1988

Temporal data, temporal data models, temporal data languages and temporal database systems.

Hom, Donald Dokeung.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

H7286

TEMPORAL DATA, TEMPORAL DATA MODELS,
TEMPORAL DATA LANGUAGES and
TEMPORAL DATABASE SYSTEMS

by

Donald Dokeung Hom

June 1988

Thesis Advisor: David K. Hsiao

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL (If applicable) Code 52	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS			
	PROGRAM ELEMENT NO.	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

TEMPORAL DATA, TEMPORAL DATA MODELS, TEMPORAL DATA LANGUAGES and TEMPORAL DATABASE SYSTEMS

12. PERSONAL AUTHOR(S)
Om, Donald D.

13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM _____ TO _____	14 DATE OF REPORT (Year, Month, Day) 1988 June	15 PAGE COUNT 70

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official
policy or position of the Department of Defense or the U.S. Government.

17.	COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Temporal Database; Data Models; Data; Data Languages

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The study of temporal database systems is relatively new in the field of computer science.
Two developments have led to the present interest. The advances of the storage technology
for large amounts of data and applications's requirement for time-dependent data have prompted
our study of temporal databases. This thesis conducts a survey of the major research areas
concerning temporal databases. Temporal data, taxonomies of temporal data models, temporal
data languages, and temporal database systems are presented. It is argued here that future
database systems should handle the temporal domain by an integrated temporal database system.

By understanding the present technology and the need of temporal database systems, our
research in the area of real-time temporal database systems can begin. It is the purpose of
this thesis to provide the background information and research references of temporal database
systems as a first step towards the real-time database system research. Real-time database
systems are time-constrained and temporally constituted. Solutions in temporal database
systems can contribute to the design of real-time military applications using temporal database
computers.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Prof. David K. Hsiao	22b. TELEPHONE (Include Area Code) (408) 646-2253	22c. OFFICE SYMBOL Code 52Hq

DD FORM 1473, 84 MAR
83 APR edition may be used until exhausted
All other editions are obsolete

☆ U.S. Government Printing Office. 1986—606-24.

UNCLASSIFIED

TEMPORAL DATA, TEMPORAL DATA MODELS, TEMPORAL DATA LANGUAGES and TEMPORAL DATABASE SYSTEMS

by

Donald D. Hom
Captain, United States Marine Corps
B.S., Rutgers University, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

June 1988

ABSTRACT

The study of temporal database systems is relatively new in the field of computer science. Two developments have led to the present interest. The advances of the storage technology for large amounts of data and applications' requirements for time-dependent data have prompted our study of temporal databases. This thesis conducts a survey of the major research areas concerning temporal databases. Temporal data, taxonomies of temporal data models, temporal data languages, and temporal database systems are presented. It is argued here that future database systems should handle the temporal domain by an integrated temporal database system.

By understanding the present technology and the need of temporal database systems, our research in the area of real-time temporal database systems can begin. It is the purpose of this thesis to provide the background information and research references of temporal database systems as a first step towards the real-time database system research. Real-time database systems are time-constrained and temporally constituted. Solutions in temporal database systems can contribute to the design of real-time military applications using temporal database computers.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. THE NOTION OF TIME

The concept of time has interested man from the very beginning of human existence. Experts in the field of philosophy, mathematics, logic, physics, and others have discussed the concept of time without a definitive definition. In Webster's New World Dictionary, time is defined as the period during which something exists or happens. But time is also valuable information concerning the life span of that something in the real world. Time is universal. A classic representation of an object or thing is a three dimensional description of its height, width and length. Little or no attention is paid to the fourth dimension, that of time. An object from a person's view is a continuity of changes over time. The value of a dollar today is not worth the value of a dollar ten years ago, due to inflation. Time has devaluated the dollar, a fact that should be represented in addition to its size and thickness.

B. CONVENTIONAL DATABASES LACK TEMPORAL SUPPORT

Present day use of database systems do not represent time aspects of data. They only reflect the most current view, a snapshot of the real world at a specific moment in time. There are several drawbacks to this scheme. First, the data in the database is only valid for the most recent updates. Any new updates will not be reflected in the database until it is actually entered. Secondly, previous information stored in the database is forever lost in an update or deletion. Database systems have been developed to model the reality, which should include historical data. The three well known data models, namely, relational, hierarchical and network, do not address the issue of time. Ad hoc methods, such as backups, checkpoints and transaction logs have been used for the retrieval of

historical data. Additionally, attributes involving time have been handle by the application program, where dates, for example, are values of some specific type, usually integers.

The argument for the database system to manage and control the temporal domain versus the application program is made in this thesis. Further, it is argued that ad hoc methods of collecting temporal data should be replaced by an integrated temporal database system.

C.   PREVIOUS RESEARCH

Temporal databases have been proposed to overcome the drawbacks of conventional database systems. The problem of integrating temporal data with a database system has been known for some time. Research and interest related to processing the time-embedded data is increasing exponentially, as illustrated in Figure 1.1. Any introduction to the topic of temporal domain would not be complete without the two major survey papers done on the subject. An extensive listing of about 70 pre-1982 time-related works were compiled by Bolour and his colleagues in [Ref. 1] , primarily focusing in the areas of artificial intelligence, logic and linguistics. Edward McKenzie followed Bolour from 1982-1986, with over 80 bibliographies in the area of the time-related research in databases [Ref. 2]. The bibliographies were classified according to the taxonomy developed by Snodgrass and Ahn [Ref. 3] which distinguished their capability to represent temporal information in a database system. The taxonomy is to be discussed in a later section.

D.   STORAGE METHODOLOGY AND TECHNOLOGY

A characteristic of temporal data is the non-deletion policy [Ref. 4] , since data provide a progressive history of transactions of the database. *Non-deletion* simply means once a record is inserted, it cannot be deleted except to correct an error in the case of the

2
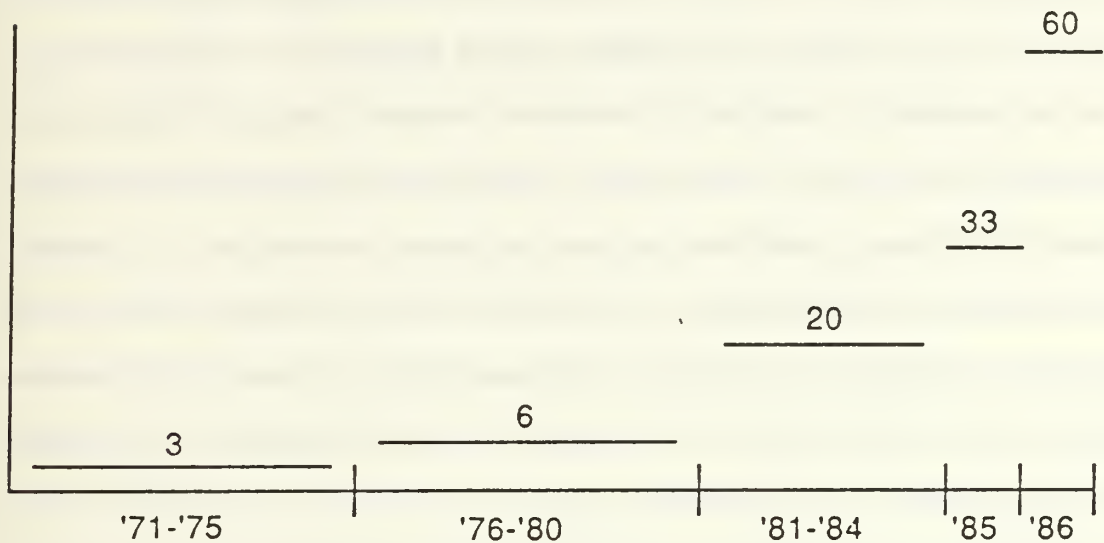
(Reprinted from [Ref. 2])
Figure 1.1   Average Number of Papers Appearing Annually.

historical database, which is defined in chapter II. One of the major obstacle of implementing temporal databases is the physical storage limitations. Enormous amounts of historical data require an efficient storage medium with the capacity to handle large volumes of data. Previously, any access to temporal data was limited to an offline-mode use of magnetic tapes or microfiches, for the purpose of archival storage. On the other hand, a practical implementation of temporal databases would require sufficient disk capacity and an online capability, providing direct access to the data. The technology of the past did not provide this capability and therefore, the study of temporal databases was not enthusiastically pursued. As storage costs continue to decrease and new

technological developments with mass storage become more apparent, the interest in temporal databases is emerging. An example of the technology found in [Ref. 5] and [Ref. 6] is optical disk with write-once capability, the 150-microsecond access time and the 5-gigabyte capacity. Another recent development is multiple optical disks with storage capacity ranging from 1250 gigabytes to 2000 gigabytes, at a maximum access time of 5 seconds [Ref. 7] and [Ref. 8]. Additional references on optical disks can be found in [Ref. 9] and [Ref. 10]. The advent of optical disk creates other problems. Write-once technologies prohibit rewriting and reorganizing of data that is commonly done in conventional database systems. New access methods and storage structure must be developed to manage the write-once storage and achieve reasonable performance. However, magneto-optical disks which combine the write-many-times magnetic-disk characteristics and the high-density-storage optical-disk characteristics are emerging [Ref. 10]. The technology for temporal data is promising.

II. TEMPORAL DATA AND THEIR PHYSICAL ORGANIZATIONS

A. WHY TEMPORAL DATA?

The term *temporal* according to Johnson [Ref. 11] is applied to the data item which is "stamped" with the time during which it is valid or in effect. The primary purpose for temporal data is to provide information on past states. In other words, historical information of all the transactions of the database determined by some time element. This is a major change in concept for databases. In the past, one and only one value was associated with an attribute. With temporal databases more than one value can be assigned to an attribute. For example, the marital status can be both married and single at different times or views of the database. If a data language as an interface is provided, then *historical queries* can be formulated against the database. In many instances, applications need both current and past data. The impetus for the temporal database is perpetuated by application's requirement for historical data. Some of the outstanding applications are introduced in the sections followed.

1. The Trend Analysis

There are several distinct advantages of incorporating temporal data into a database management system. With temporal data, it is possible to have several different versions of data as it evolves through time, known as *version management*. A useful application derived from version management is trend analysis [Ref. 4]. *Trend analysis* relies on historical data to predict future events. Trend analysis has also been referred to as time-series analysis. Businesses could certainly benefit from this application of temporal data.

In the military establishment, the use of computer systems are pervasive. The military relies on advance technology, which inevitably uses information processing

5

capabilities, to counter the advantages of the conventional forces in the Soviet-bloc countries. Computer systems are used in aircrafts, tanks and guided missiles just to mention a few. As part of a bureaucracy, the Department of Defense (DOD) generates its fair share of administrative tasks. Computers are used to manage personnel matters, maintain supply inventories and interface with communication and control systems. Trend analysis can certainly be applicable to a wide variety of military applications. By providing managers and planners with the information available from trend analysis, better decision making capabilities should reduce the level of time and effort devoted to administrative duties.

2. The Retrospective Analysis

At the opposite end of the spectrum is *retrospective analysis* which provides the application the capability to ask what-if types of questions. For example, what if the price of a product was raised by one dollar instead of fifty cents in 1985. The database management system can substitute the information into the previously specified version and bring forward the computed result to the present time. This is particularly important in business planning and scientific research.

The military is a major supporter of many such research and development programs. Retrospective analysis can be used to reduce the overall cost of these program. For example, in the testing phase, tests are conducted repetitively for a range of values of a specific variable with all other variables constant, which can become very expensive. With retrospective analysis, the system performs the test more efficiently and at a lower cost. The savings in cost can then be used in other areas of a pressing need.

3. The Retroactive Change

One of the major shortcomings of conventional databases is their lack of *retroactive changes* in the database system. If an update becomes effective the moment the fact is entered into the system, no problems occur. However, if the update refers to

6

something in the past, a loss of information will result. The fact that the change was retroactive cannot be represented. A query in this environment can result in two answers, both of which are correct, because of the overlap caused by the retroactive change. A typical example is a raise that is retroactive to a specific date. An employee's salary on 1/1/88 is $1,000. On 3/1/88 the employee's salary is changed to $2,000, but retroactive to the first of the year. There is no way to capture the information regarding retroactive change in conventional database systems. A query to determine the employee's salary on 2/1/88 could result in the value of $1,000 or $2,000, depending on the circumstances surrounding the query. If the answer is $1,000, the salary was not changed as of the query. If the answer is $2,000 the salary has been retroactively changed. Any queries of the salary after 3/1/88 will result in the answer of $2,000. This problem causes inconsistency in the database. The use of temporal data can represent the salary change retroactively without the loss of information.

4.   The Postactive Change

Frequently, changes to data are known in advance of the event, known as *postactive change.* In conventional database systems, this fact cannot be represented efficiently. For example, an employee's salary is $1,000 as of 1/1/88. A raise of $1,000 has been approved by the management to be effective on 3/1/88. This information cannot be entered into the database until 3/1/88. Otherwise, a query of an employee's salary before 3/1/88 will result in the value $2,000, an incorrect response, since the true value is $1,000. The ability to enter information into the database concerning a future event is facilitated through the use of temporal data. Otherwise, ad hoc methods which require manual intervention is needed.

5.   Additional Applications of Temporal Data

Besides the ongoing technological developments with storage devices, serving as a motivating factor for the temporal data storage, many applications need to maintain a

complete record of operations over the life of the database. These applications require current, past and future data, providing a complete history and future of transactions of the database. Interest in the temporal data storage is prompted by new applications that inherently deal with time and provide a historical perspective and future perspective of data.

One of the first implementation and use of temporal data was Time-Oriented Database (TOD) [Ref. 12] designed to meet temporal-specific requirements with regard to medical applications. Its data types are of a medical nature and the notion of time is associated with individual patient's recurring visits. It is implemented as a library of shared programs, handling a form as a three dimensional table, with the patient, patient visits and visit times as its dimensions.

The database is implemented as a set of files maintained by some local file-service system, and provides the following three categories of data:

1. The Schema files: They include source statements of the Time-Oriented Database-Data Definition Language (TOD-DDL) in one file and an inverted list of it for efficient interpretation of source statements, i.e., queries.

2. The Data files: They include a patient header (non-chronological patient information) and visit parameters (visit digital information). The latter are indexed on visit numbers. All the records of the file are arranged as doubly linked lists of visit records, where pointers are the visit numbers, which allows cross-referencing via the files' own index.

3. Accessing files: They are created upon request, usually in a batch mode, and include various secondary indices or statistical summaries to facilitate faster access and processing speeds.

The growing use of *Decision Support Systems* (DSSs) requires such systems to explicitly handle time and historical information [Ref. 12]. Businesses rely on such systems to provide analysis of historical data and to predict future events, i.e., trends

8

analysis. Retrospective analysis is also important for corporations. All of the functions previously mentioned is predicated on the use of historical data that deal with the time element.

Another area of interest where applications can benefit from the use of temporal data is *Scientific and Statistical Databases* (SSDBs) [Ref. 12]. Physical experiments, measurements, simulations and collected statistics are usually in the time domain, an analysis of which requires temporal support. These applications are inherently dependent on time and serves as an incentive to develop temporal support for them.

A proposal to use temporal data in real-time computer systems for military applications has been made. One example is the firing control computer software for guided missile [Ref. 13]. Real-time databases are time-constrained and temporally constituted. The necessity for the system to be responsive in real-time is critical. Previous applications do not have the requirement to support real-time applications. The implications of the new requirement need further research. Areas it may affect include access methods, physical organization and the design of the real-time DBMS.

Other applications have been identified which could benefit from the temporal support. They include engineering database, econometrics, survey, policy analysis, office automation and capacity planning to name a few. An exhaustive listing of all the applications that can benefit from the development of a temporal system is only limited by time, space and man's imagination. It is sufficient to say, the need for temporal database systems is driven by a very wide variety of application requirements.

B. DEFINITIONS AND REQUIREMENTS

As more papers are published on the subject of temporal support for databases, a proliferation of terms used to describe the temporal domain have appeared leading to

confusion. In this section, the definitions of temporal data are given. These definition include the temporal requirements and examples are provided to clarify concepts. James Allen's contribution is the introduction of an interval-based temporal logic consisting of time points and time intervals [Ref. 14]. The works of Snodgrass and Ahn, developing three different classifications of time, attempted to rectify the situation by establishing a standard from which to analyze the temporal domain. This paper points out that the three different times merely fall under the temporal logic of time points and time intervals.

1. Time Points Versus Time Intervals

In referring to temporal data by a query language, the notion of *time points and time intervals* becomes an important consideration. The query language must be capable of handling new semantic issues dealing with time-embedded data. Most spoken languages use the tense to describe the relationships of times and events. For example, event A happened before event B with respect to some reference point. A closer examination of an event reveals a problem with this approach. Some events appear to be an instantaneous time point, but in reality is made up of very small intervals. An event, such as turning on a flashlight can be composed of finding the switch, placing your hand on the switch and turning the switch to activate the flashlight. Time points can always be magnified, but prove useful if they are viewed as individual time intervals. The informal notion of time points composed of very small intervals becomes useful in the temporal application.

A description of facts in the world consists of an interval of time in which it is valid. The flashlight example showed time points are made up of very small intervals. To represent the period when the flashlight was on, the time interval before and after when the flashlight was off can be used. Intuitively, the conclusion we have reached above is obvious. The problem deals with the representation of the endpoints in an interval. The interval can be either opened or closed. If it is opened, there exists a time

point within the interval where the flashlight is neither on nor off. This creates a rather serious logic problem in the temporal domain. However, if the interval is closed, then this can be interpreted to mean the flashlight is both on and off for a time point within the time interval, a more serious problem than the first. One apparent solution to this problem is to adopt a convention that a interval is closed at its lower end and open at its upper end. Therefore, each interval would have only one endpoint avoiding the problem of two endpoints in an interval. This solution only reinforces the unique notion of time which cannot correspond to the model of time based on points on the line of real numbers. James Allen's definition of a time interval, if intervals could be modeled by their endpoints, is as follows: Assuming a model consisting of a fully ordered set of points of time, an interval is an ordered pair of points with the first point less than the second. Relationship between intervals can then be defined according to Figure 2.1. assuming for any interval t, the lesser endpoint is denoted by t- and the greater by t+. A

Interval Relation	Equivalent Relations on Endpoints
t < s	t+ < s-
t = s	(t- = s-) & (t+ = s+)
t overlaps s	(t- < s-) & (t+ > s-) & (t+ < s+)
t meets s	t+ = s-
t during s	((t- > s-) & (t+ =< s+)) or ((t- >= s-) & (t+ , s+))

(Reprinted from [Ref. 14])
Figure 2.1   Interval Relation Defined by Endpoints.

further subdivision of the interval is possible. The inclusion of the inverses of the interval relations are introduced. There are thirteen possible relationships as depicted in Figure 2.2. A complete discussion of temporal interval relation is found in [Ref. 14].

a. The User-defined time

An example of a time point is the *user-defined time,* the first of the three kinds of time introduced by Snodgrass and Ahn. User-defined times, also known as application times become part of the record schema and are in the form of mm/dd/yy, i.e., for dates. The date can also be referred to as a time-stamp. The values of the user-

Relation	Symbol	Symbol for Inverse	Pictorial Example
X before Y	<	>	XXX YYY
X equal Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

(Reprinted from [Ref. 14])
Figure 2.2   The Thirteen Possible Relationships.

defined temporal domain are not interpreted by the DBMS and are thus the easiest to support. They can be treated like a regular attribute in a record. The requirements for the user-defined time is the internal representation, usually in integers and the input/output functions. DBMS that support the concept of the user-defined time have been implemented in such systems as Query-by-Example, Enform DBMS, MicroIngres and others.

b. The Transaction time

Transaction times are the second of the three kinds of time defined by Snodgrass and Ahn. A transaction time comes under the auspices of time interval and is defined as the time at which the data item concerning the event was stored in the database. In other works, the transaction time has also been called physical time, registration time, data-valid-time-from/to and start/end time. It is of the form mm/dd/yy - mm/dd/yy and thus consists of two time-stamps. The first time-stamp entry is created with the current time and a null value in the second time-stamp entry when the tuple is first inserted into the database. The second time-stamp entry is updated with the current time when the tuple is deleted from the database.

c. The Valid Time

Very similar to the transaction time, the *valid time* is the last of the three kinds of time. It also comes under the classification of a time interval. However, a valid time is defined as the time an event occurs in the real world. Other authors have referred to the valid time as the logical time, event time, effective time, state, and start/end time. The valid time is also of the form mm/dd/yy - mm/dd/yy. A special case of the valid time is event relation, which only requires one time-stamp. An event relation is binary where the event is either on or off.

2.   Record Versioning

Two schemes to represent temporal version were identified by Ahn [Ref. 4]. The first is *record versioning*. In a relational table each row (corresponding to a record in conventional file) has the addition of an time interval [time_from time_to], indicating the period the row is valid, see Figure 2.3. When a row is first place in the table, the time_to component of the time interval is set to "∞" meaning the row is currently valid. The deletion of an existing row in the table changes the time_to component form "∞" to some time, usually the current time, although it can be explicitly stated. The replacement of an existing row consist of a delete operation as described above and a new version of the row augmented with the appropriate interval. The valid time and transaction time are examples of record versioning. They both require a time interval to represent a period in which something is true. The obvious drawback to this scheme is the amount of redundancy created in the secondary storage. An update to a single attribute in a row requires the whole row be recreated. Even though the new storage technology is emerging, this scheme consumes valuable storage spaces.

Name	Title	Salary	(time_from, time_to)
John	Programmer	25	(Jun 81, Sep 82)
John	Programmer	30	(Sep 82, Mar 83)
John	Manager	30	(Mar 83, Dec 84)
John	Manager	35	(Dec 84, ∞)
Tom	Programmer	27	(Sep 83, Jun 84)

(Reprinted from [Ref. 4])
Figure 2.3   A Relation in Record Versioning.

3. Attribute Versioning

Attribute versioning is an alternative scheme to represent temporal version. In attribute versioning, an attribute is considered dynamic or static. A dynamic attribute changes over time, while a static attribute is constant and does not require any temporal support. In the case of the dynamic attribute of a record, it is represented by a set of <value,interval> pairs, where interval is similar to record versioning except that the interval is associated with each version of an attribute value, see Figure 2.4. The operations for the interval in attribute versioning are similar to record versioning.

a. Time Sequence

Arie Segev and Arie Shoshani's modeling of temporal data is an example of attribute versioning [Ref. 15]. The concept of a *time sequence* is the sequence of values in the time domain for a single entity instance, such as the salary history of a person. A temporal data value is defined for some object (e.g., a person), at a certain time point (e.g., March, 1988), and for some attribute of that object (e.g., salary), thus forming a triplet (s,t,a) where s is the surrogate for the object, t is the time and a is the attribute value. For a given surrogate the temporal data value are ordered in time. In other words, they form an ordered sequence called time sequence represented by

Name	Title		Salary	
John	Programmer	(Jun 81, Mar 83)	25	(Jun 81, Sep 82)
	Manager	(Mar 83, ∞)	30	(Sep 82, Dec 84)
			35	(Dec 84, ∞)
Tom	Programmer	(Sep 83, Jun 84)	27	(Sep 83, Jun 84)

(Reprinted from [Ref. 4])
Figure 2.4   A Relation in Attribute Versioning.

<s,(t,a)*>, where (t,a)* is a sequence of zero or more time-value pairs for a given surrogate.

b.   Properties of the Time Sequence

There are four properties related to time sequence; they are the time granularity, life span, regularity and type. These properties allow a uniform treatment of time sequences. First, operators can be defined for time sequences of different types. Secondly, the physical structure can be designed in the same manner for different types of time sequences. Finally, the properties permit a more efficient design for storage and access.

(1) Time Granularity.   The *time granularity* determines the size of an interval between time points. According to [Ref. 16] two representation are permitted: ordinal and calendar. Ordinal representation states the time points are counted by integer ordinal position (1,2,3,...). Calendar representation takes on the values: year, month, day,...,seconds,etc.

(2) Life Span.   The *life span* defines the period in which the time sequence is valid. The use of a start_times and end_times is represented by ordinal or calendar. The time granularity and life span need not have the same representation. There are three cases of the life span of interest.

1. Start_point and end_point fixed.

2. Start_point is fixed and end_point is current time.

3. A fixed distance is defined between the start_point and the end_point. The end _point is "current time" and the start_point is dynamically changed to maintain the fixed distance from the end_point.

The life span can be disjoint, non-continuous segments represented by the use of "null" value for the time-point value. This indicates a value does not exist for this time point.

(3) <u>Regularity</u>.   The *regularity* is used to distinguish regular and irregular time sequences. Two factors make the distinction of regular from irregular time sequences necessary. The regularity is a semantic property needed by the system to interpret time sequences and is important in the physical organization of time sequences. Regular time sequences contain a value for each time sequence in the interval, and have a value at each time point, usually measured by some sort of device. Irregular time sequences are unpredictable events that occur with no regularity and therefore contain values for only a subset of time points in the interval. A typical example of a regular time sequence is scientific experiments where measurements are taken at regular intervals. Sales is an example of irregular time sequence, since there is no way to predict when this may occur.

c.   The Time-Sequence Collection

A useful way of thinking about the time-sequence is collecting the objects that belong to the same class. A class is any collection of objects that have the same attributes, such as a person. Each object within the class has a unique identifier called the surrogate. A composite class requires more than one surrogate for the identifier. For example, the 'attendance' class has surrogates student and course. *Time-sequence collections* provide the capabilities to address an entire class of temporal attributes and relate them to other possible non-temporal attributes of the classes.

(1)   <u>The Simple Time-Sequence Collection</u>.

The component of a simple time-sequence collection, namely S, T and A, all represent a single element. A simple time-sequence collection is the collection of all the temporal values for a single attribute for all of the surrogate of a simple class. A simple class is a class with a single surrogate as its identifier. It is represented graphically in a two-dimensional space as shown in Figure 2.5. Each row represent a

17
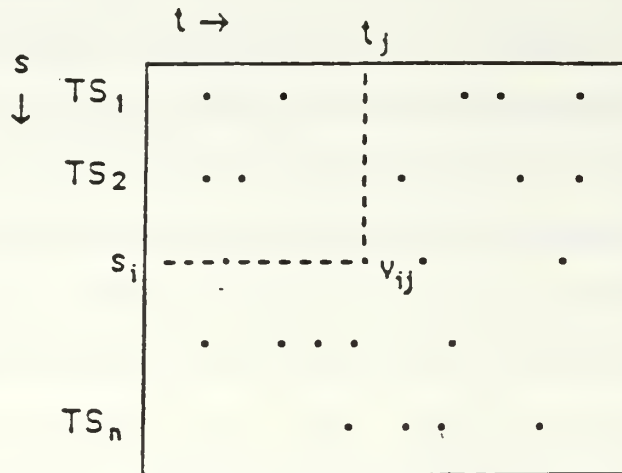
(Reprinted from [ref. 15])
Figure 2.5   A Representation of a Time Sequence Collection.

surrogate of a time sequence. The data indicates the presence of a temporal value for
some surrogate s_i and some time t_j.

(2)   The Complex Time-Sequence Collection.

The components in a *complex time-sequence collection* is not
restricted to a single element. The case where S is not a single element is denoted by
$(\bar{S}, T, A)$. S in this situation is a composite class. The same two-dimensional pictorial
representation applies except that the rows are labeled with the composite surrogate
identifier of the class. The case where T is included has more than one time sequence
associated with temporal values. Candidates for the time attribute are the valid time and
transaction time which were discussed in length in an earlier section. This situation is
also represented as $(S, \bar{T}, A)$. The third case involves the attribute A occurring more than

18

once as in $(S,T,\overline{A})$. An example would be a collection of several pieces of information taken at regular intervals. Any combination of the three cases can occur simultaneously.

C. CONTRASTS AND DIFFERENCES

In this section, time points versus time intervals and record versioning versus attribute versioning are analyzed. They become a fundamental aspect of any temporal data model. Some of the affects will impact the storage structure, access method and query language of the database system. The design of the DBMS will be predicated on the selection of these properties and schemes.

1. Time Points and Time Intervals

Physically the difference between time points and time intervals are obvious. Time points use a single time-stamp to represent an event and time intervals use two time-stamps to represent a period of time. Time points are relatively easy to support through the application software. Treated like normal integers, time points are manipulated by user-defined functions. On the other hand, time intervals are semantically more complex, requiring new access methods and query languages.

Within time intervals the distinction between the valid time and the transaction time is important. The transaction time is a representation of the time when the data concerning an event is stored in the database. The valid time models reality, the time the data actually happened in the real world. This is manifested in the data model structure. The use of transaction times limits the flexibility of the database. Once a time is added to the database, it can never be modified. This in turns means that any addition to the database is restricted to appending only. This is not the case with the valid time. Discrepancies between information in the database and the actual event are subject to misinterpretation. These errors can be modified if we utilize valid times.

19

2. Record Versioning and Attribute Versioning

Record versioning augments the entire row with two time-stamps. The disadvantage to this method is the duplication of data during a single attribute update. Imagine if you have a 2,000 byte record with 1 byte representing marital status. If a person becomes married the entire record is now invalid. A new record must be created with a single byte change to reflect the new marital status. The other remaining bytes were unchanged. For this reason attribute versioning is designed to avoid the waste of storage space. Only those attributes that have changed over time are augmented with time intervals. The contents of attribute versioning and record versioning both contain the same amount of information. It is possible to derive one from the other. The conversion from record versioning to attribute versioning is done according to the operations (insert, delete, replace) described in attribute versioning. The conversion from attribute versioning to record versioning requires two operations, UNNEST and SYNCH [Ref. 4]. The UNNEST operation developed by Jaeschke and Schek transforms a non-first-normal form relation to a first normal form. After completion of the UNNEST, the SYNCH operation determines the largest interval for which attribute versioning is in effect. Null intervals are deleted and the transformation is complete.

D. PHYSICAL ORGANIZATIONS

This decade has been called the information age due to the advances in computer technology, specifically, storage devices, which has made the concept of time-embedded databases feasible. The key element for implementing temporal data is an effective storage structure. This structure must address the very nature of temporal data, its large volume of data. Temporal data are historical, which may be kept around for years. As years past the amount of information accumulated is significant, yet the database system must handle storing and accessing the data efficiently. An effective structure must

address several characteristics of temporal data. The first is the non-deletion property, already mentioned in chapter I. Since a record will never be deleted in the case of a historical database except for error correction, designing an effective storage structure and accessing methodology is essential. Secondly, the design for temporal data should consider the application's intended use. Most queries will access current data, while temporal queries are sporadic. Access methods must be developed to prevent penalizing non-temporal queries yet provide reasonable access time for historical queries. Temporal data generate several versions of data. If conventional access methods are used, such as hashing and indexing, performance degradation occurs. This is caused by overflow chains and collisions. The need to develop storage techniques and access methods specifically designed for temporal support is therefore identified. Without new developments in this area, the implementation of temporal data will indeed be difficult.

As always, there are several different physical organizations of temporal data. Certain properties unique to temporal data should be considered. Three widely held views of organizing temporal data are discussed. The first view is a two level storage structure as identified in [Ref. 4]. The second approach views temporal data as time sequences, for which a two-dimensional array is the suggested organization [Ref. 16]. The third approach is that of a multidimensional partioning of the file [Ref. 17]. Each is discussed below.

1.   The Two-Level Storage Structure

The two-level storage structure is designed to take advantage of certain characteristics unique to temporal data. The existence of a large number of temporal versions render conventional access methods inefficient. Other access methods, such as B-trees, grid files, extendible and dynamic hashing have been suggested. But these methods require complex algorithms and create excessive overheads. The two-level storage structure is design to differentiate the characteristics between current data and

history data. Current data is relatively static, after reaching a certain point. History data require large amounts of storage and do not require updates except in the case of adding historical data. Additionally, the pattern of accessing data influences the decision for a two-level storage structure. Current data is more frequently accessed than history data. That seems obvious since users are more interested in the latest information. Therefore, a difference between the two types of data is the storage and accessing requirements.

The two-level storage structure separates the current data from the history data. The current data is stored in an area known as the *primary store*. The history data is located in an area called the *history store*. This provides the capability to have two separate access methods, each design to take advantage of the unique characteristics of the data and store media. The access method used in primary store can support non-temporal queries and more-frequently-required history data. By using different access methods in the two-level storage structure concept, non-temporal queries will not be penalized with long access times.

A variation of the two-level storage structure was first identified by [Ref. 18]. which has proposed a chaining of versions of a tuple in the reverse time order. The beginning of the chain contains the current version of the tuple and the oldest version is last in the chain. If the tuple is deleted, then the chain will be moved to become the history chain. To implement this methodology in a two-level storage structure is simple, Figure 2.6. When a tuple is first inserted, it is placed in the primary store. When the current version of the data is updated to reflect changes, the old version is moved to history store with appropriate updates to the pointers. To access information, the current version is located and, by following the pointer, older versions of the data can be accessed.

22
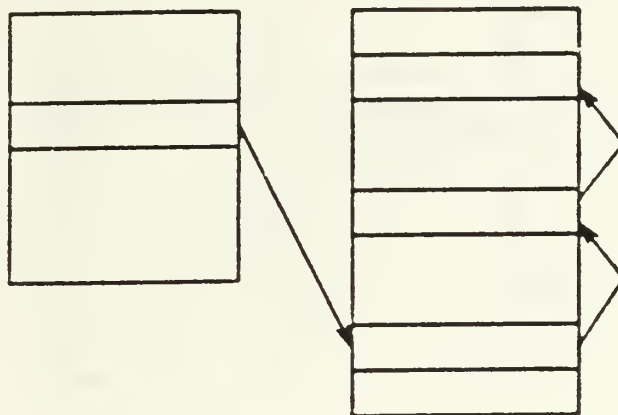
(Reprinted from [Ref. 4])
Figure 2.6   Reverse Chaining.

If the lengths of chains in history store are excessively long, then performance degradation occurs. An accession list is inserted between the primary store and history store. The accession list is nothing more than an index into the history store, see Figure 2.7. The accession list is accessed by a pointer in the current version. The accession list contains information about each version, eliminating the need to physically access each history version. Therefore, only the required history version as determined by the information in the accession list will be accessed. Since an accession list will be frequently accessed as compared to history version, it is kept on the magnetic disk. History versions do not need pointers or temporal information, since it is kept in the accession list.

Another variation is to cluster all history versions of each tuple into the minimum number of pages, see Figure 2.8. This method reduces the disk accesses to
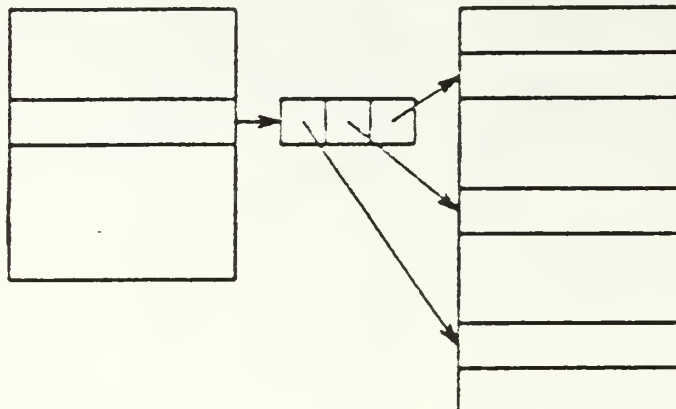
(Reprinted from [Ref. 4])
Figure 2.7 Accession List.
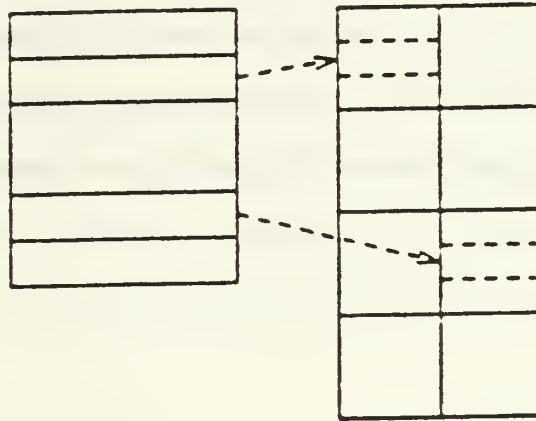
(Reprinted from [Ref. 4])
Figure 2.8   Clustering

history versions but requires a complex algorithm to maintain the clustering and high storage utilization. A simple but inefficient manner to store history data is to assign a page for each tuple. To better utilize storage, several tuples may share the same page. If an overflow occur the page is split into two pages, each containing all the unique tuples selected for the particular page. A link between the primary store and the history store is needed to connect current version with its history version. A physical pointer is used as a link which also requires the maintenance if overflow occurs. Another problem with a physical link is concurrency problems. The primary store is locked during an update. The problem can be resolved with the variations of dynamic storage structure.

A stack consists of tuples with an equal number of history versions in the history store, see Figure 2.9. This application is useful when a fixed number of history versions is needed. As new history versions are added and exceed a predetermined number, the older versions are discarded. This behavior is similar to a stack with finite amount of storage where older versions are pushed through the bottom. Optical disks cannot be used for this method since it requires rewriting some of the data. The drawbacks are the limited number of versions and possibly lower storage utilization.
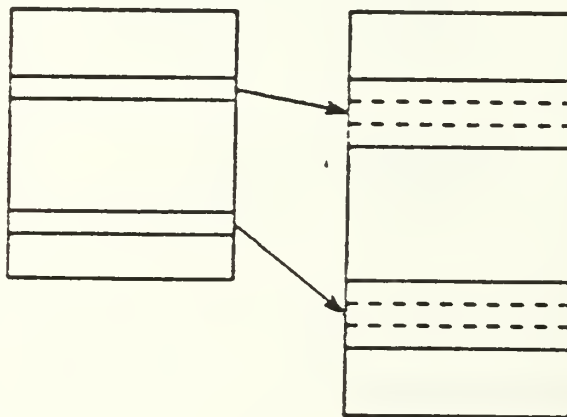
(Reprinted from [Ref. 4])
Figure 2.9   Stacked Versions

Cellular chaining combines some of the advantages of the previous methods. Cellular chaining is essentially reverse chaining with multiple versions in a cell, see Figure 2.10. If an overflow occurs another cell is added to the chain. Clustering occurs within a cell. The cell can be thought of as a stacked version since several versions of the same tuple exists in the cell.

2.   The Two-Dimensional Array

Time sequences are implemented in a two-dimensional array structure and also seek to take advantage of temporal characteristics, see Figure 2.5. Changes to historical
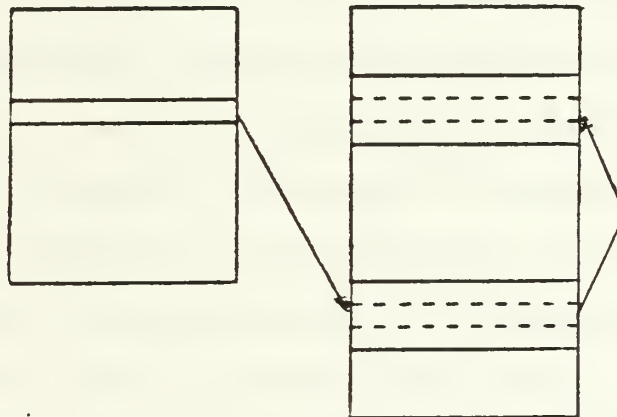
(Reprinted from [Ref. 4])
Figure 2.10   Cellular Chaining

data are limited. As in the two-level storage structure, better utilization of storage and efficient access of data are goals of the two-dimensional array structure for time sequences.

The two indices into the two-dimensional array are on the surrogate and time values. This is efficient because these values are only stored once for each data value. This implementation is a direct result of the goals. Time sequences can be viewed as a time sequence array. This method works well for the regular time sequence array, but creates problems when the time sequence array is sparse. The discussion of the sparse array can be found in any data structure book. There are five assumptions that guide the selection of indexing the surrogate and time domain.

The first assumption is that the order of values in time sequences is important. Frequently, information requested ranges over a time period. If these values are preserved in the physical structure, located in one area, a more efficient access would result. This serves to minimize the input/output (I/O) overhead associated with the disk reads. The second assumption is on random access with respect to the time domain. This enables direct accesses to certain portions of the time sequence. Otherwise, every access would proceed sequentially. The third assumption is on random access to the surrogate. This serves the same purpose as indexing by the time domain. The fourth assumption is on the ordering in the surrogate domain. This forms a partioning of closely related information, which can be easily accessed through indexes. Finally, it is assumed that the secondary index is not required for data values in most applications. Usually, an access to data is determined by the time and surrogate domain.

3. Multidimensional File-Partioning

Multidimensional file-partioning (MDFP) is another approach to organize temporal data. MDFP consists of three levels. The first level consists of a file with

28

information about the partioning points. Partioning points are the possible ranges of values for each attribute. These ranges are then segmented into partitions with the defining cell. Each tuple belongs to a cell as derived from the segmentation. Algorithms to determine partioning points are discussed in detail in [Ref. 17]. This first level information is kept in fast storage. The second level is a directory for each entry of a cell of the partioning. This entry is a pointer to the page that contains information about the tuples of the cell. The last level contains the actual data. The last two levels are kept on the disk storage because of their sizes. This method is particularly useful for calculating statistics for value ranges along the time dimension.

III. TEMPORAL DATA MODELS

A. WHY DO WE NEED MODELS?

A data model is used to describe the logical structure of a database in a database system. One definition [Ref. 19] for a data model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints of a database. Another definition [Ref. 20] of a data model is it represents a small subset of the reality, appropriate to one application of the database concept. In other words a data model is nothing more than an abstraction of the database. For a user a data model facilitates the user's interaction with the DBMS, so the user need not be concerned with the details of the DBMS. This enables the user to concentrate in what the user wants to accomplish with his application instead of how to accomplish it with the DBMS.

B. A TAXONOMY OF TEMPORAL DATA MODELS

Presently, most of the research and work concentrates on extending the relational model to include time-embedded information. Therefore, the focus of this section deals with the relational model. However, of interest is the research being conducted at the International Business Machine (IBM) Scientific Center located in Heidelberg, West Germany, which incorporates the time element in a hierarchical data model. Two approaches to extending the relational model have been identified [Ref. 21]. The first is to incorporate the semantics of time in the database model. The semantics used is based on the formulation of intensional logic, which provides consistency and uniform treatment of time. The logic serves as a formalism for time-embedded database much like the first-order logic serves as a formalism for the relational model. A detailed discussion is found in [Ref. 22]. The second approach is the addition of time associated

30

attributes, for which a taxonomy will be defined.

1.   The Static Database

Snodgrass and Ahn developed a taxonomy which introduced four kinds of databases distinguished by their capability to represent temporal information. The first kind of databases to be considered is called the *static database* or conventional database, which consists of a snapshot of the real world at a particular point in time, although it may not be current. Usually there is a period of time that exists between a change in the real world and the physical update to the database to reflect the change. This causes the database to be inconsistent with the real world until the update occurs. Updates to the database permanently destroys any previous information stored. An example of these conventional databases is the relational database, which can be graphically represented by a two-dimensional table, see Figure 3.1. A major disadvantage of static database is its inability to answer historical queries such as: What was John's salary five years ago? Other methods had to be developed so this type of queries can be answered, since a user
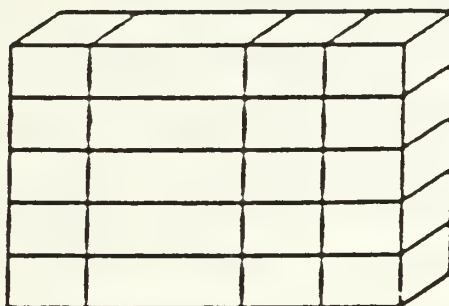
(Reprinted from [Ref. 3])
Figure 3.1   A Static Relation.

may want to ask questions of this type. Static databases also lack the facility to handle trend analysis, retroactive change, and postactive change, which have been addressed in Chapter II.

2. The Static Rollback Database

To solve the problem of losing information in the static database environment, all past states are indexed on time forming a three-dimensional table as in Figure 3.2. In other words, it is a table of a sequence of states. This index called the transaction time is the third axis in the three-dimensional table. To retrieve information in any past state, the term rollback is applied. Rollback is an operation that simply takes a vertical slice of the three-dimensional table. The term also gives this kind of a database the name, the *static rollback database*. A horizontal slice of the cube provides a historical perspective
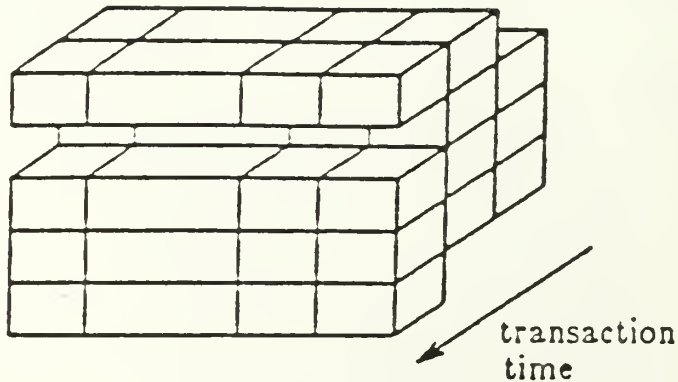
(Reprinted from [Ref. 3])
Figure 3.2   A Static Rollback Relation.

32

of the tuple. Any transaction, whether addition or deletion will be appended to the front of the cube. Therefore, the front of the cube contains the most current version of the database. The problems with this approach are many. The static rollback database represents the history of a database transaction, i.e., the time the entry is put into the database, not the real-world time of the event happening. Additionally, updates to past or future states are not possible. Implementing this sort of database poses several problems. If a tuple does not change between states, the data is duplicated and brought forward to the front of the cube. This creates an excessive waste of the storage space. To overcome this situation a start/end time of the transaction time is associated with each tuple and maintained by the database. In Lum's paper [Ref. 18] , several other drawbacks of this method have been identified. As the number of versions increases, the performance decreases. This degradation is caused by large relational tables as indexing of all tuples are required. Another problem with this strategy is schema anomalies. Temporal schema anomalies are caused by restructuring the database. By adding new attributes to a relation it may be impossible to enter old data into the new schema. As a database evolves in this environment, a conflict appears. Old relations that do not have time-stamps coexist with new relations that have time-stamps. A similar problem occurs by transforming a non time-varying relation into a time-varying relation by adding the time-stamp. The system must be able to retrieve non time-varying relations, which are valid relations, but contain no time-stamp. A more detailed discussion is found in [Ref. 23]. The advantages of this approach is the ease and quickness of implementing the time-added attribute to the existing database system without major modifications to the system.

3. The Historical Database

The *historical database* differs from the static rollback database by replacing the transaction time with the valid time as the third axis in the three-dimensional table as

in Figure 3.3. A historical database records one historical state per relation. This can be viewed as taking a horizontal slice of the cube. Unlike a static rollback database, errors can be corrected in an historical database, but there is no record of the corrections made and previous states are forgotten. A second difference of the two is that the historical database can record knowledge about the past while static rollback database can only rollback to past states. With the inclusion of the valid time in the database management system, the query language becomes more complex in order to handle the semantics of valid times. Like static rollback databases, the endpoints of the valid times are appended to each tuple for storage efficiency. Another feature that the historical database support is the user-defined time.
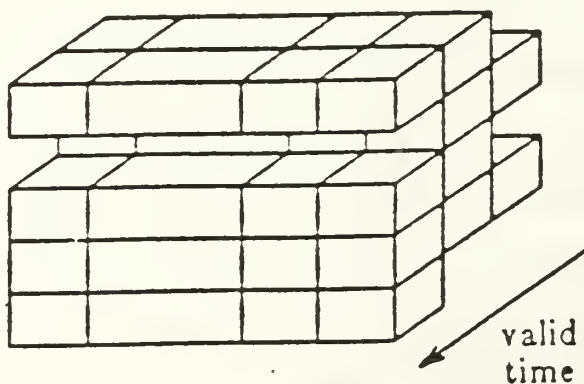
(Reprinted from [Ref. 3])
Figure 3.3   An Historical Relation.

4. The Temporal Database

Temporal databases , see Figure 3.4, take advantage of both the static rollback database and the historical database by combining the valid time and the transaction time into a four dimensional table, enabling the DBMS to view tuples valid at certain moments as seen from some other moment. A complete history of retroactive and postactive changes will be captured in a temporal database. A relation in a temporal database can be thought of as an sequence of historical states of which contains a complete historical relation. Therefore, temporal relations are append-only.
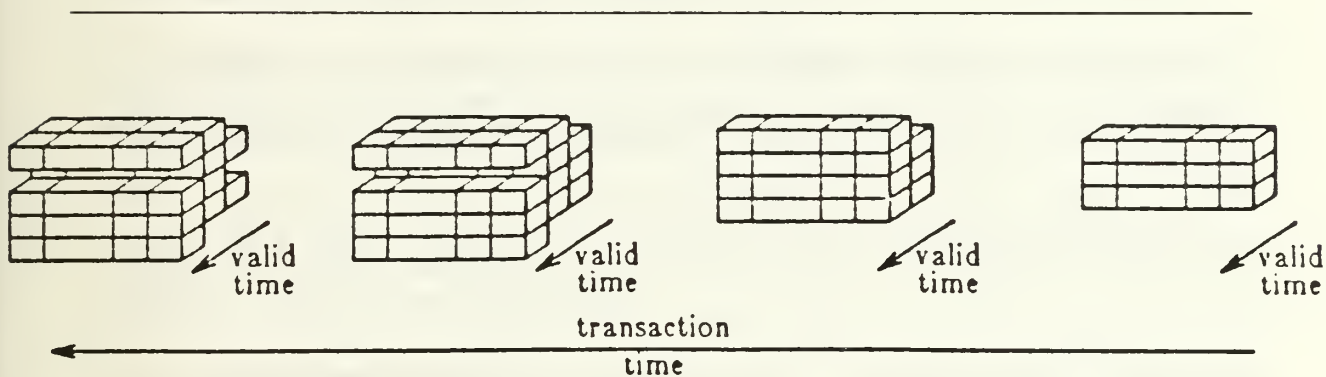
(Reprinted from [Ref. 3])
Figure 3.4   A Temporal Relation

IV. TEMPORAL DATA LANGUAGE

A. WHY DO WE NEED LANGUAGES?

An essential component of a database system is the user-interface. This component is the primary means for a user to control the information, going and coming into the database. The construct is the data language or data manipulation language. According to [Ref. 24] , the data language allows the user to write queries and transactions which consist of generic operations on the database. Another definition is a language that enables users to access or manipulate data as organized by the appropriate data model [Ref. 19]. Typical operations of data languages are retrieval, insertion and deletion of data.

B. A TAXONOMY OF TEMPORAL DATA LANGUAGES

The development of a temporal query language is closely related to the data model. In chapter III, two possible approaches to temporal data models are discussed. By extending the semantics of the relational data model to include time is the first method. Adding temporal attributes to a conventional data model is the second method. The first approach uses temporal logic as the basis for the data language. Many in the research community have developed temporal logic to deal with the time element. A survey of the research is discussed below. In the second approach, the data model is not aware of the time element at all, rather the query language does the work. The language serves as the interface between the time element and the data model. The language translates operations involving time in a form understandable to the relation. These languages have a formal semantic to make understandable the meaning of each construct. The languages are based on relational algebra and relational calculus, which are powerful and flexible.

They provide the capability to prove correctness of operation. Therefore, the taxonomy of the data languages is based on temporal logic and the constructs of relational algebra and relational calculus. Logic developed for the temporal support of data language are introduced with specific references cited where a detailed description is found.

1. Temporal Logic Language XYZ

The XYZ language proposed by Tang implements a limited set of *temporal logic*. Recall that a state refers to a single instance of the world as reflected in the database. Time is perceived to be a sequence of ordered states. Temporal logic is used to determine operations across state boundaries. Temporal logic, through the use of precedence relationship, establishes rules and use of operators to access states. A more detailed discussion of the Temporal Logic Language XYZ is found in [Ref. 25]. Three temporal operations are described below according to Tang.

1. next time This operator enables us to express something that must necessarily be true in the next state of the system. It also enables us to address the contents of program variables in the next state of the system.

2. eventually This operator is different from the next time operator, in that the truth it expresses must not necessarily be true in the next state, although it must become true sometimes in the future. By definition, "sometimes in the future" includes now.

3. necessary The necessity operator is used to state something that must necessarily be true in all system states from now on. We will use it to express invariants. By definition, "from now on" includes now.

2. Modal Temporal Logic

Modal temporal logic is concerned with the multiple states of a time-embedded database system. A fact that is true in one state may not be true in other states. Inferences can also be drawn between the possible states. Additionally, a proposition in a state may depend on other possible states. Modal temporal logic is the idea used to consider the various possibilities among states. Temporal logic is based on a

mathematical foundation. This foundation proves useful in formal correctness proof of modal operators in the language. In [Ref. 26] the use of the relations earlier-than and later-than represent the relationship of possible states.

A modal temporal structure is described, which consists of a set of states. Each state contains a set of proposition that are true and related by an immediate predecessor-successor relation. Then a branch of time is defined as the possible sequence of states for a relation. A unified branching temporal logic is used to project properties of future states. A more detailed discussion, to include syntax, semantic and axiom is found in [Ref. 26]. Although the language is based in the context of a natural language question-answering, it demonstrates the modal temporal logic in sufficient detail. Another formal presentation of modal temporal logic is found in [Ref. 27].

3. An Interval Logic

A pervasive concept in temporal reasoning is that of a property being true for an interval. This forms the basis for an *interval logic* to represent temporal relationships. The use of formulas, defined in the interval logic, derives certain properties of intervals. These properties in conjunction with temporal operators compose new intervals. The development of an interval logic is closely associated with concurrent research in the area of distributed systems, concurrent programs and communications protocol. Therefore, the specification and verification methods are based on the concept of intervals. A formal presentation of an interval logic is found in [Ref. 28].

4. Intensional Logic

Intensional logic is a formal presentation to represent the semantics in time-embedded database systems. Intensional logic is analogous to the formalism of first-order logic for relational database model, likewise, intensional logic is a formalism for understanding semantic in the temporally oriented data model. Intensional logic is based on the work of Montague. The basic idea behind intensional logic is that time-varying

38

attributes are represented as functions from a set of times into values. Intensional logic is a typed, higher order lambda calculus. Every expression in intensional logic has an associated type to determine the kind of object in the language, which can be assigned an interpretation function. Intensional logic is higher order in the sense it allows quantification over variables of every type. Intensional logic is covered in depth in [Ref. 22].

5. Relational Algebra

The use of *relational algebra* to support the relational data model is well known. Relational algebra is a procedural query language consisting of five basic operations: select, project, cartesian-product, union and set-difference. A relational query expression is satisfy by generating a sequence of operations that answer the query. Extending the relational algebra to include temporal support is founded on several basis. Relational algebra has been rigorously tested and provides a firm foundation upon which to build on. Optimization techniques for the relational algebra is already developed. The use of the extended relational algebra meant additional operators for the temporal domain and new commands that modify the database. Furthermore, new rules and identities are prescribed. In [Ref. 29] new algebraic operators are identified in support of the proposal to use attribute versioning. Time-stamping attributes imply non-first normal form relations, from which the relational algebra is extended. In [Ref. 30] extending the algebra to support transaction time, valid time and user-defined time is the underlying concept.

6. Relational Calculus

Relational calculus is a nonprocedural language. A formal description of what information is needed without specifying the steps to retrieve the data is the basis of the calculus. Of particular importance is the tuple relational calculus of which Temporal Query Language (TQUEL) is based on. Relational tuple calculus is expressed in the

39

form of "there exist" or "for all" through the use of predicate calculus in mathematical logic. QUEL is the query language for the INGRES relational database system. TQUEL is a minimum extension of QUEL, syntactically and semantically, designed to support temporal queries. Three design aspects of TQUEL are important. Any QUEL statements are also valid TQUEL statements. An implication of the previous sentence is such statements have identical semantics in QUEL and TQUEL, when the time domain is fixed. Finally, the constructs of QUEL and TQUEL are direct analogue of each other. Examples are provided to illustrate the temporal constructs of TUEL.

The as of clause in TQUEL is used in a static rollback relation to specify the transaction time, see Figure 4.1.

name	rank	transaction time	
		(start)	(end)
Merrie	associate	08/25/77	12/15/82
Merrie	full	12/15/82	∞
Tom	associate	12/07/82	∞
Mike	assistant	01/10/83	02/25/84

range of f is faculty
retrieve (f.rank)
where f.name = "Merrie"
as of "12/10/82"

rank
associate

(Reprinted from [Ref. 3])
Figure 4.1   An Example of a Static Rollback Relation.

41

The when predicate and a valid clause consisting of start of, precede and overlap is used in a historical relation to specify the valid time, see Figure 4.2.

name	rank	valid time	
		(from)	(to)
Merrie	associate	09/01/77	12/01/82
Merrie	full	12/01/82	∞
Tom	associate	12/05/82	∞
Mike	assistant	01/01/83	03/01/84

range of f1 **is** faculty
range of f2 **is** faculty
retrieve (fl.rank)
where f1.name = "Merrie"
and f2.name = "Tom"
when f1 **overlap start of** f2

rank	valid time	
	(from)	(to)
full	12/01/82	∞

(Reprinted from [Ref. 3])
Figure 4.2   An Example of a Historical Relation.

42

The combination of the constructs in a static rollback relation and a historical relation is used in a temporal relation, see Figure 4.3. A complete presentation of TQUEL and a comparison of TQUEL to other query language supporting time are in [Ref. 21].

name	rank	valid time		transaction time	
		(from)	(to)	(start)	(end)
Merrie	associate	09/01/77	∞	08/25/77	12/15/82
Merrie	associate	09/01/77	12/01/82	12/12/82	∞
Merrie	full	12/01/82	∞	12/15/82	∞
Tom	full	12/05/82	∞	12/01/82	12/07/82
Tom	associate	12/05/82	∞	12/07/82	∞
Mike	assistant	01/01/83	∞	01/10/83	02/25/84
Mike	assistant	01/01/83	03/01/84	02/25/84	∞

range of f1 **is** faculty
range of f2 **is** faculty
retrieve (f1.rank)
where f1.name = "Merrie"
and f2.name = "Tom"
when f1 **overlap start of** f2
as of "12/10/82"

rank	valid time		transaction time	
	(from)	(to)	(start)	(end)
associate	09/01/77	∞	08/25/77	12/15/82

(Reprinted from [Ref. 3])
Figure 4.3   An Example of an Temporal Relation.

V. TEMPORAL DATABASE SYSTEMS

A. THE SYSTEM, MODEL AND LANGUAGE

In the previous chapters, two possible approaches of handling the time element in a database system is introduced. A majority of the researchers have appended a time-stamp at the attribute level or tuple level to incorporate time in the database. Time-stamps are used as a representation for valid time and transaction time. This method requires the data languages to transform queries involving time to a form understandable to the underlying static relations. Examples of such systems are abundant and include TRM [Ref. 31] , CSL [Ref. 32] , LEGOL 2.0 [Ref. 33] , TERM [Ref. 34] and TQUEL [Ref. 21]. Alternatively, some researchers have chosen to extend the semantics of the relational model to incorporate time. Extending the semantics of the data model is facilitated through the development of various temporal logic. The logic becomes an integral part of the data model from which the data language is based upon. Examples of such systems are DMTLT [Ref. 27] and IL_s [Ref. 22].

B. A TAXONOMY OF TEMPORAL DATABASE SYSTEMS

From the above discussion a taxonomy of database systems can be derived. The taxonomy is based on the approach taken to extend the relational model to incorporate time. The taxonomy consists of the Time-Stamp Based Database System and the Temporal Logic Based Database System.

1. Time-stamp Based Database System

The *Time-Stamp Based Database System* is the term defined for databases that append a time-stamp at the attribute level or tuple level. The data language for these systems are based on relational calculus or relational algebra, which acts as an

44

interpretation mechanism. It has the advantage of simplicity. The relational data model is relatively simple to learn and understand. The relational calculus and relational algebra has been formalized and thoroughly studied. From an implementation point of view, adding time-related features to the relational model is the easier approach. Time-stamp based database system can support valid time, transaction time or both.

2.   Temporal Logic Based Database System

The *Temporal Logic Based Database System* is the term defined for database systems to extend the semantics of the relational model to include time. It can be thought of as an approach to develop the logic required for time within the database system. The internal functions of the database system is fully aware of the time element. This approach is more complex. The logic developed is new and less rigorous than those of relational calculus or relational algebra. However, a advantage is the efficiency and better performance achieved through a temporal logic specifically designed for a database system. It is not an "on top" approach but a "within" approach.

VI. <u>CONCLUDING REMARKS</u>

A broad overview of present technology related to handling time in database systems is presented. Basic concepts and fundamental principles associated with temporal data, temporal data models, temporal data languages and temporal database systems are presented. A taxonomy is defined for the data models, data languages and database systems to provide a summarization of the work to date. The compilation of the various works in this paper serves as an index into the various topics of interest in the temporal domain of database systems. The references cited provides a source of additional information related to time in database systems.

Most of the research to date have concentrated on the relational data model, because of its simplicity. Other data models as well can incorporate the time element. The knowledge and experience gained with the relational model should help. Presently, there is no commercial application of time-embedded database systems. They have been restricted to prototype implementation. However, many applications have identified a need for temporal support. This and the increasing improvements in storage technology will make temporal database systems feasible. However, more research is needed to fill this void.

After reviewing the present technology of temporal database systems, no mention was made of real-time temporal database systems. It is the opinion of the author that real-time temporal database systems provide a brand new area of research opportunity. Real-time systems are time constrained. One immediate question is the present technology of temporal database systems adaptable in the real-time environment. With real-time constraints many issues dealing with temporal database systems need to be

rehashed. It is the goal of this paper to provoke new thought in the area of real-time temporal database systems. Further research can be conducted in many various areas such as access methods, query optimization, concurrency control, temporal constraints and database restructuring. Many implications of real-time constraints need to be considered.

VII. ANNOTATED REFERENCES

Abbod, T., Brown, K., Noble, H., Providing Time-Related Constraints For Conventional Database Systems, Proceedings of the 13th Very Large Data Base, pp. 167-175, 1987.

A tuple based historical database is presented. The model supports both physical and logical time. To represent a period of time, version numbers that reflect the ordinal position of the tuple is used instead of double logical time-stamps. The reason for this is because the second logical time-stamp is not available at creation time when used in conjunction with write-once laser disc. The distinction between version and correction update is made. The notion of set-valued attributes much similar to time sequence of Segev and Rotem is introduced. Rules for the user-defined integrity constraints for correction-updates are defined in support of the model. A temporal front-end to Ingres called SIS-BASE is described.

Adiba, M., Quang, N. and Oliveira, J., Time Concept In Generalized Databases, ACM, pp. 214-223, 1985.

The paper is devoted to implementing temporal aspects into the Tigre model, which is an extension of the entity-relationship model. The extension consist of abstract data type concepts to handled generalized document or multimedia document.

Ahn, I., Towards An Implementation of Database Management Systems with Temporal Support, Proceeding of the International Conference on Data Engineering, pp. 374-381, February 1986.

The paper focuses on two aspects of temporal support. First, it distinguishes between two schema structures, tuple versioning and attribute versioning, used in time-embedded database systems. Second, variations of a two-level storage structure used to stored temporal data are presented.

Ahn, I. and Snodgrass, R., Performance Evaluation of a Temporal Database Management System, Proceedings of ACM SIGMOD 86 International Conference on Management of Data, v. 15, pp. 96-107, June 1986.

This paper follows the work done on taxonomy by Snodgrass. A prototype temporal database was built by extending Ingres and utilizing TQUEL. A benchmark was run with the four types of database and two load factors. The results were analyzed to determine

the factors with greatest impact. This is the first paper published, concerning the performance evaluation of temporal database systems.

Allen, J. F., Maintaining Knowledge about Temporal Intervals, Communications of the ACM, v. 26, pp. 832-843. November 1983.

This paper introduces an interval-based temporal logic that is both expressive and computationally effective. Representing the interval can be done by the user input data or the relationship is deduced. A balance is reached by the introduction of reference intervals, which can control the amount of propagation through the system. The notion of time point versus time intervals is presented.

Ariav, G., A Temporally Oriented Data Model, ACM Transaction on Database Systems, v. 11, pp. 499-527, December 1986.

A complete presentation of Temporally Oriented Data Model (TODM) and Temporally Oriented SQL (TOSQL), which is TODM-based specified syntax is introduced. TODM is based on a three dimensional construct of time, object and attribute, much like the HDB of Ahn. A historical development of temporal data is surveyed. A guideline for temporal data model is then critically compared to TODM based on the guidelines presented.

Ariav, G. and Clifford, J., New Directions For Database Systems, Ablex Publishing Corporation, Chap. 12, pp. 168-185, 1986.

Chapter 12 of the book provides a broad overview of temporal data management. Easy to read and understand, the novice reader can use this as an introduction into the topic. The theory behind time-embedded database are examined. A survey of implemented time-embedded database is used to guide the reader. Applications and new issues of time models and systems are presented.

Ariav, G., Clifford, J., Jarke, M., Panel on Time and Databases, Proceedings of ACM SIGMOD International Conference on Management of Data, v. 13, pp. 243-245, May 1983.

The panel discusses and categorizes time-embedded databases into four areas: time and database theory, the implementation of historical database systems, the user interface to historical database, and application of historical database. Research of time in databases follow two trends. The first records history data as it changes over time. The second maintains a transition from database state to the next.

Ariav, G. and Morgan, H., MDM: Handling The Time Dimension in Generalized DBMS, Department of Decision Sciences, The Wharton School, University of Pennsylvania, pp. 1-34, May 1981.

The Dynamic Alerting Transaction Analysis (DATA) model system is presented. Originally conceived, designed and implemented, the authors discuss the development and lessons learned from DATA. The DATA system is a relational model and can be classified as a static rollback database. The DDL and DML are fully described.

Ben-Zvi, J., The Time Relational Model, PH.D. Thesis, Department of Computer Science, University of California at Los Angeles, 1987.

The Time Relational Model is an extension of the relational model with temporal data viewed as a three-dimensional structure.

Bolour, A., Anderson, T., Dekeyser, L., and Wong, H. K. T., The Role of Time in Information Processing: A Survey, ACM SIGMOD RECORD, v. 12, pp. 28-42, Spring 1982.

This survey is mainly concerned with the role of time in the area of artificial intelligence, logic and linguistic. It contains roughly 70 pre-1982 bibliographies and about half are reviewed in depth. The bibliographies are in chronological order dating from the 1960's.

Breutmann, B., Falkenberg, E., Mauer, R., CSL: A language for Defining Conceptual Schemas, Data Base Architect, North-Holland Publishing Company, pp. 237-256, 1979.

CSL is a high level data definition language used for defining conceptual schemas. Conceptual Schemas define the universe of discourse for a database. A time handling facility is introduced which requires an extension of CSL constructs.

Castilho, J., Casanova, M., Furtado, A., A Temporal Framework For Database Specification, Proceeding of the Eighth International Conference on Very Large Data Bases, pp. 280-291, September 1982.

The paper addresses static constraints and transition constraints. Static constraints determine what data is to to be stored. Transition constraints determine how the data can be updated. A multi-level database specification methodology is proposed. The levels determine the specificity of update operations. The first level uses a variant of Temporal

Logic that assumes a database does not have any built-in update operations. The second level contains predefined update operations for any update. These levels are expressed through the use of temporal language.

Clifford, J. and Tansel, A., On An Algebra For Historical Relational Databases: Two Views, ACM SIGMOD International Conference on Management of Data, v. 14, pp. 247-265, December 1985.

Two independent works by the respective authors were combined and submitted to the conference. The papers are concern with the semantics involved to support temporal requirement. This leads to two issues that must be addressed. Extending the relational algebra and the use of attribute versioning. Part I by Clifford is an extension of previous work in the area of extended relational database model. Clifford discusses the semantic issues involved when the time element is added. Examples are given to demonstrate the subtle issues caused by time. Part II by Tansel is a formal presentation of the extended relational algebra, that answers some of the issues raised by Clifford. Highlights of the algebra are the use of time-stamped attributes and non-first-normal-form relations.

Clifford, J. and Warren, D., Formal Semantics for Time in Databases, ACM Transactions on Database Systems, v. 8, pp. 214-257, June 1983.

This paper deals with incorporating the semantic of time in a historical database. The use of formal logic can contribute to the semantics of the database. In this vain, a form of intensional logic formulated by Richard Montague is introduced as a formalism for understanding temporal semantics. It can be compared to formalism of first order logic for relational database model.

Copeland, G. and Maier, D., Making Smalltalk a Database System, Proceedings of ACM SIGMOD International Conference on Management of Data, v. 14, pp. 316-325, June 1984.

Servio Logic Corporation is developing a computer system to support a set-theoretic data model (STDM) in an object-oriented programming environment. Smalltalk, an object-oriented language, is used to address the issues. One of which is to include the temporal domain. The GemStone data model is a similar approach.

Dadam, P., Lum, V., Werner, H., Integration of Time Versions into a Relational Database System, Proceeding of the Tenth International Conference on Very Large Data Bases, pp. 509-522, August 1984.

51

This paper analyzes design issues associated with implementing time version support into a relational model. Some of the issues raised were physical or object versioning, selection of time-stamps, storage requirements, recovery, concurrency control and temporal anomalies. One of the solutions is to integrate time version support into the database system instead of an 'on top' approach, where the database does not know of any support for the temporal domain. Several versioning strategies are presented and narrowed down to one possible implementation.

Ferg, S., Modelling The Time Dimension In An Entity-Relationship Diagram, The 4th International Conference on Entity-Relationship Approach, pp. 280-286, October 1985.

A technique for representing time in the entity-relationship model was developed in 1984, for the Federal Reserve Board to gather and calculate banking statistics. The model uses states and events, which are time-stamped relationships between entities. Events have a time duration, whereas states do not. Attribute changes were model as a relationship between an entity and a domain. The use of Relationship, Attribute, Keys and Entity, (RAKE) was a formalism to represent the model.

Findler, N., Chen, D., On The Problem Of Time, Retrieval Of Temporal Relations, Causality, And Co-Existence, Second International Joint Conference on Artificial Intelligence, pp. 531-545, September 1971.

The paper discusses the issue of temporal events within a question-answering program. The system should be able to deduct inconsistencies, causal relationships and co-existence.

Gadia, S., Toward A Multihomogeneous Model For A Temporal Database, Proceedings of the International Conference on Data Engineering, pp. 390-397, February 1986.

This paper is an extension of his earlier work on homogeneous model. Essentially going from tuple versioning to attribute versioning. The model is based on temporal element, temporal assignment and temporal expression. An essential element of the model is to minimize the distinction between a logical unit of data and its physical representation. Thus the snapshots are in first normal form and the building blocks for the model. Related works are compared to the model.

Gadia, S. and Vaishnav, A Query Language For A Homogeneous Temporal Database, Proceedings of the ACM Symposium on Principles of Database Systems, pp. 51-56, February 1986.

The Homogeneous Temporal Query Language (HTQUEL) is presented for the homogeneous temporal database. Homogeneous is used in the above term because the temporal domain does not vary from one attribute to another. This notion has been modified to allow each attribute to contain temporal information, in other words, attribute versioning. Examples are used to demonstrate the language. Time units in this model are considered discrete. Two data types are employed: temporal relations and temporal element.

Johnson, R. and Lorentzos, N., Temporal data management, Information Update pp. 5-11, September 1987.

This paper is an abstract of their technical report and proposes an extension to relational algebra in support of temporal data. A series of examples are implemented in Ingres. A solution to the problem of functional dependencies are given. The characteristic of other models and their approaches are included.

Jones, S. and Mason, P. J., Handling The Time Dimension In A Data Base, Proceedings of the International Conference on Databases, pp. 65-83, July 1980.

The LEGOL 2.0 data model is a relational specification language designed to handle the time dimension. Illustrations and examples are provided to demonstrate the capabilities of the model. There are two attributes defined with every tuple (start,end). These attributes are user-defined and therefore do not represent the time element uniquely in the database.

Jones, S., Mason, P., Stamper, R., Legol 2.0: A Relational Specification Language For Complex Rules, Information Systems, v. 4, pp. 293-305, 1979.

The Legol language is design for writing rules with explicit time handling capabilities. The paper describes the language.

Klopprogge, M., Term: An Approach to Include the Time Dimension in the Entity-Relationship Model, Entity-Relationship Approach to Information Modeling and Analysis, North Holland, P.P. Chen (ed), pp. 473-509, 1983.

A thorough discussion of extending the entity-relationship model to include the temporal domain is investigated. An emphasis is placed on the handling of incomplete or correction of erroneous data in the schema. The model developed Time-extended entity relationship model (Term), include data definition and data manipulation. The language is based on PASCAL.

Kung, C., On Verification of Database Temporal Constraints, Proceedings of ACM-SIGMOD 1985 International Conference on Management of Data, v. 14, pp. 169-179, December 1985.

A temporal framework for database specification and verification are presented. The specifications are: static constraints, temporal constraints and operation descriptions. The method is developed to check consistency, to ensure the operations are permissible and the sequence of operation are within the temporal constraints.

Lum, V., Dadam, R., Erbe, R., Guenauer, J., Pistor, P., Walch, G., Werner, H., Woodfill, J., Design Of An Integrated DBMS To Support Advanced Application, Proceeding of the Conference on Foundation Of Data Organization, pp. 31-49, May 1985.

This paper identifies three new requirements for databases: support for both normalized and non-normalized models directly at the system interface level, support for text processing and support for the temporal domain. It is argued that building these functions on top of existing DBMS are inefficient. A better way is to design a system to satisfy the new requirements. A description of the DBMS to satisfy the requirements are presented. Other issues of the design are discussed.

Lum, V., Dadam, R., Erbe, R., Guenauer, J., Pistor, P., Walch, G., Werner, H., and Woodfill, J., Designing DBMS Support for the Temporal Dimension, Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 115-130, June 1984.

This paper indicates the difficulties associated with implementing temporal aspect on top of existing database systems. It proposes an integrated approach to directly support the time element. The implementation chosen for temporal data constructs is the chaining of tuples in reverse time order. Other issues such as indexing, storage efficiency, and access methods are presented.

Martin, N., Navathe, S. and Ahmed, R., Dealing with Temporal Schema Anomalies in History Database, Proceeding of the 13th Very Large Data Bases Conference, pp. 177-184, 1987.

This paper describes the problem of temporal anomalies, when temporal schemas and non-temporal schemas coexist in the same environment. Some of the causes of temporal anomalies are identified. The introduction of Temporal Relational Model (TRM) allows both time varying relations and non-time varying relations to be in the same database. Schema Temporal Logic (STL), through the use of modal temporal logic is a suggested solution. Modal logic, partitions the relations and defines a clear interaction between them.

Mays, E., A Temporal Logic for Reasoning About Changing Data Bases in the Context of Natural Language Questions-Answering, Expert Database Systems Proceedings From the First International Workshop, pp. 559-578, 1986.

A modal temporal logic is presented in relation to changes in the database affecting the natural language question-answering system. It should be able to handle a case like this:
U Is New York less than 50 miles from Philadelphia?
S No, shall I let you know when it is?
The temporal logic is defined and demonstrated.

McKenzie, E., Bibliography: Temporal Databases, ACM SIGMOD RECORD, v. 15, pp. 40-52, December 1986.

This article is a comprehensive survey of over 80 bibliographies from 1982-1986, categorized according to the taxonomy developed by Snodgrass and Ahn, by four topics: rollback database, historical database, temporal database, and other time-related research. This work follows that of Bolour.

McKenzie, E. and Snodgrass, R., Extending the Relational Algebra to Support Transaction Time, Proceedings of Association for Computing Machinery Special Interest Group On Management of Data, pp. 467-478, May 1987.

The paper is an extension of previous work. Conventional relational algebra is extended to support valid time, yielding a temporal algebraic language. One of the benefits permit the mapping from the algebraic operations to update operations can be proven correct. Denotational semantics is used to define the language. The language is then described fully, to include the syntax, the semantic domain and the semantic function.

Rotem, D. and Segev, A., Physical Organization of Temporal Data, Proceedings Third International Conference on Data Engineering, pp. 547-553, February 1987.

A multidimensional partitioning of the file in a time-embedded database is proposed. Two algorithms, symmetric and asymmetric, are presented. Experimental results show the algorithms are better than grid-type partitioning.

Schwartz, R., Melliar-Smith, P., Vogt, F., An Interval Logic for Higher-Level Temporal Reasoning, Proceedings of the Second Annual Symposium on Principles of Distributed Computing, pp. 173-185, August 1983.

An interval logic is introduced to represent temporal relationships. The specification and verification methods are based on the interval concept. The language and the logic is presented with examples and illustrations.

Segev, A. and Shoshani, A., Logical Modeling of Temporal Data, Proceedings of Association for Computing Machinery Special Interest Group on Management of Data, pp. 454-466, May 1987.

This paper is an extension of previous works and concentrates on precise specification of time sequences.

Sernadas, A., Temporal Aspects Of Logical Procedure Definition, Information Systems, v. 5, pp. 167-187, 1980.

The paper is concern with adding the time element in a message-oriented relational model to achieve memory independence. Memory independence is recalling the past without the current system being involve in representing the information. The analysis indicates the use of temporal databases to achieve memory independence. A special modal tense logic is describe to facilitate the use of temporal database.

Shoshani, A. and Kawagoe, K., Temporal Data Management, Proceeding of the Twelfth International Conference on Very Large Data Bases, pp. 79-88, August 1986.

The concept of time sequence and time sequence array are introduced. Properties and operations of time sequence are discussed. Design issues, such as access patterns that affect the physical organization are presented.

Snodgrass, R., The Temporal Query Language TQUEL, ACM Transactions on Database System, v. 12, pp. 247-298, June 1987.

TQUEL is designed to be a minimum extension to QUEL, both semantically and syntactically. The article covers the syntax and semantics of TQUEL. A comparison of TUQEL to other query languages supporting the time element is included. The criteria for the comparison are defined and explained. A tuple relational calculus semantic for TQUEL is provided.

Snodgrass, R., Research Concerning Time in Database Project Summaries, ACM SIGMOD RECORD, v. 15, pp. 19-39, December 1986.

This paper is an attempt to determine the current research areas of time in databases. A letter was sent in July of 1986 to 38 researchers active in time related database research. The results are a compilation of 22 responses. Addresses, mailbox address and telephone numbers are provided though not verified. The article provides a source of current active researchers in temporal database and their interest.

Snodgrass, R. and Ahn, I., A Taxonomy of Time, Proceedings of the International Conference on Management of Data, ACM SIGMOD, v. 14, pp. 236-246, December 1985.

The paper defines three types of time: transaction time, valid time, user-defined time and four types of databases: static database, static rollback database, historical database, temporal database. An example is presented for each type of database. Included is a survey of various time models.

Svobodova, L., A Reliable Object-Oriented Data Repository For A Distributed Computer System, Proceeding of 8th Symposium on Operating System Principles, pp. 47-58, December 1981.

The repository is part of a distributed data storage system and contains a stable append-only storage called Version Storage (VS). The repository provides large reliable long term storage of history objects. New techniques for data storage and organization are presented.

Tansel, A., Adding Time Dimension To Relational Model And Extending Relational Algebra, Information System, v. 11, pp. 343-355, 1986.

Extending the relational model and relational algebra to support new time oriented operations are presented. The model is the first to specifically propose adding time at the attribute level versus tuple level. The use of relational algebra is used extensively. New research in the area of graphical query language and prototype implementation in support of the time domain are envisioned.

Urban, S. and Delcambre, L., An Analysis of the Structural, Dynamic And Temporal Aspects of Semantic Data Model, IEEE, pp. 383-389, 1986.

This paper addresses the structural, dynamic and temporal aspect of object-oriented semantic model. Work with temporal features are almost nonexistent and plenty of research is readily available in this area.

LIST OF REFERENCES

1. Bolour, A., Anderson, T. L., Dekeyser, L. J., and Wong, H. K. T., "The Role of Time in Information Processing: A Survey," *ACM SIGMOD RECORD*, v. 12, pp. 28-42, Spring 1982.

2. McKenzie, E., "Bibliography: Temporal Databases," *ACM SIGMOD RECORD*, v. 15, pp. 40-52, December 1986.

3. Snodgrass, R. and Ahn, I., "A Taxonomy of Time," *Proceedings of the International Conference on Management of Data, ACM SIGMOD*, v. 14, pp. 236-246, December 1985.

4. Ahn, I., "Towards An Implementation of Database Management Systems with Temporal Support," *Proceeding of the International Conference on Data Engineering*, pp. 374-381, February 1986.

5. Fujitani, L., "Laser Optical Disk: The Coming Revolution in On-line Storage," *Communication ACM*, v. 27, pp. 546-554, June 1984.

6. Hoagland, A., "Information Storage Technology: A Look at the Future," *IEEE Computers*, v. 18, pp. 60-67, July 1985.

7. Kenville, R. F., "Optical Disk Data Storage," *Computer*, v. 15, pp. 21-26, July 1982.

8. Rothchild, E., *Optical-Memory Media*, v. 8, pp. 86-106.

9. Chi, C. S., "Advances in Computer Mass Storage Technology," *Computer*, v. 15, pp. 60-74, May 1982.

10. Feudo, C. V., *Modern Hardware Technologies and Software Techniques for Online Database Storage and Access,* M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1985.

11. Johnson, R. and Loretzos, W., "Temporal Data Management," *Information Update Database Technology*, pp. 5-11, September 1987.

12. Ariav, G. and Clifford, J., *New Directions For Database Systems,* pp. 168-185, Ablex Publishing Corporation, 1986.

13. Ferguson, M., *Modern Database-System Design for the Tomahawk Common Weapon Control System (CWCS),* M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1984.

14. Allen, J. F., "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, v. 26, pp. 832-843, November 1983.

15. Segev, A. and Shoshani, A., "Logical Modeling of Temporal Data," *Proceedings of Association for Computing Machinery Special Interest Group on Management of Data*, pp. 454-466, May 1987.

16. Shoshani, A. and Kawagoe, K., "Temporal Data Management," *Proceeding of the 12th International Conference on Very Large Database*, pp. 79-88, August 1986.

17. Rotem, D. and Segev, A., "Physical Organization of Temporal Data," *Third International Conference on Data Engineering*, pp. 547-553, February 1987.

18. Lum, V., Dadam, R., Erbe, R., Gruenauer, J., Pistor, P., Walch, G., Werner, H., and Woodfill, J., "Designing Database Management System Support for the Temporal Dimension," *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 115-130, 1986.

19. Korth, H. and Silberschatz, A., *Database Systems Concepts,* McGraw-Hill, 1986.

20. Wiederhold, G., *Database Design,* McGraw-Hill, 1977.

21. Snodgrass, R., "The Temporal Query Language Tquel," *ACM Transactions on Database Systems*, v. 12, pp. 247-298, June 1987.

22. Clifford, G. and Warren, D., "Formal Semantics for Time in Databases," *ACM Transaction on Database Systems*, v. 8, pp. 214-257, June 1983.

23. Martin, N., Navathe, S., and Ahmed, R., "Dealing with Temporal Schema Anomalies in History Database," *Proceeding of the 13th Very Large Data Bases Conference*, pp. 177-184, 1987.

24. Hsiao, D., *Modern Database System Architecture,* (unpublished), Naval Postgraduate School, Monterey, California 93943.

25. Tang, C., "A Temporal Logic Language to represent Data, Knowledge, Algorithm and Specification in a Uniform Framework," *The International Pre-VLDB' 86 Symposium*, pp. 278-291, August 1986.

26. Mays, E. , "A Temporal Logic for Reasoning About Changing Data Bases in the Context of Natural Language Questions-Answering," *Expert Database Systems Proceedings From the First International Workshop*, pp. 559-578, 1986.

27. Sernadas, A., *Temporal Aspects Of Logical Procedure Definition, Information Systems*, v. 5, pp. 167-187, 1980.

28. Schwartz, R., Melliar-Smith, P., and Vogt, F., "An Interval Logic for Higher Level Temporal Reasoning," *Proceedings of the Second Annual Symposium on Principles of Distributed Computing*, pp. 173-185, August 1983.

29. Tansel, A., *Adding Time Dimension To Relational Model And Extending Relational Algebra* , v. 11, pp. 343-355, 1986.

30. McKenzie, E. and Snodgrass, R., "Extending the Relational Algebra to Support Transaction Time," *Proceedings of Association for Computing Machinery Special Interest Group On Management of Data*, pp. 467-478, May 1987.

31. Ben-Zvi, J., *Department of Computer Science*, Ph.D. Dissertation, University of California at Los Angeles, 1987.

32. Breutmann, B., Falkenberg, E., and Mauer, R., *CSL: A language for Defining Conceptual Schemas, Data Base Architect,* North-Holland Publishing Company, 1979.

33. Jones, S. and Mason, P., "Handling The Time Dimension In A Data Base," *Proceedings of the International Conference on Databases*, pp. 65-83, July 1980.

34. Klopprogge, M., ''Term: An Approach to Include the Time Dimension in the Entity-Relationship Model,'' *Entity-Relationship Approach to Information Modeling and Analysis*, pp. 473-509, North Holland, 1983 .

INITIAL DISTRIBUTION LIST

No. Copies

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chief of Naval Operations Director, Information Systems (OP-945) Navy Department Washington, D.C. 20350-2000	1
4.	Commandant of the Marine Corps Code TE 06 Headquarters, U.S. Marine Corps Washington, D.C. 20360-0001	1
5.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	2
6.	Curriculum Officer, Code 37 Computer Technology Naval Postgraduate School Monterey, California 93943-5000	1
7.	Professor David K. Hsiao, Code 52Hq Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	2
8.	Professor Thomas C. Wu, Code 52Wq Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	2

9.   Donald D. Hom   2
             505 Norwood Street
             East Orange, New Jersey 07018