, and
  •• persists at least as long as the interval.

Temporal Database

The temporal database can be viewed as a special case of a database management system which includes the following:

- *A database* (called a time map) capable to:
  •• capture events and their effects over time,
  •• record the truth of changing propositions,
  •• handle the addition of new information,
  •• and the removal of old information.

- *A query language* that:
  •• enables application programs to construct and explore hypothetical situations,
  •• support simple retrieval, e.g., is it possible that P is true at time T given what is currently known?, and
  •• handle retrieval of the form: Find an interval satisfying initial constraints such as (P1 $\wedge$ ... Pn) is true during an interval.
- A *method of extending the information in the data base*, such that:
  •• The user should be able to engage in some sort of forward prediction.
  •• The prediction added to the data base should depend on the antecedent conditions in a meaningful way.

- *A mechanism for monitoring the continued validity of conditional predictions.*

The routines for retrieving data, maintaining internal consistency, and handling forward inference are combined in what is called a *time map management system* (TMM).

2.4.3 Mixed (Time-based) Approach

The time-based approach described in this section is based on the model presented in [Shoham and Goyal 19881. This approach is considered to be a crystallization and a generalization of the ideas proposed by Allen [19841 and McDermott [19811.

The temporal logic constructed in this formalism contains the following features:

• The *assertions* are interpreted over *time intervals.* There are no assertions over time points. Instead, interpretations over intervals of zero duration are allowed.

• The basic *temporal objects* are *points*, following the McDermott formalism for an interval as an ordered pair of points. An interval is presented as a *pair* of its *end-points.*

• No *apriori associations* are made between the truth of an assertion over an interval and its truth over other intervals.

• The *time structure:* no commitment is made, though most of the time the time is viewed to be *linear* and *dense.*

• The *logical form:* the *reified first-order logic is* used. In the reified sentences, one separates the atemporal component of assertions from their temporal component, also called a *proposition type.* This is done by using a predicate, which will take three arguments, two time points denoting an interval, and a proposition type. For example, if M is an interpretation, the expression

$$M \models True\ (T_1, T_2, Color(House17, Red))$$

associates the proposition type **Color(House17,Red)** with an interval from T_1 *to* T_2.

Propositions

Shoham has categorized the propositions into the following six more rich and flexible categories. The different kinds of propositions will be distinguished by how the truth of the proposition over one interval relates to its truth over other intervals [Shoham and Goyal 19881:

- A proposition type X *is downward-hereditary* (written ↓ X) if whenever it holds over an interval it holds over all of its subintervals. Example: "The robot traveled less than two miles."
- A proposition type *x is upward-hereditary* (written ↑ x) if whenever it holds over all proper subintervals of some nonpoint interval (except possibly at its end-points), it also holds over the nonpoint interval itself. Example: "The robot traveled at a speed of two miles per hour."
- A proposition type *x is liquid* (written ↕x) if it is both upward -hereditary and downward -hereditary. Example: "The robot's arm was in the GRASPING state."
- A proposition type is *clay-like if* whenever it holds over two consecutive intervals it also holds over their union. Example: "The robot traveled an even number *of* miles."
- A proposition type is *gestalt if* it never holds over two intervals *of* one which properly contains the other. Example: "Exactly six minutes passed."
- A proposition type is solid if it never holds over two properly overlapping intervals. Example: "executed the NAVIGATE procedure (from start to finish)."

2.4 Discussion

In the previous sections, we have described three models of the timebased approach: the interval-based, the point-based, and the mixed model. Commonalties and differences between these models can be summarized this way:

• Differences in the *conceptual* issues:

•• The interval-based approach considers time intervals as primitives and does not allow time points (however it allows various granularities of time primitives), while the point-based and the mixed approaches allow both time points and intervals.

•• The interval-based model allows *assertions* over intervals only, while the point-based model allows both assertions over time points and intervals. The mixed model allows assertions over time intervals; however, it allows intervals with zero duration as a substitute for time points.

•• The interval-based model applies relations between intervals as a basis to the assertions: the point-based model applies timemaps: and the mixed approach uses inheritance characteristics to categorize the propositions.

•• All three models do not require the exact time, since one can find an assertion for temporal reasoning without it, such as by using previous knowledge about inter-relations among intervals.

• Differences in the structural issues

•• The interval-based and the mixed models consider linear and totally ordered time, while the point-based approach considers branching time.

•• The interval-based model considers discrete time to avoid technical difficulties while the other two models consider a dense, complete, unbounded, and real time line.

• The difference in the logical issues

The interval-based model applies a temporal logic based on temporal intervals and first-order predicate calculus. The point based model implements a temporal logic based on modal logic with different instants playing the role of different possible worlds. The mixed model uses a logic based on the reified sentences.

Chapter 3: Temporal DBMS and Temporal Query Languages

This chapter surveys the temporal DBMS literature. Several surveys on temporal databases, such as the references Bolour et al. [19821; McKenzie and Snodgrass 119861; and Stam and Snodgrass [19881 cover a period from the early 1970's to 1988.

The analysis of the references cited in this chapter will consider the following four points of view:

i) Data Models. Existing models (e.g., relational, EntityRelationship, etc.) have been designed for a world in which time is used for one purpose only-that of ordering transactions. The enrichment of such a model with temporal capabilities may violate some model's premises; therefore, an existing data model will need major amendments to be able to handle time.

ii) Languages. Temporal languages may differ in

• The representation of time (e.g. point-based vs. intervalbased).

• The semantics of time (e.g. linear vs. branching time, valid vs. transaction time).

• The type and expressiveness of the temporal constructs allowed.

iii) Logical design of databases. Classical and logical design methods consolidate, in a consistent fashion, the individual views of the database users into a single schema. Classical methods, however, consider information to be composed of static entities and ignore the dynamic behavior of the data. Temporal *methods* for logical design provide a unified method that covers both static and dynamic considerations.

iv) Physical issues. Two physical issues were discussed in the literature:

• The storage of temporal databases.

• The optimization of temporal queries.

The storage issue arises because the temporal databases record both the current version of the data and a log of the past versions. Typically, the historic part is the largest; therefore, the log is preserved in a low-cost ROM device while the current versions reside in more conventional storage devices.

The optimization issue arises from the convenience of using the properties of temporal intervals to re-parse queries into more convenient forms.

This chapter has four sections. The first one reviews the different data models; afterwards, the different capabilities of query languages will be described. The last two sections will present the design techniques based upon temporal concepts and describe the different physical implementation strategies.

Throughout this chapter we will refer to databases that do not handle time as *static databases.*

3.1 Data Models for Temporal Databases

The temporal models of data of all the DBMS reviewed here are based on an existing *static model* for representing non-temporal data; we will refer to this static model as the *underlying model.*

We will discuss the reviewed temporal models of data under two aspects:

i) the model of time used, and
ii) the changes applied to the underlying model to include time.

These two aspects are not completely orthogonal, since the use of a particular model of time may determine some properties of the temporal model of the data. For example, if one uses the semantics of *transaction time,* our model should be incapable of updating or deleting data items. All previous states of the database should be preserved.

This section is made of three subsections: the first one reviews the different models of time encountered, the second one describes some issues common to all temporal models of data, and the last one examines the references for the underlying model of data.

3.1.1 The Time Models.

The models of time used in temporal databases differ in two aspects:

• the perspective of time

• the mathematical representation of time.

Different perspectives of time arise, for example, when time is considered as:

• either an immutable property of an object, like the age of a rock that can be determined by an isotope content and never be modified

• or a mere attribute of the object, similar to a date assigned by an antique dealer to a piece of furniture, etc.

On the other hand, the mathematical representation of time disregards those semantical considerations and models time as a mathematical structure. Time can be mathematically represented as:

• continuous

• discrete, etc.

Even if the perspectives of time and its mathematical representation may seem independent, there are cases in which they affect each other: for example, branching time (in which several possible futures are planned) prescribes the use of a partially ordered time.

This subsection reviews the different perspectives and the different mathematical models of time.

3.1.1.1 Perspective of Time

Different meanings can be given to time, depending on the perspective used to analyze it. Two perspectives of time were found in the literature:

• The reality vs. representation perspective.

• The fact vs. possibility perspective.

The Reality vs. Representation Perspective

In the "reality" of a DBMS, time is just a mechanism for ordering transaction and their components. The various types of time, called either *physical time* or *transaction time* [Snodgrass and Ahn 1985] are useful for

a) Determining the relative occurrences of events.

b) Reconstructing the state of a database at a given moment.

Transaction time works in a manner similar to that of the *time stamps* used in the concurrency control of DBMS [Bernstein, Hadzilacos et al. 19871. Each object is associated with a transaction time assigned by the machine, which cannot be changed by the user. Whenever an object is modified, the system generates a reincarnation of the object with a new transaction time; the previous version of the object, if it exists, is kept. Since objects are never really destroyed, the user may reconstruct past versions of the database.

The database can also store "representations" of objects that exist in a temporal world. The time associated to these objects is called logical time (in contradiction to physical or transaction time). Snodgrass and Ahn subdivides this logical time into two categories: valid time and user time. Valid *time will* describe the temporal evolution of an object, and it has the semantics used by Allen [1983] or McDermott [19811; *user time is* just an attribute of type "time" with no additional objects. An object may have more than one valid or user times associated with it.

[Snodgrass 1987] uses those definitions of time to classify databases into the following categories:

• Static database, in which objects are not associated to any model of time.

• Rollback databases, that are associated with transaction time.

• Historical databases, that support valid time for their objects

• Temporal databases,, that support both valid and transaction times.

The various relations are illustrated in figure 4 below and their time representation summarized in table 1.

The Facts vs Representation Perspective

Another perspective of time is obtained by considering the ways in which points in time are arranged, either in a *total* order or a *partial* order.

A total order of time points leads to a *linear* time while partial orders lead to *branching* times. Linear models of time plan one future and recall one past only; branching times may consider several possible futures or pasts.

All temporal DBMS reviewed considered linear time only; branching time was found in one reference only, for designing databases from a given temporal specification [Carmo and Sernadas 19881. Carmo and Sernadas use a particular form of branching time that considers a road-like pattern of *trajectories:* time is linear within a trajectory.

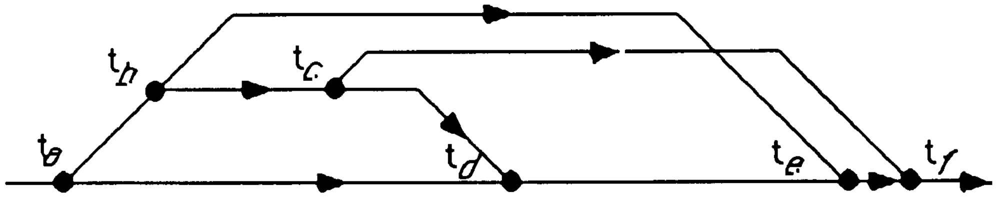


Figure 5: Branching time relations.

Figure 5 above shows the relationships of eight trajectories in branching time, namely (t_a,t_b), (t_b,t_c), (t_c,t_d), (t_d,t_e), (t_a,t_d), (t_d,t_e),(t_e,t_f), (t_b,t_e).

Valid time and transaction time are modelled by two types of different mechanisms: valid time can be represented by temporal logic while transaction time needs dynamic logic [Worboys 1990c].

3.1.1.2 Mathematical Models of Time

The different mathematical models of time can be classified according to two criteria:

1) The Temporal Building Blocks: As seen in chapter 2, there are two basic approaches to define the basic temporal constructs:

i) To model time by an atemporal entity called *instant* that corresponds to a point in time.

ii) To model time by *intervals*, i.e., connected segments that may contain, overlap, abut, etc. other intervals.

The two approaches have been formally proved to be equivalent [Tsang 19871. Example of temporal databases that use intervals as their basic building blocks are [ElMasri and Wuu 19901 [Dutta 19891 [Gadia and Viashav 19851 [Navathe and Ahmed 19881 [Snodgrass 19871 [Segev and Ganadhi 1989]; the databases of [Adiba and Quang 19861 [Ariav 19861 are based upon *instants*.

A representation by instants can be translated to another by intervals by using a continuity assumption that prescribes that everything remains equal unless an instantaneous change occurs. A representation by intervals can be translated to one by instants by considering the instantaneous start and finish of an interval. Intervals seem to be more adequate for modeling *valid* time while an *instant* representation seems more suitable for *transaction* time.

2) The Granularity of Time: As seen in chapter 2, points in time can be *dense* if a new point can be found between any two points in time; otherwise, time is discrete. Similar definitions of granularity exist for the case in which time is modeled by intervals.

Discrete models of time [Adiba and Quang 19861 [Ariav 19861 [Clifford and Tansel 19851 [Lorentzos and Johnson 19881 usually consider a uniform subdivision of time in units called *chronons. A* system that supports several levels of time granularity is proposed in [Clifford and Croker 19881; the chronons at those levels refine each other (i.e., they are years, months, etc.), and operations are defined inside and between the levels.

Dense models of time may be subdivided according to the completeness of their intervals: most systems are not restricted to only use either open or closed intervals.

3.1.2 Common Issues in Temporal-Models of Data

Temporal models of data are based on an underlying static model and provide the underlying model with temporal properties.

Despite the underlying model chosen. three issues are present in temporal databases:

• the integration of temporal attributes into temporal objects

• the extension of the semantics of the underlying static model to the temporal model for the database

• the need for object identity.

In the remainder of this section, we will briefly cover those three issues. We will work with an *interval* model of time and will suppose, for simplicity, that only one type of time will be used.

3.1.2.1 Temporal Attributes and Temporal Objects

For a given instance of an object, a static model associates a value with a particular attribute. In a temporal model, an attribute is associated with a set of pairs *[value* $_i$ *, interval* $_i$*]*. The union of all temporal

intervals associated with an attribute (or object) is called the <u>temporal domain </u>of that attribute or object.

Providing attributes and objects with temporal domains can pose problems when those temporal domains do not coincide. To avoid this problem, the concept of homogeneity was introduced in [Gadia et al., 19851.

Inhomogeneity of Temporal Domains.

An object is *homogeneous* if the temporal domains of all its attributes coincide with the temporal domain of the object. An example of a nonhomogeneous object that models an employee of the diplomatic service is the following:

Name		Av. Salary		Embassy ...	
John	[1980-now)	30K	[1980-1985)	Asuncion	[1980-1984)
		50K	[1985-now)	Ulan Bator	[1986-now)

In the previous object, attributes "Name" and "Salary" have {[1980now)} as their temporal domain while that of "Embassy" has a temporal domain {[1980-1984), [1986-now)}.

Working with heterogeneous objects makes it necessary to use several types of Null values. For example, 1985 does not appear in the temporal domain John's embassy attribute. The absence of 1985 can be represented by either of two types of Nulls:

Null $_1$: The name of John's embassy in 1985 is unknown.
Null $_2$: John was not assigned to an embassy in 1985.

Clifford and Tansel [19851 propose an additional type of Null, Null$_3$, for attributes that have always been unknown.

3. 1.2.2 Extending the Semantics of a Static Data Model

The temporal models of the examined data were all based on an underlying (static) data model. *Snapshots* relate a temporal data model to its underlying model of data: a snapshot of a temporal object describes it at a particular instance of time.

Snapshot semantics are used to extend operations from a static model of data to the temporal model of data. According to snapshot semantics

• Two temporal objects are identical if and only if all their snapshots are equal and

• The output of a temporal operator is defined as a collection of snapshots.

3.1.2.3 The Need for Object Identity

Many problems caused by associating temporal domains to objects can be avoided if the objects are provided with an identity. To illustrate similar problems found, let us consider two versions of an object

Version 1

Name		Av. Salary		Embassy ...	
Peter	[1980-now)	30K	[1980-1984)	La Paz	[1980-now)
		40K	[1984-now)		

Version 2

Name		Av. Salary		Embassy ...	
Peter	[1980-now)	30K	[1980-1984)	La Paz	[1980-1984)
Peter	[1984-now)	40K	[1984-now)	La Paz	[1984-now)

Let us now consider a query that requests the number of people assigned to La Paz during [1981-19851. According to the snapshot semantics, version 1 and Version 2, are identical; however, if the objects in version 2 do not have an identity, the different versions may lead to different answers. This disagreement may occur when the system assumes there are two distinct "Peter's" in Version 2.

Many problems of providing relational databases with a temporal dimension stem from the absence of an object identity in relational systems.

3.1.3 Data Models

The temporal-data models found in the literature were based on either the relational data or the Entity-Relationship (ER) model. This subsection will cover those two models.

3.1.3.1 Temporal Relational Model

In the temporal relational data-model, attributes are associated with a temporal domain. A tuple is said to be *homogeneous* (Sec. 3.1.2. 1) if all its attributes have the same temporal domain.

For example, in the following relation, Dave's tuple is homogeneous while Matt's tuple is not.

Name		Av. Salary		Embassy ...	
Dave	[1980-now)	60K	[1980-now)	Rome	[1980-now)
Matt	[1980-1987)	65K	[1982-now)	Rome	[1980-now)

Two decisions dictate the type of relational model used: the use of NULL values and the use of relations in the First Normal Form RNF).

Usinig NULLS: If snapshot semantics are used for the interpretation of the temporal data model, either the tuples are homogeneous, or one should use NULL values [Clifford and Tansel 19851; the utilization of NULL values should be considered as a method of homogenizing relations.

Using 1NF: Let us consider a homogeneous temporal relation R1 that is not in a First Normal Form (N1NF).

R_1

Name		Av. Salary		Embassy ...	
Peter	[1980-now)	30K	[1980-1984)	La Paz	[1980-now)
		40K	[1984-now)		

The relation R1 can be decomposed into two 1NF relations, R2 and R3:

R_2

Name		Embassy ...	
Peter	[1980-1984)	La Paz	[1980-1984)
Peter	[1984-now)	La Paz	[1984-now)

R_3

Name		Av. Salary	
Peter	[1980-1984)	30K	[1980-1984)
Peter	[1984-now)	40K	[1984-now)

The vertical partition of a temporal relation such as R1 into several 1NF temporal relations is called a *horizontal* anomaly in [Gadia and Viashav 19851. The same reference defines the occurrence of repeated tuples with different time domains (as in R2) or a *vertical* anomaly.

Both INF and NINF models are found in the literature. 1NF temporal models either resort to horizontal anomalies [Navathe and Ahmed 19881, or to a mixture of horizontal and vertical anomalies [Ariav 19861 [Clifford and Tansel 19851 [Lorentzos and Johnson 19881 [Snodgrass 19871. Extending a 1NF static model to a NlNF temporal model involves

i) Representing a temporal attribute as a set of pairs *[value i, temporal-domain i }.*

ii) Imposing a disjointness condition between temporal domains that guarantees that if two values of an attribute are different, their associated temporal domains do not intersect.

Examples of NINF models can be found in Segev and Ganadhi [19891; Gadia and Viashav [19851; Ahn [1986]; and Tansel [1988]. Gadia and Yeung [19881 describe a NINF model in which tuples are provided with *keys.* This work uses a snapshot interpretation to extend nontemporal operations to the temporal domain: such an interpretation is unambiguous as long the results contain a key. Results without a key (e.g., those of a projection) can be given a unique interpretation by assigning alternative keys to them. This assignment, however, may not be unique.

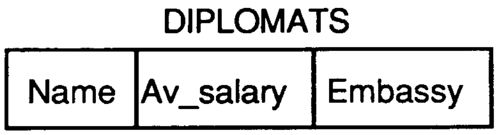
3.1.3.2 Entity Relationship Model

ElMasri and Wuu [1990] present a model for a *historical* ER database. In that reference, entities are made homogeneous by associating a system-defined attribute which is invisible to the user and which acts as a surrogate for that instance with each entity's instance. The temporal domain of this surrogate attribute defines the temporal domain of the entity. Other attributes may have their own temporal domain, and they are assigned a NULL value during those times when the temporal domain of the attribute and the entity do not coincide. A relationship can be defined as non-NULL only on the intersection of the temporal domain of its composing entities.

Adiba and Quang [1986] presents a model for a *Transaction* ER database. It uses a discrete time model (i.e., chronons), and considers both static and temporal data. To preserve consistency, there can be relationships exclusively between entities of the same kind, either static or temporal. Temporal data can be modified only at the beginning of each chronon. Unless prescribed, this model preserves in a log-file all past versions of the temporal data. Temporal entities and relationships can be either *successive* or *manual.* Successive objects are dumped into a log-file after each chronon: manual objects are logged only whenever a change occurs. The model has an option that limits the number of past versions of an object kept inside the log-file.

3.2 Temporal Query Languages

The temporal query languages found are patterned after existing languages of static relational systems. They are based in either relational calculus or relational algebra. A good comparison of eleven temporal query languages can be found in Snodgrass [19871. This section is composed of two parts: one dedicated to calculus languages and the other to algebraic languages. Both subsections will use a homogeneous temporal extension to relation DIPLOMATS that has the following schema:


DIPLOMATS

Name	Av_salary	Embassy

3.2.1 Calculus-Based Temporal Query Languages

A complete temporal query language is TQUEL [Snodgrass 19871. TQUEL is based on QUEL, the language of INGRES [Held et al. 1975].

A query in QUEL is composed of three clauses, called the **range** clause, the **target_list** (or **retrieve** clause), and the **conditional** or **where** clause. The structure of a query on the static version of DIPLOMATS is displayed in the following paragraph, this query obtains the names of the persons in La Paz:

RANGE OF d is DIPLOMATS

RETRIEVE INTO Results(Name = d.name)

WHERE d.Embassy= "La Paz"

TQUEL uses a homogeneous first normal form (1NF) data-model: thus, all attributes of a tuple have the same temporal domain that coincides with the temporal domain of the tuple. To refer to the beginning or the end of the temporal domain of a given tuple variable x, the constructs **BEGIN-OF**. x and **END-OF**. x are used.

TQUEL is designed to support both *valid* and *transaction* time; to do that, it enriches QUEL with three more clauses: the **WHEN,** the **VALID**, and the **AS-OF** clauses.

The **WHEN** clause is used to qualify the *valid* temporal domain of attributes: it is the temporal equivalent of the **WHERE** clause. In the same way that the **WHEN** clause in QUEL can include predicates of the

form $a \leq b$, $c \neq d$, etc., the **WHEN** clause of TQUEL can use Allen's temporal relations [Allen 19831 to qualify tuple variables. For example, the following query gives the names of all persons who worked in Brasilia after John worked in La Paz:

```
QUERY 1:

RANGE OF d1 is DIPLOMATS

RANGE OF d2 is DIPLOMATS

RETRIEVE INTO Results (Name = d1.name)

  WHERE d1.Embassy = “La Paz” and d2.Embassy = “Brasilia”

  WHEN d2 AFTER d1
```

The **VALID** clause is similar to the **target-list** clause of QUEL: it specifies the values of *valid* time to be in the answer. The following query gives the names of all the persons who worked in Brasilia after John stopped working in La Paz. It also includes the time at which they started their work.

```
QUERY 2:

RANGE OF d1 is DIPLOMATS

RANGE OF d2 is DIPLOMATS

RETRIEVE INTO Results(Name = d1.name)

WHERE d1.Embassy= "La Paz" AND d2.Embassy= "Brasilia"

VALID AT BEGIN_OF d2

WHEN d2 AFTER d1
```

Finally, the AS-OF clause handles *transaction* time and dictates the version to be used. QUERY 3 gives the answer that QUERY 2 would have rendered if the query was posed in 1988.

```
QUERY 3:

RANGE OF d1 is DIPLOMATS

RANGE OF d2 is DIPLOMATS

RETRIEVE INTO Results (Name =  d1.name)

WHERE d1.Embassy= “La Paz” AND  d2.Embassy= “Brasilia”

VALID AT BEGIN_OF d2

WHEN d2 AFTER d1

AS_OF "1988"
```

Several enhancements not provided in TQUEL appear in other languages: scan, aggregate functions, windows, and the management of non first normal form (NINF) relations.

Sequential temporal scan: TSQL [Navathe and Ahmed 19881 and HMMD [Adiba and Quang 19861 can access NEXT, PREVIOUS, LAST, FIRST, Nth adjacent intervals.

Aggregate Functions: TSQL [Navathe and Ahmed 19881 and GORDAS [ElMasri and Wuu 19901 provide aggregate functions for objects (MAX, MIN, COUNT, AVERAGE). GORDAS provides, in addition, aggregate functions for temporal domains (TNLAX, TMIN).

Windows: TSQL [Navathe and Ahmed 19881 introduces an SQL dialect with windows, i.e., intervals of time that can float along the whole temporal domain. Windows can be used, among other things, to provide aggregate functions for temporal domains. An example of this is the following query that provides the two-year time-period in which the salary of each diplomat increased the most:

```
QUERY 4

SELECT Name, MAX (LAST salary - FIRST salary), WINDOW

FROM DIPLOMATS

MOVING WINDOW 2 years
```

Events and intervals: Sarda [19901 presents a temporal SQL designed to work both with discrete-time events and with intervals; the underlying data-model considers homogeneous-lNF databases Among others, the query language provides constructs for seperating intervals into events and for coalescing consecutive events into intervals.

Management of NlNF relations: TBE [Tansel et al. 19891 is a temporal extension to QBE, and HTQUEL [Gadia and Viashav 19851 is a temporal extension to QUEL. Both languages allow for the use of N1NF relations. Let us consider the following N1NF relation:

Name		Av_Salary		Embassy ...	
John	[1980-now)	30K	[1980-1985)	Asuncion	[1980-1984)
		50K	[1985-now)	Ulan Bator	[1984-now
Peter	[1980-now)	30K	[1980-1984)	La Paz	[1980-now)
		40K	[1984-now)		
Henry	[1980-now)	60K	[1980-1984)	London	[1980-now)

Queries in HTQUEL may not return whole N1NF tuples. For example, a query asking for the names, salaries and embassies of those persons earning less than 40K is posed as follows:

QUERY 5

RANGE OF d is DIPLOMATS

RETRIEVE INTO Results$_1$(Name = d.name, Av_Salary = d.Av_Salary, Embassy = d.Embassy)

WHERE d.Av_Salary < 40K

This query yields a result that contains fragments of the N1NF tuples for John and Peter:

Results $_1$

Name		Av_Salary		Embassy ...	
John	[1980-now)	30K	[1980-1985)	Asuncion	[1980-1984)
				Ulan Bator	[1984-1985)
Peter	[1980-now)	30K	[1980-1984)	La Paz	[1980-1984)

HTQUEL provides a **CLOSURE** qualifier for providing *whole* NlNF tuples instead of fragments of them. For example, let us consider the following query:

```
QUERY 6

RANGE OF d is DIPLOMATS

RETRIEVE INTO Results2

CLOSURE (Name = d.name, Av._Salary = d.Av_Salary, Embassy =
              d.Embassy)

WHERE d.Av_Salary < 40K
```

Query 6 returns relation Results_2 that contains complete N1NF tuples.

Kaffer et. al. [19901 describe the application of an extensible DBMS to modelling temporal objects. In it, valid time is just one more attribute and the history of an object is modelled as an aggregation relationship; the query language is a dialect of SQL that allows navigation across relationships.

Results_2

Name		Av. Salary		Embassy ...	
John	[1980-now)	30K	[1980-1985)	Asuncion	[1980-1984)
		50K	[1985-now)	Ulan Batot	[1984-now)
Peter	[1980-now)	30K	[1980-1984)	La Paz	[1980-now)
		40K	[1984-now)		

3.2.2 Algebra-Based Temporal Query Languages

We will review two approaches for the algebraic manipulation of NINF relations.

Clifford and Tansel [19851 and Clifford and Croker [19871 present a NlNF model with NULLS by enriching the basic operators of relational algebra in the following way:

i) They give several versions of the Join of two relations Ri and R2:

- Temporal-join: The results of the Join are a pair of tuples of $r_1 \in R_1$ and $r_2 \in R_2$ such that the temporal domains of r_1 and r_2 intersect.

- Attribute_join: This operation is similar to the one in static relational algebra.

ii) They introduce two new unary operators:

- **WHEN**: when applied to a relation, yields the union of the temporal domains of the tuples.

- **TIME_SLICE**: takes a temporal domain t_{dom} as one of its arguments, and projects a temporal relation onto that temporal domain.

Another version of NlNF temporal algebra enriches the standard algebra with five operators [Tansel 1988]:

- **UNPACK**: transforms a N1NF relation into a lNF one.

• **PACK**: translates a lNF relation into a N1NF one.

• **T_DEC**: translates a temporal domain into a pair of attributes.

• **T_FORM**: transforms a pair of attributes into the start and end of a temporal domain.

• **SLICE**: similar to the WHEN operator [Clifford and Tansel 19851.

3.3 Time-Based Design of Databases

Standard methods for the logical design of databases are aimed to obtain a global data schema. The design proceeds in a series of stages [Kahn 19781:

• Obtainment of the real-world requirements.

• Obtainment of the local information structures.

• Integration into a global information structure.

• Specification of entities and relationships.

• Accommodation to a target DBMS.

The requirement phase of those design methods consists of an analysis of transactions, which is aimed at two goals:

• the description the information accessed by a transaction, and

• the identification of the operations performed on that information.

As described by the requirements of transactions, those methods are concerned with modeling the real world to express the model within the DBMS. Since the requirements considered static conditions, in which *before, after,* and *because* of did not exist, the resulting design will faithfully model a static world.

Three features are missing in a static design:

1. <u>Consistency conditions.</u> Consistency is related to change: a database should be *created* in a consistent state; and the state of the database should remain consistent *before* and *after* transitions.

2. <u>Relationships between transactions and consistency.</u> To insure consistency, the specification of a transaction should consider pre- and post-conditions, i.e., conditions that take into account the state of the database before and after the transaction.

3. <u>Specification of consistency-preserving mechanisms.</u> Corrective measures should be taken whenever a transaction violates any consistency condition, such as rejecting an offending operation, aborting the transaction, and spawning a corrective process, etc. The specification of a corrective measure should describe what to do *(action),* under which circumstances *(condition),* and the proper time to do it *(event).*

Temporal methods for database specification lack those deficiencies. Two methodologies were found for the temporal specification of information systems that coincide in their use of

• temporal logic, and

• an approach based upon layers of increasing refinement.

They differ on their interrelations between the state of a database and the events of the database:

A state-based approach defines events as changes in a state of a database [Castilho et al. 19821.

The *INFOLOG* approach considers the state of a database at any given time as the collection of all events that have occurred until then [Canno and Semadas 19881.

We will now describe the different layers of refinement of these two approaches.

3.3.1 The State-Based Approach

The specification method in Castilho et al. [19821 uses the states of a database as well as their transitions. That method considers two types of constraints: *static* constraints that describe invariants of the database, and *transition* constraints that prescribe the relations between the state of a database before and after a transition. Two examples of *static* constraints follow: "the ages of all employees should be positive" and "no salary may decrease." Example of a *transition* constraint is "The social security number of an employee should appear on the payroll file after an employee is hired."

Specification is made at three layers of increasing detail:

Layer 1: No updates- In this level, conditions are expressed in temporal logic: no explicit reference to the change is made. For example, the specification that salaries may never decrease can be posed as "For any employee *e,* if *e* has now salary s, then later, if *e is* still an employee, *e* must have a salary s'. with s' ~! s."

Layer 2: Encapsulation- The conditions of this level involve updates. These conditions may either refine those of Level 1 or be new. In neither case should contradictions arise. The formalism used at this level is that of a multi-sorted temporal language. Such a language has states and programs among its sorts that are defined by pre- and post-conditions. That language involves the use of triggers: a trigger is a way of implementing actions to be taken when a certain condition is fulfilled.

Layer 3: Update definition- This level involves the definition of built-in updates by programs.

3.3.2 The INFOLOG Approach

This approach models a database as a system that can perform actions and communicate with other systems by messages. Events are constructs used for modeling atomic operations, such as sending and receiving a message. The state of a database is the history of all events that have so far occurred.

The INFOLOG specification method has five layers:

Layer 1: Static constraints- Sets forth conditions that do not need temporal constructs, such as *"Salaries are non-negative."* The formalism used here is a first-order logic. This layer and the following one are equivalent to Layer I of the statebased approach [Castilho et al. 19821.

Layer 2: Dynamic constraints- Specifies dynamic constraints without referring to events, e.g., *"The salary of an employee may not decrease with time."* The formalism used here is a multi-sorted temporal logic.

Layer 3: Application-oriented operations- Specifies conditions to be held *before* and *after* events, such as *"an employee must be on the payroll before being fired; afterwards, the employee must disappear from the payroll."* The formalism used here is the same as the one of layer 2.

Layer 4: Liveness- Stipulates cause-effect relationships among events, such as "if *an employee requests a loan, the employee should receive later a written answer."* The formalism used here is one of temporal logic with branching time.

Layer 5: System behavior- Sets forth two set of rules: evolution and synchronization rules. Evolution rules couple events to actions; synchronization rules guarantee a proper and deterministic behavior by dictating the order in which actions should be taken. An example of an evolution rule is seen where an employee has to change office, i) the employee should vacate the office and ii) another office should be available. A synchronization rule would dictate that the new office should be available before the employee starts vacating the old one. The formalism used here is the same as the one of layer 4.

3.4 Physical Concepts in Temporal Databases

The literature on the physical issues in temporal databases is centered on two issues: *temporal data structures* and *the optimization of temporal queries.* The inclusion of time in a database adds certain peculiarities to these two issues: the data used in the rollback databases should be permanent, thus making convenient the use of optical disks, and the semantics used in temporal predicates allow new possibilities to query optimization.

3.4. 1 Data Structures

A method for storing the information of a rollback database presented in [Lum et al. 1984]. It

i) segregates the current and the past versions of the data, and

ii) provides separate access methods for those versions, by two B + trees.

The past versions of an object are chained backward and ordered according to their transaction time.

Another approach recommends the segregation of data in two levels: one corresponding to the *current,* and the other to its *past* versions [Ahn 1986]. Several mechanisms are suggested to improve the delays involved in traversing a chain, such as *accession lists, clustering, stacked versions,* and *cellular chaining.* It has been claimed that the two-level segregation of data is advantageous and that ISAM and hash mechanisms are not suited for temporal data [Ahn 19861.

Rotem and Segev [19871 consider the problem of partitioning the pages of a historical database when overflow occurs. To guide the partition they propose to consider tuples as if they are defined in a two-dimensional space having *(tuple-identifter, time)* as coordinates. They propose and compare several symmetrical and asymmetrical partition algorithms; asymmetrical algorithms perform better in reducing overflow.

Kolovson and Stonebraker [19891 describe indexing methods for historical databases used in Postgres. They present two indexing schemas which are based on R-trees, use magnetic and optical disk, and resort to a vacuum-cleaner process (VCP) that transfer data from magnetic to an optical disk.

The first indexing schema, called OD-RT-1, has a single R-tree that uses both optical and magnetic disk. When the magnetic disk is partially full, the VCP transfers a fraction of the leaf pages, corresponding to the oldest leaves, from magnetic to optical storage. Afterwards, all intermediate leaves in the magnetic disk that point to optical leaves are also "vacuumed."

The second indexing schema, called OD-RT-2, has two R-trees: the first one is completely stored in magnetic media while the second resides entirely, except its root, in an optical disk. When the first tree is full, it is totally vacuumed into the optical storage.

OD-RT-1 indexing occupies less space, but requires more disk accesses than its counterpart. Both indexing methods compare favorably to indices that reside entirely on optical disk.

Elmasri et al. [19901 use a B+ tree for accessing versions in an appendonly database. Each tree leaf contains pointers to the versions active during certain period and adjacent leaves correspond to consecutive time intervals. Since an object's version may span several periods, a method is proposed that combines lists of active versions with lists of changes. Finally, the authors propose the use of a two-level index (first by attribute, then by version) to assist in selections involving a temporal constraint.

Gunadhi and Segev [19901 compare the efficiency of temporal vs. standard DBMS's for processing of temporal queries. They conclude that, because of efficiency, the former are necessary even if the latter ones can be programmed to solve temporal queries; a standard DBMS may need several passes for operations that require a single step in a temporal DBMS.

3.4.2 Query Optimization

The references found use the properties of time intervals and the semantics of time predicates to guide the manner in which access operations are performed.

Segev and Ganadhi [19891 present methods for performing outer-joins in a N1NF temporal relation. Temporal Joins between two relations R_1 and R_2 consider *both* the values and temporal domains of the attributes of R_1 and R_2. Since in a N1NF relation, a temporal domain may be a disconnected set of intervals, the standard methods for performing a Join in the static methods must be modified to account for that. Four new Join algorithms are presented, one of them based upon a new structure called an *append-only* tree. The results point towards the convenience of using either sort methods or the append-only tree for temporal joins.

Leung and Muntz 11990] present methods for performing joins in a 1NF temporal relation. Their methods are different from those of Segev and Ganadhi [19891 and are based upon standard computational geometry algorithms for counting line intersections [Preparata and Shamos 1985]. The semantics of the interval predicates of Allen [1983] can be used in the optimization of temporal queries [Leung and Muntz 1990].

3.5 Discussion

All the temporal data models presented in this chapter were extensions of existing static data models. There are some reasons for extending an existing model instead of designing a new one:

• User friendliness. It is convenient to give the users something close to their past experiences.

• Interoperability. The temporal data models are to be used in conjunction with existing static models. Interoperability is easier if the temporal models are a generalization of existing ones.

• Easiness of implementation. The effort for providing temporal capabilities to an existing DBMS temporal capabilities is usually less than the one required in building an "ad-hoc" system.

Modifying an existing system, however, brings forth complications. In the case of the static -to -temporal DBMS modification, typical difficulties arise when

• The static model has some limitations. For example, in relational systems, the absence of tuple identifications and the incapability of modeling aggregation or generalization cause problems (heterogeneity, nulls, etc.).

• Several perspectives of time are included. This capability may introduce non-uniform methods of handling those temporal perspectives.

To be useful, these modifications bring more convenient features than problems. Because the enriched semantics allow

• A better atmosphere for modelling a temporal environment.

• The capability of working with versions.

• DBMS that require very inexpensive recovery procedures.

• More possibilities for query optimization.

Chapter 4: Conclusions

The literature about time and databases fall into two broad classes:

• Temporal Knowledge Representation Systems (TKRS)

• Temporal Database Management Systems (TDBMS)

We will present our conclusions on the differences and similarities found in the literature, and on the desired characteristics for a TDBMS.

4.1 Comparison of the Approaches

All references deal with the same problem of answering queries in a temporal environment. Depending on which class was included, the references differ in the following aspects:

i) Underlying design criteria

Temporal KRS are designed for *generality* in the representation of *knowledge.* To that end, they provide the user with a simple but powerful set of tools for temporal reasoning. On the other hand. temporal DBMS are designed for *efficiency* in the storage, retrieval, and maintenance of *data;* therefore, they supply the user with high-performance methods for handling well-structured information.

A large temporal KRS may include thousands of objects while a big temporal DBMS may handle millions of them. This alleged weakness is compensated with the ability of expressing more information about those fewer objects.

In brief, a temporal DBMS trades generality for efficiency and reliability. The opposite assertion is true for a KRS.

ii) Perspectives of time

KRS usually consider a single perspective of time, while temporal DBMS may include several perspectives. The time used in KRS may admit uncertainty in its representation or a branching model, while those representations used in temporal DBMS require both a linear model of time and exact dates.

iii) Entities used in their representation

Information in KRS is represented by propositions. DBMS store information in a more structured way, by providing a temporal extension to the data models (e.g. relational, Entity-relationship, etc.) used in classical DBMS.

The entities found in both temporal KRS and temporal DBMS can be associated by temporal relationships of the type presented by [Allen 19831.

There are differences, though, in the other mechanisms for associating entities. In KRS, when a proposition holds on a certain interval, the type of the proposition may dictate the manner in which its validity is inherited to their subintervals. The models used in temporal DBMS do not include such a property. They are supplied instead with mechanisms of aggregation and generalization absent in temporal KRS.

iv) Logiical formalism used

KRS may use first-order logic, reified sentences, or modal temporal logic, while DBMS apply first-order logic only.

In summary, the differences between temporal KRS and DBMS can an be explained in terms of their basic design criteria: DBMS achieve efficiency by providing more structure, but with fewer capabilities.

4.2 Desired Characteristics for a Temporal DBMS

Desirable temporal DBMS features are discussed by enumerating a collection of characteristics that are either missing in some implementations or that have been proposed in the literature as convenient enhancements:

• The ability to model both intervals and points in time. This will enable the usage of both point-based and interval-based models.

• Since objects may change in time, the existence of an *object identity* is necessary to discriminate the occurrence of a single object in different states from that of several distinct objects.

• The uniform treatment of several perspectives of time. Existing query languages use ad-hoc constructs for the different perspective of the time they handle. Some efforts have been made, however, to formulate a single set of constructs that can be used by all perspectives.

• The treatment of generalization. There is a body of literature on how aggregation is affected by time (e.g., homogeneous and nonhomogeneous models). No similar effort was found for generalization.

• Temporal specification methods for database design. These methods seem to be particularly suited to the design of active databases, since they consider triggers, causality, and liveness conditions.

• User interfaces for temporal navigation. That is, interfaces that allow users to move forward or backward along a given perspective of time. In particular, references have been made to the use of spatial metaphors to handle time.

Acknowledgments

This work was partially supported by Digital Equipment Corporation under grant TP-765536 and NSF under grant SES 88-10917. Their support is truly appreciated. The authors thank also Andrew Frank and Max Egenhofer for valuable remarks, Robert Cicogna for his remarks on the English proof reading, and lastly we thank Ms. Sue Comins for the final English editing of this report.

References

1. Adiba, M. and N. Buiquang. "Dynamic Database Snapshots, Albums and Movies." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 203217, 1988

2. Adiba, M. and N. B. Quang. "Historical Multi-Media Databases." in proceedings of 12th Very Large Databases Conference. pp. pp. 63-70. August 1986.

3. Ahn, I. "Towards an implementation of database management systems with temporal support." in proceedings of *2nd* Conference on Data Engineering. pp. pp. 374-381. January 1986.

4. Ahn, I. and R. Snodgrass. "A Performance Evaluation of a Temporal Database Management System." in proceedings of SIGMOD Conference. pp. pp. 96-107. June 1986.

5. Al-Taha, K. and R. Barrera. 'Temporal Data and GIS: An Overview." in proceedings of 1990 GISMIS. pp. California.

6. Allen, J. F. "Maintaining Knowledge about Temporal Intervals." Communication of the ACM. 26 (11): 832-843, 1983.

7. Allen, J. F. "Towards a General Theory of Action and Time." Artificial Intelligence. 23 : 123-154, 1984.

8. Allen, J. F. and P. J. Hayes. "A common-Sense Theory of Time." in proceedings of 9th International Conference on Artificial Intelligence. pp.

9. Allen, J. F. and H. A. Kautz. "A Model of Naive Temporal reasoning." Formal Theories of the Commonsense World. Ablex Publishing Corporation. Norwood, NJ, pp. 1985

10. Ariav, G. "Design Requirements for Temporally Oriented Information Systems." Temporal Aspects in Information Systems. North Holland. pp. pp. 3-16, 1988

11. Ariav, G. A. "A Temporally Oriented Data Model." ACM Transactions on Database Systems. ILI (4): pp. 499-527, 1986.

12. Armstrong. "Temporality in Spatial Databases." in proceedings of GIS/LIS'88. ACSM. pp. 880-889. San Antonio.

13. Barbic, F. and R. Maiocchi. "Planning in Time." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 141-156, 1988

14. Barrera, R. Active databases: its applicabilily to ISDS. department of Surveying Engineering and NCGIA, University of Maine, 107 Boardman Hall, Orono ME 04469, ASDS project report, 1990.

15. Barrera, R. and K. AI-Taha. Models in Temporal Knowledge Representation and Temporal DBMS. department of Surveying Engineering and NCGIA, University of Maine, 107 Boardman Hall, Orono ME 04469, Survey of temporal reasoning and DBMS's in the literature., ASDS project report, 1990.

16. Bernstein, P. A., V. Hadzilacos and N. Goodman. Coneurrency Control and Recovery in Database Systems-.1987.

17. Blanken, H. and A. Ijbema. "Language Concepts for Versioned Hierarchical Objects." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 187-201, 1988

18. Bolour, A., T. L. Anderson, L. J. Dekeyser and H. K. T. Wong. "The Role of Time in Information Processing: a survey." SIGMOD. Record (12)3: 27-50, 1982.

19. Carmo, J. and A. Sernadas. "A Temporal Logic Framework for a Layered Approach to Systems Specification and Verification." Temporal Asl2ects in Information 5ystems. Rolland, Bodart and Leonard ed. Elsevier. pp. pp. 31-46, 1988

20. Castilho, J. M. V. d., M. A. Casanova and A. L. Furtado. "A Temporal Framework for Database Specifications." in proceedings of 8th International Conference on Very Largre Data Bases. pp. 280-291. Mexico City, September.

21. Clifford, J. and A. Croker. "The Historical relation Data Model (HRDM) and Algebra Based on Lifespans." in proceedings of 3rd International Conference on Data Engsineerin . pp. pp. 528-537. January.

22. Clifford, J. and A. Croker. "Objects in Time." Database Engineering. 11 (4): pp. 189-196, 1988.

23. Clifford, J. and A. Rao. "A Simple, General Structure for Temporal Domains." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 1728, 1988

24. Clifford, J. and A. Rao. "A Simple, General Structure for Temporal Domains." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 1728, 1988

25. Clifford, J. and A. U. Tansel. "On an Algebra for Historical Relational Databases: two views." in proceedings of Proceedinas SIGMOD Conference. pp. pp. 247-265. May.

26. Dean, T. L. and D. V. McDermott. "Temporal Data Base Management." Artificial Intelligence. 32 : 1-55, 1987.

27. Dietz, J. L. G. and K. M. V. Hee. "A Framework for the Conceptual Modeling of Discrete Dynamic Systems." Temporal Aspects in Information ~9,ystems. Rolland, Bodart and Leonard ed. North Holland. pp. 61-75, 1988

28. Dutta, S. "Generalized Events in Temporal Databases." in proceedings of 5th International Conference on Data Engineering. pp. 118-125.

29. ElMasri, R. and G. T. Wuu. "A Temporal Model and Query Language for ER databases." in proceedings of 6th International Conference on Data Engineering. pp. 76-85. January.

30. ElMasri, R., G. T. Wuu and Y.-J. Kim. "An Access Structure for Temporal Data." in proceedings of 16th VLDB Conference. pp. Brisbane, Australia 1990.

31. Elmasri, R., G. T. J. Wuu and Y. I. Kim. "'Me Time Index: an access structure for temporal data." in proceedings of 16th VLDB- pp. 1-12. August 1990.

32. Frank, A. "SVE 451, Lecture." Orono, 1988

33. Frank, A. Layered concepts in ISDS.-University of Maine, Department of Surveying Engineering and NCGIA, internal memorandum, 1990.

34. Gadia, S. and J. Viashav. "A Query Language for a Homogeneous Temporal Database." in proceedings of 4th Conference on Principles of Database Systems. pp. 51-56. March.

35. Gadia, S. K. "A Generalized Model for a Temporal relational Database." in proceedings of SIGMOD Conference. pp. 251259. June 1988.

36. Gadia, S. K. "A Homogeneous Relational Model and Query Languages for Temporal Databases." ACM Transactions on Database Systems. 13 (4): 418-448, 1988.

37. Garnick, D. K., A. T. Cohen and H. A. Sowizral. "Timestamping in Virtual Time Systems." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 221-234, 1988

38. Ghargava, G. and S. K. Gadia. "Achieving Zero Information-loss in a Classical Database Environment." in proceedings of 15th Ve1y Large Databases Conference. pp. 217-224. August.

39. Guesgen, H. W. Spatial Reasoning Based on Allen's Temporal Reasoning. International Computer Science Institute, Berkely, CA, Report No. TR-89-049, 1989.

40. Gunadhi, H. and A. Segev. "A Framework for Query Optimization in Temporal Databases." in proceedings of 5th Conference on Statistical and Scientific Database Management. Michalewicz ed. pp. 131-147. LNCS Springer, April 1990.

41. Harniak, E. and D. McDermott. Introduction to Artificial Intelligence. Addeson Weslet, 1985.

42. Held, G., M. Stonebraker and E. Wong. "INGRES-A Relational Database System." in proceedings of 1975 National Computer Conference. pp. 409-416. May 1975.

43. Henrichsen, B. Directions of major archives. IASSIST Quarterly, 3. 3-6, 1986.

44. Hintz, R. J. and H. J. Onsrud. A Methodology for Upgrading Real Propea Boundga Information in a GIS using a Temporally Efficient Automated Survey Measurement Management System. Department of Surveying Engineering, University of Maine, 1990.

45. Hunter, G. J. "Non-current data and geographical information systems. A case for data retention." International Journal for Geographical Information Systems. 2 (3): 281-286, 1988.

46. Hunter, G. J. and I. P. Williamson. "The development of a historical digital cadastral database ." Int. J. Geographical Information Systems. 4 (2): 169-179, 1990.

47. Jeffrey, R. Formal Logic: Its Scol2e and Limits. Gamer ed. McGrawHill, Inc., 1981.

48. Kdffer, W., N. Ritter and H. Sch6ning. "Support for Temporal Data by Complex Objects." in proceedings of 16th VLDB. pp. 24-35. August 1990.

49. Kahn, B. "A Structure Logical Database Design Methodology." in proceedings of NYU Symposium on Database Design. pp. 15-24. May.

50. Kolovson, C. and M. Stonebraker. "Indexing Techniques for Historical Databases." in proceedings of 5th International Conference on Data Engineering. pp. 127-137.

51. Lamport, L. "Sometimes is Sometimes Not Never." in proceedings of ACM Symposium on the Principles of Programming Languages. pp. 174-185.

52. Langran, G. "A review of temporal database research and its use in GIS applications." International journal for Geographical Information Systems. 3 (3): 215-232, 1989.

53. Langran, G. Time in Geographic Information 5ystems, University of Washington, Ph.D, 1989.

54. Langran, G. and N. Chrisman. "A Framework for Temporal geographic Information." Cartographica. 25 (3): 1-14, 1988.

55. LePape, C. and S. F. Smith. "Management of Temporal Constraints for Factory Scheduling." Temporal Aspects in InformationSystems. Rolland, Bodart and Leonard ed. North Holland. pp. 159-170, 1988

56. Lester, M. "Tracking the Temporal Polygon: A Conceptual Model of Multidimensional Time For Geographic Information Systems." in proceedings of The temporal GIS workshop. pp. University of Maine, Orono, ME, October 13, 1990.

57. Leung, T. Y. C. and R. R. Muntz. "Query Processing for Temporal Databases." in proceedings of 6th International Conference on Data Engineering. pp. 200-208. January.

58. Lorentzos, N. A. and R. G. Johnson. "Requirements Specification for a Temporal Extension to the Relational Model." Database Engineering. 11 (4): 204-211, 1988.

59. Lorentzos, N. A. and R. G. Johnson. "IRA: a model for temporal relational algebra." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 95-108, 1988

60. Lum, V. and P. D. e. al. "Designing DBMS Support for the Temporal Dimension." in proceedings of SIGMOD Conference. pp. 115-130. June 1984.

61. McCarthy, J. M. and P. J. Hayes. "Some Philosophical Problems From the Standpoint of Artificial Intelligence." Readings in Artificial Intelligence. : 431-450, 1981.

62. McDermott, D. A Temporal Logic for Reasoning about Pro cesses and Plans. Computer Science Department, Yale University, RR 196, 198 1.

63. McKenzie, E. "Bibliography: Temporal databases." 15 (4): 40-52, 1986.

64. McKenzie, E. and R. Snodgrass. "Extending the Relational Algebra to Support Transaction Time." in proceedings of SIGMOD Conference. pp. 467-478.

65. Moens, M. "Temporal Databasesand Natural Languages." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 171-183, 1988

66. Navathe, S. B. and R. Ahmed. "TSQL: A Language Inference for History Databases." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 109-121, 1988

67. Newell, R. G., D. Theriault and M. Easterfield. "Temporal GIS -Modelling the evolution of spatial data in time." in proceedings of Functionali:Ly Conference. pp. Univ. of Leicester, U.K., March 21-22.

68. Oberweis, A. and G. Lausen. "On the Representation of Temporal Knowledge in Office Systems." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 125-140, 1988

69. Pratt, V. R. "Semantical Considerations on Floyd-Hoare Logic." in proceedings of 17th FOCS. IEEE. pp. 109-121. New York.

70. Preparata, F. and M. Shamos. Computational Geometry: An Introduction. Springer-Verlag, 1985.

7 1. Reiter, R. "Towards a logical reconstruction of relational database theory,." On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and Programming Languages. M. M. L. Brodie Schmidt ed. Springer Verlag,. New York, pp. 191-233, 1984

72. Rotem, D. and A. Segev. "Physical Design of Temporal Databases." in proceedings of 3rd International Conference on Data Engineering. pp. 547-553. January 1987.

73. Saake, G. and U. W. Lipeck. "Foundations of Temporal Integrity Monitoring." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 235249, 1988

74. Sarda, N. L. "Extensions to SQL for Historical Databases." IEEE Transactions on Knowledge and Data Engineering. 2 (2): 220-230, 1990.

75. Segev, A. and H. Ganadhi. "Event-Join Optimization in Temporal Relational Database." in proceedings of 15th VeKy Large Database Conference. pp. 205-216.

76. Segev, A. and A. Shoshani. "Modeling Temporal Semantics." Temporal Aspects in Information Systems. Rolland, Bodart and Leonard ed. North Holland. pp. 47-57, 1988

77. Sernadas, A. "Temporal Aspects of Logical Procedure Definition." Information Systems. 5 : 167-187, 1980.

78. Shoham, Y. and N. Goyal. "Temporal Reasoning in Artificial Intelligence." EXploring Artificial Intelligence. Shrobe ed. Morgan Kaufmann. pp. 419-438, 1988

79. Snodgrass, R. "The Temporal Query Language TQuel." ACM Transactions on Database Systems. 12 (2): 247-298, 1987.

80. Snodgrass, R. and I. Ahn. "A Taxonomy of Time in Databases." in proceedings of SIGMOD Conference. pp. 236-246.

81. Stam, R. B. and R. Snodgrass. "A Bibliography on Temporal Databases." Database Engineering. 11 (4): 231-239, 1988.

82. Strasser, A. Strukurierte Darstellung juristishen Wissens. Technische Universitdt Mfinchen, Institfit ffir Informatik, Arcisstr. 21, 8000 MQnchen 2, W Germany, Forschungsberichte KfAnstliche Intelligenz, 1989.

83. Strasser, A. C onsistency- Checking of Legral Contracts Using a Teml2oral Model. Technische Universitdt MfAnchen, Institilt Mr Informatik, Arcisstr. 21, 8000 MfAnchen 2, W Germany, Forschungsberichte Kiinstliche Intelligenz, 1990.

84. Tansel, A. "Non-First Normal Form Temporal Relational Model." Database Engineering. 11 (4): 224-230, 1988.

85. Tansel, A., E. Arkun and G. Ozsoyoglu. 'Time by Example Query for Historical Database." IEEE Transactions on Software Engineering. 15 (4): 464-478, 1989.

86. Tsang, E. P. K. "Time Structures for Al." in proceedings of 10th International Joint Conference on Artificial Intelligence. PP.

87. Vrana, R. "Historical data as an explicit component of land information systems." International Journal for Geographical Information Systems. 3 (1): 33-49, 1989.

88. Worboys. "The role of modal logics in the description of knowledge in a geographical information system." in proceedings of Cognitive and Linguistic Asl2ects of GeograRhic SRace, NATO ASI, (Spain 1990a).

89. Worboys, M. F. "Reasoning about GIS Using Temporal and Dynamic Logics." in proceedings of The Temporal GIS Workshpp. pp. University of Maine, Orono, Maine, October 12-13, 1990b.

90. Worboys, M. F. 'Temporal Logic: presentation." The Temporal GIS Workshop. University of Maine, Orono, ME, October 1213, 1990c