THEORETICAL FRAMEWORK

OF

TEMPORAL DATABASES

by

Lam Wing Hee

A thesis submitted to the

Department of Computer Science

The Chinese University of Hong Kong

in partial fulfillment of the requirements

for the degree of

Master of Philosophy

June 1991

Theoretical Framework of Temporal Databases

Lam Wing Hee

Abstract

Temporal database systems are database systems concerning with the storage and retrieval of historical data. As the three classical data models do not possess generic capabilities to handle historical data, various temporal data models have been defined to provide the bases for temporal database management systems.

This thesis is an attempt to give a theoretical framework for the study of temporal databases under a specific data model, the Temporal Relational Model, which is based on the relational model with timestamping at the tuple level. The data model will be defined in the thesis, together with a formal query language, the Temporal Relational Algebra, which support operations analogous to the join in the relational model.

Properties of temporal databases will be analyzed by considering various data dependencies in the data model. Temporal counterparts to functional and multivalued dependencies will be defined. The new multivalued dependency is found to be a necessary and sufficient condition for the lossless decomposition of a temporal relation into two. A new data dependency, the asynchronous dependency (AD), will be introduced. The presence of AD in a temporal relation is shown to cause update anomalies, and a new normal form is defined to show the condition under which no update anomaly is caused by AD.

To those friends and colleagues

known collectively as

the M Club

Contents

List of Figures

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. C. Yau, for his invaluable guidance, advice, and support and who has showed the greatest patience. I would also like to thank Professor T.C. Chen and Dr. Peter L.M. Liu for their useful comments and suggestions and for serving as thesis committee members. Thanks are also owed to Miss Grace S.W. Chat for all the valuable discussions and advice.

Last but not least, I would like to acknowledge the support of the Department of Computer Science of the Chinese University of Hong Kong which provides a stimulating environment for the research project.

Chapter 1

Introduction

Time is a unique concept that we have to consider as computer scientists. This is not a surprising fact as time is also an intricate concept in a wide range of disciplines - philosophy, psychology, linguistics - as well as in our everyday life. In computer science, information systems have the problem of representing and retrieving current as well as historical information. Researchers in artificial intelligence need to have models of problem solving that can cope with a changing world, and that can reason about temporal relationships of all relevant entities and events. In natural language processing, we have to extract and capture tense information in sentences. This research project focuses on the issue of incorporating the time concept in database design, which is regarded as one of the important problems in information systems.

1.1 Historical Data and Temporal Databases

Information systems deal with the storage and retrieval of data and information about real-world objects. An important aspect of these data is their temporal properties. For example, we may need to know the medical history of a patient, the price history of a stock, etc.

The three classical data models, the hierarchical, the network and the relational model, do not provide facilities for the storage and retrieval of historical data. If an application is required to handle historical data, the designers must incorporate the temporal aspects

of the data into the conceptual schema themselves, and the database management system (DBMS) cannot recognize that historical data is being manipulated or assist in the handling of the historical data itself. Actually there are general functions which can handle temporal data and can be incorporated into the DBMS as a systematic tool.

The three classical data models' lack of facilities for temporal data is reflected in their semantics of data manipulation. When data about an entity is modified or deleted, the old data is lost to subsequent retrieval, only the most recent data is available. That is, if a system want to retain an old piece data while storing a new piece, it should not use the DBMS modification command, it should add the new piece of data and distinguish the old and new data by timestamping the two records.

Temporal databases work on the same principle, they adopt a nondeletion policy [Copeland 1982] and timestamp each piece of data it stored. The difference with DBMSs of classical data models is that for temporal DBMS the nondeletion policy and the timestamping is handled by the DBMS directly, without the need for users' attention.

There are many examples of database applications with a nondeletion policy in the real-world. Human memory is found to have no deletion mechanism, although memories will decay with time. A fundamental principle of accounting theory is that no journal entry or ledger posting deletes any existing information [Copeland 1982]. In general, most accounting, financial and legal databases should adopt the nondeletion policy, although we have the option to put the historical data on paper, on microfilm or online.

1.2 Valid Time and Transaction Time

For the purpose of timestamping data we have to distinguish between two concepts, and following the terminology in [Snodgrass and Ahn 1985], we will call them the *valid time* and the *transaction time* respectively. Briefly, the valid time period for a proposition is the period the proposition is true in the real world, while the transaction time period is the period the proposition is recognized as true by the database system. For example, in the statement "The price of crude oil on 2/2/88 is $7.56 according to the record of 1/1/89", "2/2/88" indicates an instant in the valid time period while "1/1/89" indicates an instant in the transaction time period.

These two concepts distinguishes several types of databases according to their abilities to support valid time or transaction time [Snodgrass and Ahn 1985, 1986].

1.2.1 Snapshot Databases

Snapshot databases do not support the valid time or the transaction time. They model the real world by a snapshot of the world at a particular moment in time. A state of a database is its current data content. Updating the databases writes a new state to the database and past states are lost completely. That is, it is not possible to retrieve past information from the database.

In the relational model - which can be seen as a model of snapshot databases, a database is a collection of relations, and each relation can be represented by a two dimensional

table, as shown in fig.1.1. As changes are made to the database, updates are written to the tables.

Conventional databases are snapshot databases. Data manipulation to such databases can be performed using conventional data manipulation languages, such as SQL, which does not include operations to retrieve or update past information.

Fig.1.1. A table in a snapshot database

Emp-no	Salary	Rank
E10	5000	SA
E11	4000	AP
E12	6000	SSA
E20	5000	SA

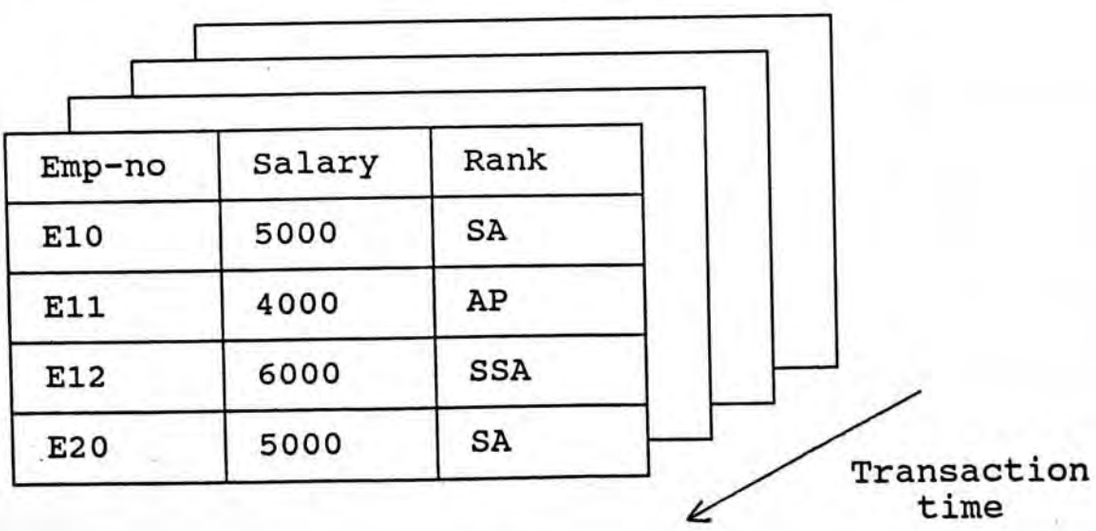
1.2.2 Rollback Databases

A rollback database incorporates past information by storing all past states of the database, indexed by time, as the data in the database is updated. The time index is a *transaction time* timestamp - the time instance when the information is stored in the database. Rollback database thus can handle queries such as "what is stored in the database on 1/1/89 about the price of crude oil?".

A relation in a rollback database can be visualized as a three dimensional block of data with one of the axis representing the transaction time continuum, as in fig.1.2. By moving along the transaction time axis and taking a slice of the block, i.e. by rolling back, one retrieves a snapshot of the relation as of some particular moment in the past. In this way, historical information can be retrieved from a rollback database. Data manipulation languages for rollback databases must therefore have additional operations for rolling back the database to past states.

In contrast to the retrieval, updates to the database can be made only to the most recent state. The new information creates a new state of the database which is appended to the front of the past states, creating the three dimensional block of information. The old states are still stored. We can see that rollback database records the histories of database activities, rather than the histories of the real world. This approach limits the representation of information - there is no way to record retroactive or postactive changes, nor to correct errors in past states of the database.

Fig.1.2. A table in a rollback database



1.2.3 Historical Databases

Rollback databases store the entire history of the database. On the other hand, historical databases store the history of the entities.

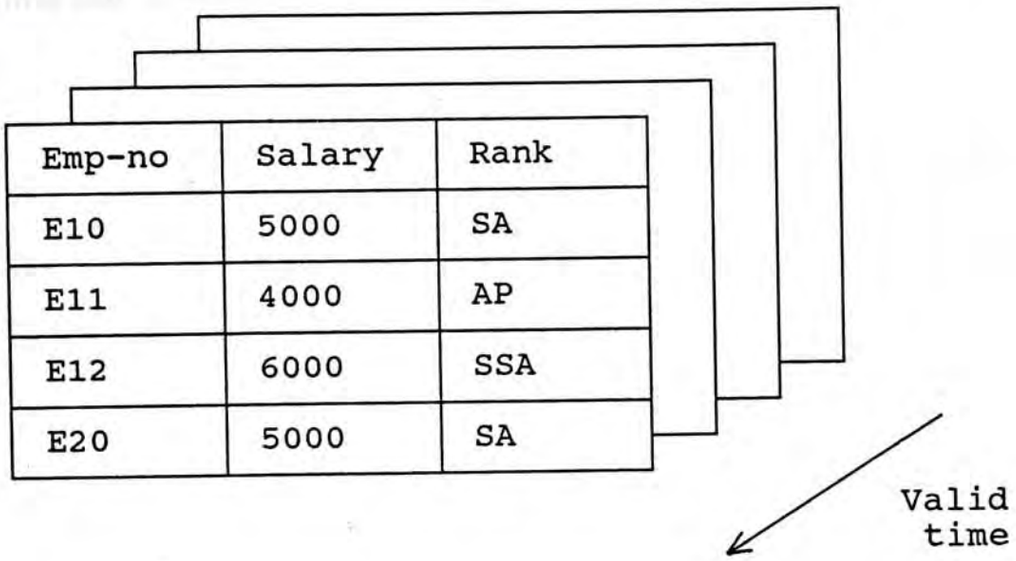
In a historical database, the time supported by the database represents the valid time of the data - i.e. the time that the stored data in the tuple is valid in the real world. Queries such as "what is the price of crude oil on 2/2/88?" thus can be handled.

A relation in a historical database can also be visualized as a three dimensional block of data, as shown in fig.1.3. The difference between historical and rollback databases is that the time axis now represent the valid time, rather than transaction time.

Historical databases allow changes, e.g. error correction, to a data in any particular valid time moment. If an error is discovered about past data, they can be corrected by updating with the correct data. However, in this case the past error is lost from the database forever.

To support the valid time dimension, data manipulation languages for historical databases then should have facilities for specifying the valid time of the data to be retrieved or to be updated.

Fig.1.3. A table in a historical database



1.2.4 Temporal Databases

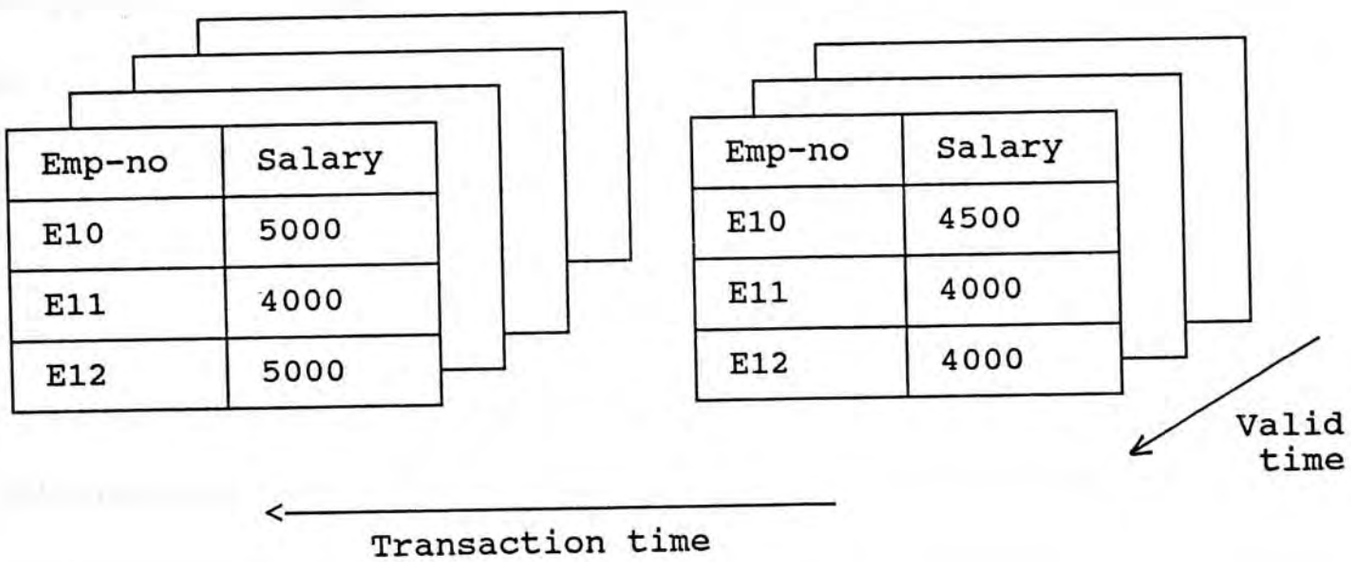
Benefits of both rollback databases and historical databases can be obtained by supporting both transaction time and valid time in the database - information in the database is time-stamped with both the transaction time and valid time. Databases using this approach is termed a temporal database. Temporal databases can answer query such as "what is stored in the database on 1/1/89 about the price of crude oil on 2/2/88?".

An additional benefit of temporal databases is the ability to represent proactive changes, i.e. an update to the database can be registered well before the change happens in the real world, and retroactive changes, an update registered after the change has happened in the real world.

A relation in a temporal database is visualized as a four dimension block, as shown in fig.1.4. Two time axis is present - the transaction time axis and the valid time axis.

As both valid and transaction time are supported, data manipulation languages for temporal database require facilities for rollback as well as for specifying the valid time of the data.

Fig.1.4. A table in a temporal database



1.3 Literature Review

In this section, we will present an overview of current status of research in temporal databases. We will survey data models, query languages, and logical design issues discussed in the literature. Topics in physical data structures and implementation issues will not be discussed, as they are outside the scope of discussion for the thesis.

1.3.1 Data Models

Temporal (or historical) data models in most published papers can be divided into four categories:

a) Tuple timestamping

Data models with tuple timestamping extend the relational model by a nondeletion policy and by timestamping each tuple - historical data models ([Clifford and Warren 1983], [Navathe and Ahmed 1987a, 1987b], [Lorentzos and Johnson 1988], [Sadra 1990]) use timestamps for valid time and temporal data models ([Snodgrass 1986], [Snodgrass and Ahn 1987], [Chat 1990]) use timestamps for both valid and transaction time. Fig.1.5 shows a relation with tuple timestamping.

Since data models with tuple timestamping is based on the relational model, temporal DBMS can easily be prototyped as a front end system to a relational DBMS. Moreover, temporal query languages for these data models resemble relational query language, making them easy to learn and use.

Fig.1.5 Relation with tuple timestamping

EMP	SAL	RANK	Time1	Time2
1234	6000	P1	1/1/88	31/12/88
1234	7000	P1	1/1/89	31/12/89
1234	8000	AP	1/1/90	NOW
2345	5000	AP	23/4/90	NOW

b) Attribute timestamping

Data models using attribute timestamping are based on the nested relational model (or called the non-first normal form relational model) [Abiteboul and Bidiot 1986, Kambayashi et al. 1983, Roth and Korth 1987], with a nondeletion policy and with timestamps for every attribute value. A relation with attribute timestamping is shown in fig.1.6.

As no commercial DBMS is available for the nested relational model, temporal DBMS based on attribute timestamping cannot be easily prototyped. Query languages for data model with attribute timestamping will also inherit the quite complex structures of nested relational query languages.

Papers advancing the attribute timestamping approach include [Clifford and Tansel 1985], [Tansel 1986], [Tansel and Garnett 1989].

Fig.1.6 Relation with attribute timestamping

EMP	SAL	RANK
1234 <1/1/88,->	6000 <1/1/88, 31/12/88> 7000 <1/1/89, 31/12/89> 8000 <1/1/90,->	P1 <1/1/88, 31/12/89> AP <1/1/90,->
2345 <23/4/90,->	5000 <23/4/90,->	AP <23/4/90,->

c) Time cube approach

The time cube approach ([Ariav 1986, Segev and Shoshani 1987]) does not model temporal data into the two-dimensional data models - the relational model or the nested relational model. They are two-dimensional in the sense that a value or a set of value is identified by specifying the pair (entity, attribute). Time cube approach views the data as having three or more dimensions, where a value is given by specifying all these: (entity, attribute, time1, time2, ...). A schematic diagram showing a table in a time cube data model is shown in fig.1.7.

This approach is quite neat from a theoretical point of view. But the simplicity of the model complicates the query languages. Query languages for data models of this approach are often difficult to use.

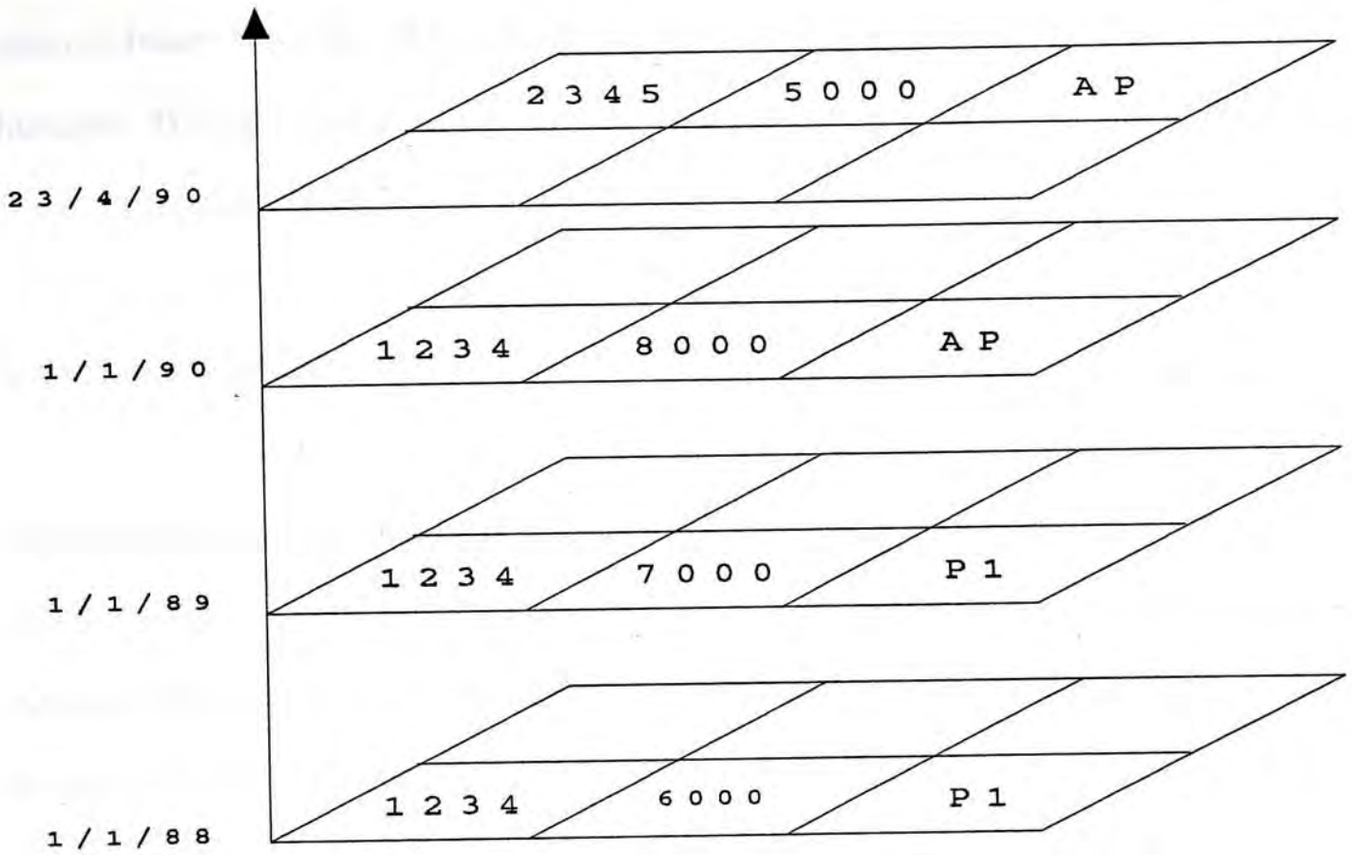
d) Other approaches

Other temporal data models are mainly based on semantic data models [Peckman and Maryanski 1988], examples are [Klopprogge 1981], [Klopprogge and Lockemann 1983], both are extensions to the entity-relationship model [Chen 1976]. Some are based on the related objected-oriented data models, like [Copeland and Maier 1984].

1.3.2 Query Languages

Data models of tuple timestamping, attribute timestamping and time cube approach are all directly or indirectly based on the relational model. Therefore query languages for these data models can be classified in similar way as for relational model languages:

Fig.1.7 Schematic diagram for time cube approach to temporal data model



a) Algebraic language.

A number of temporal query languages are based on the relational algebra, they include [Tansel 1986], [Segev and Shoshani 1987], [Navathe and Ahmed 1987a], [Lorentzos and Johnson 1988], [Tansel and Garnett 1989], [Sadra 1990].

b) Calculus-based language

Few calculus-based language has been designed. One for a temporal data model based on tuple timestamping is TQuel [Snodgrass 1987].

c) SQL-based language

For our purpose we will treat SQL-based languages separate from algebraic language or calculus-based language. Those historical or temporal query languages based on SQL includes TOSQL [Ariav 1986], TSQL [Navathe and Ahmed 1987b], TempSQL [Chat 1990], and HSQL [Sadra 1990].

1.3.3 Logical Design

Few have been written on logical design method for a particular temporal data model, and few on data dependencies in temporal databases. A good attempt is [Navathe and Ahmed 1987a], which includes a discussion of a "temporal dependency" and "temporal normal form". They are in the same spirit as the asynchronous dependency to be introduced in chapter 5, though Navathe is treating the issue in historical databases, instead of our concern in temporal databases.

Chapter 2

The Temporal Relational Data Model

In this chapter, we will first define the basis of our analysis -i.e. the data model we use. A data model is an abstract description of the structure of the conceptual level of the database. Ullman [Ullman 1982 p.18] has a good characterization of what we usually meant by a data model: a data model consists of two elements:

i. a mathematical notation for expressing data and relationships between them

ii. operations on the data that serve to express queries and other manipulations of the data.

Therefore, in this chapter, we will define the notations for expressing the data and relationships. We will also define the structural elements of the model, namely, attributes, schemes, tuples, relations and keys.

As different schemes of operations on the same structural elements can be defined, we will define one of the possible schemes of operations - the *temporal relational algebra*, in the next chapter.

2.1 The Temporal Relational Data Model - Informal Description

In the relational model, all data are organized into relations which can be visualized as a set of tables. In the temporal relational model, data are maintained as *temporal relations*, which can also be look upon as a set of tables. But each of these tables has four additional columns in it. The column headings are Vs, Ve, Ts and Te, which stand

for *valid time start*, *valid time end*, *transaction time start* and *transaction time end* respectively.

An example of temporal relation represented as a table is shown in fig.2.1. Vs, Ve, Ts and Te are the time attributes, the values of which is explicitly stored for each tuples. "-" represents that the transaction time for a tuple has not yet ended.

Taking the last row in the figure as an example. It may have the meaning that the employee with the number 2345 have a monthly salary of $5000 and the rank of an analyst/programmer, and that this information is valid from time 1 (in a certain time scale) to now, and that this information is considered up-to-date in the database from time 1 onwards.

Fig.2.1 An example temporal relation

EMP	SAL	RANK	Vs	Ve	Ts	Te
1234	6000	P1	1	NOW	1	7
1234	6000	P1	1	6	7	-
1234	7000	AP	6	NOW	7	-
2345	5000	AP	1	NOW	1	-

2.2 The Temporal Relational Data Model - Formal Description

Having discussed temporal relations informally, we will construct the formal definitions of the structural elements of the data model below.

2.2.1 Valid and Transaction Time Intervals

For temporal relations, there are four special time attributes Vs, Ve, Ts and Te. They can identify two time intervals: the *valid time interval* [Vs, Ve) and the *transaction time interval* [Ts, Te). We will use the symbols V and T to represent the intervals of [Vs, Ve) and [Ts, Te) respectively. We assume the domains of V and T be DOM(V) and DOM(T) respectively which are sets of time intervals [a, b) with $a \leq b$.

2.2.2 Attributes, Tuples and Temporal Relations

In the definition of tuples, a non-time attribute is defined as a symbol taken from a certain given finite set U. We would use the letters A, B, C, D to denote attributes, and use R, X, Y, Z to denote sets of attributes. We usually do not distinguish between an attribute A and the attribute set {A}. The union of X and Y is denoted by XY. For example, the set {A, B, D} will be denoted by ABD.

Underlying each attribute A is a domain of values, denoted by DOM(A), out of which the data values are taken. The set $\bigcup_{A \in U} DOM(A)$ is denoted by Dom.

A *tuple* on an attribute set X is then defined as a mapping $u: X \cup V \cup T \rightarrow Dom \cup DOM(V) \cup DOM(T)$, such that $u(A) \in DOM(A)$ for all $A \in X$, $u(V) \in DOM(V)$, and $u(T) \in DOM(T)$. A *temporal relation* on X is a set of tuples on

X. Finally, if I is a relation of tuples on X, then X is called the *relation scheme* or just *scheme*, of I.

2.3 What is a Key in Temporal Relations?

In the relational model, an attribute or a set of attributes whose values can uniquely identify an individual tuple in a relation is called a key for that relation. Since each tuple represents an entity, a key serve as an identifier to individual entity in an entity set.

Due to the representation method in the temporal relation model, the relationship between tuple and entity has changed. A tuple in a temporal relation now only describes a particular period in the entire life of an entity. A number of tuples collectively is now required to describe the whole of the "entity life".

We now want to define the notion of a key as an identifier to an entity life, without the burden of having to specified the valid time and the transaction time intervals of the entity's existence. For example, we will intuitively call EMP the key of the temporal relation in fig.2.1, since a value for EMP such as 1234 identifies the set of tuples describing the employment history of a single employee with employee number 1234.

Lets take a look at what if we take the relational model approach to the notion of keys, instead of the above mentioned. Assume that a key of a temporal relation is defined as the identifier to a tuple in the relation. Consider the temporal relation in fig.2.2. We may decide that {EMP, Vs, Ve, Ts, Te} is a key for the relation. But note that the two

tuples just denote that employee 7890 has two different salary at the same time when V is, say time 5. While we want the key to identify a single tuple, we are missing the possibility that duplicate or contradicting descriptions are held about the same entity. This is the reason why the concept of key in the temporal relational data model is not an identifier to a tuple, but is instead an identifier to an entity life.

Fig.2.2 A problem with the relational approach to key in temporal relational model

EMP	SAL	RANK	Vs	Ve	Ts	Te
7890	6000	P1	1	10	1	10
7890	7000	P1	2	11	1	10

According to the previous discussion, a key K of a temporal relation I on scheme X is a subset of X with the following property: in a snapshot of the database at a given combination of valid and transaction time instance, two tuples agreeing on the value of the key must have equal values for all attributes, except possibly on the time attributes.

We define a superkey K for a temporal relation I on scheme X as a subset of X such that for all u, v in I, if u[V]∩v[V] and u[T]∩v[V] are both non-empty and u[K]=v[K], we have u[I]=v[I].

Note that we denote by u[X] the restriction of the tuple (a mapping) u to X, taken the set-theoretic sense of a restriction mapping. Informally, u[X] is the "sub-tuple" whose scheme is restricted to X.

A key K of a temporal relation I is then a superkey of I such that no proper subset of K is a superkey. For example, EMP is a key for the temporal relation in fig.2.1, while {EMP, SAL} and {EMP, RANK} (and others) are superkeys.

Chapter 3

The Temporal Relational Algebra

As stated in the introduction of chapter 2, the definition of a data model should include the data manipulative aspect of the data model.

Actually, different schemes of operations, or *data manipulation languages*, can be defined on the same underlying data structure of a data model. Moreover, the non-query aspects of a data manipulation language are often straightforward, being concerned with the insertion, deletion and modification of tuples [Ullman 1982 p.151]. Query is the part which requires a rich, high level language for its expression. Most query languages can be classified as either an *algebraic query language* - in which queries are expressed by applying operators to data structures, or a *calculus-based query language* - where queries specifies a predicate which the answer of the query should satisfy [Ullman 1982, Maier 1983].

In this chapter a formal algebraic query language over the temporal relational model, called the *temporal relational algebra*, will be proposed.

3.1 Operations in the Temporal Relational Algebra

Temporal relational algebra is basically a collection of operators whose operands are temporal relations. It is based on the *relational algebra* [Ullman 1982, Maier 1983] for the relational data model. All operators in the temporal relational algebra return

temporal relations. In the following sections, the operations provided in the language will be defined in turn.

3.1.1 Union and Set Difference

Union - $I_1 \cup I_2$ and set difference $I_1 - I_2$, apply to temporal relation I_1, I_2 which must have the same relation scheme. The meaning is the same as the set-theoretic meaning of union and set difference: i.e. $I_1 \cup I_2$ is the set of tuples that are in I_1 or in I_2 or both; $I_1 - I_2$ is the set of tuples that are in I_1 but not in I_2. Fig.3.1 shows examples of union and set difference.

3.1.2 Selection

Selection selects a number of tuples from a temporal relation according to a specified criteria P, where P is a predicate created from:

i) operands that are constants or attribute names or Vs, Ve, Ts or Te;

ii) arithmetic comparison operators $<, =, >, \leq, \neq$, and $\geq$, and

iii) logical operators $\wedge$, $\vee$, $\neg$,

following the usual rule for constructing a conditional expression.

The select operator returns tuples that satisfies the predicate P from the temporal relation I operated on:

$$\sigma_P\{I\} = \{u \epsilon I: P(u)\}$$

Examples are shown in fig.3.2.

Fig.3.1 Union and set difference

I_1

EMP	SAL	Vs	Ve	Ts	Te
1234	5000	2	3	2	3
5678	7500	1	7	1	8

I_2

EMP	SAL	Vs	Ve	Ts	Te
1234	6000	4	5	4	5
5678	7500	1	7	1	8
5678	9500	8	10	9	10

$I_1 \cup I_2$

EMP	SAL	Vs	Ve	Ts	Te
1234	5000	2	3	2	3
1234	6000	4	5	4	5
5678	7500	1	7	1	8
5678	9500	8	10	9	10

$I_1 - I_2$

EMP	SAL	Vs	Ve	Ts	Te
1234	5000	2	3	2	3

Fig.3.2 Selections

I

EMP	SAL	Vs	Ve	Ts	Te
1234	5000	2	3	2	3
1234	6000	4	5	4	5
5678	7500	1	7	1	8
5678	9500	8	10	9	10

$\sigma_{EMP=1234}(I)$

EMP	SAL	Vs	Ve	Ts	Te
1234	5000	2	3	2	3
1234	6000	4	5	4	5

$\sigma_{Vs\geq 4 \wedge Ts\geq 4}(I)$

EMP	SAL	Vs	Ve	Ts	Te
1234	6000	4	5	4	5
5678	9500	8	10	9	10

3.1.3 Projection

In the temporal relational model, if I is a relation on X and if Y is a subset of X, we define the projection of I onto Y, written $\pi_Y(I)$, as the set

$$\pi_Y(I) = \{w[Y \cup VT]: w \epsilon I\}$$

Informally, the columns of the attributes in the set Y are selected and copied to the projection, as well as the columns of V and T. We cannot use projection to eliminate one or more of the time attribute columns.

Fig.3.3 Projection

I

EMP	SAL	RANK	Vs	Ve	Ts	Te
1234	5000	AP	2	3	2	3
1234	6000	AP	4	5	4	5
5678	7500	AP	1	7	1	8
5678	9500	MGR	8	10	9	10

$\pi_{\{EMP,SAL\}}(I)$

EMP	SAL	Vs	Ve	Ts	Te
1234	5000	2	3	2	3
1234	6000	4	5	4	5
5678	7500	1	7	1	8
5678	9500	8	10	9	10

3.1.4 Join

Join is a binary operator to combine two temporal relations into one. We will provide two versions of join, the natural join and its generalization, the θ-join. Before the joins are defined, we will describe some design criteria for evaluating the design of these two constructs.

First, the join operation must correspond to some natural combination methods of two relations, that the users wish to use for querying the database.

Second, we may want the join to be commutative and associative. A commutative and associative join will diminish the burden of the users in arranging a chain of joins into a sequence.

3.1.4.1 Natural Join

The natural join combines two temporal relations into one temporal relation. The operation put together tuples from several temporal relations when these tuples agree on values for the attributes that they have in common.

Formally, if I_1 and I_2 are temporal relations on scheme X_1 and X_2 respectively, the join $I_1 * I_2$ as defined as a relation on $X = X_1X_2$ defined by

$I_1 * I_2 = \{$ w is a tuple on X: there exists $u \epsilon I_1$ and $v \epsilon I_2$ such that $w[X_1]=u[X_1]$, $w[X_2]=v[X_2]$, $w[V]=u[V] \cap v[V]$ is non-empty and $w[T]=u[T] \cap v[T]$ is non-empty$\}$

Using the example in fig.3.4, we can see that the time intervals V and T in the only tuple of the joined relation are [3, 7) and [4, 8) respectively. These two intervals is actually the intersections of the V and T values in the tuples of the two original relations, i.e. [1, 7)∩[3, 9) and [1, 8)∩[4, 10) respectively.

The motivation is that when we say that "employee 5678 is associated with a salary of 5000 and with the rank of AP in a certain time period", it should be the case that "employee 5678 have salary 5000 at that period" AND also "his/her rank is AP at that period".

On the other hand, when we say that the information "employee 5678 is associated with a salary of 5000 and the rank of AP" is up-to-date in a certain time period, it should be the case that the information "employee 5678 have salary 5000" is up-to-date at that period AND also the information "the rank of employee 5678 is AP" is up-to-date at that period. This line of reasoning accounts for the above definition.

The commutativity of natural join can be seen from the symmetry of the two operands in the definition, and the associativity can be easily obtained by manipulation.

Fig.3.4 Natural join (example 1)

I_1

EMP	SAL	Vs	Ve	Ts	Te
5678	7500	1	7	1	8

I_2

EMP	RANK	Vs	Ve	Ts	Te
5678	AP	3	9	4	10

$I_1 * I_2$

EMP	SAL	RANK	Vs	Ve	Ts	Te
5678	7500	AP	3	7	4	8

Fig.3.5 Natural join (example 2)

J_1

EMP	SAL	Vs	Ve	Ts	Te
1111	4000	1	3	1	3
1111	5000	3	6	3	6
2222	9500	8	12	5	12

J_2

EMP	RANK	Vs	Ve	Ts	Te
1111	P	2	5	2	5
2222	AP	8	10	9	10
3333	AP	8	10	9	10

$J_1 * J_2$

EMP	SAL	RANK	Vs	Ve	Ts	Te
1111	4000	P	2	3	2	3
1111	5000	P	3	5	3	5
2222	9500	AP	8	10	9	10

3.1.4.2 θ-Join

The θ-join is a generalization of natural join and differs from it in that tuples are combined according to user-specified criteria. Before we define θ-join, we have to introduce some operators on time intervals, called temporal operators [Chat 1990] first:

a) beginof - unary operator, returns the start point of an interval

beginof [s,e) = [s,s)

b) endof - unary operator, returns the end point of an interval

endof [s,e) = [e,e)

c) span - binary operator, returns the smallest interval that include the operands

[s1,e1) span [s2,e2) = [min(s1,s2), max(e1,e2))

The θ-join of I_1 and I_2 (with scheme $X = X_1X_2$) on attributes A and B is written $I1 *_{\theta,\kappa,\lambda} I2$, where θ is a predicate on the non-time attributes in the schemes of I_1 and I_2, κ is a predicate on the interval V of I_1 and V of I_2, and λ is a commutative operator on the interval V of I_1 and V of I_2 constructed from the above temporal operators together with ∩ and ∪, returning a time interval. It is defined as

$I_1 *_{\theta,\kappa,\lambda} I_2 = \{$ w is a tuple on X: there exists $u \in I1$, $v \in I2$ such that $w[X_1]=u[X_1]$, $w[X_2]=v[X_2]$, $w[V]=\lambda(u[V], v[V])$, $w[T]=u[T] \cap v[T]$ is non-empty and θ and κ$\}$

Note the asymmetric treatment for the V-intervals versus the T-intervals, and note that natural join can be expressed as $*_{True,u[V]\cap v[V]\neq\varnothing,u[V]\cap u[V]}$. Fig.3.6 shows an example which

can be interpreted as "who earns more than $4000 as a programmer and when did it happen?".

From the definition above, one can see that whether a θ-join is commutative depends on the commutativity of λ. Therefore λ is restricted to allow only commutative operations.

Fig.3.6 θ-join

I_1

EMP	SAL	Vs	Ve	Ts	Te
1111	4000	1	3	1	3
1111	5000	3	6	3	6
2222	9500	8	12	5	12

I_2

EMP	RANK	Vs	Ve	Ts	Te
1111	P	2	5	2	5
2222	AP	8	10	9	10
3333	AP	8	10	9	10

$I_1 *_{RANK=P \wedge SAL>4000,\ I1[V] \cap I2[V] \neq \emptyset,\ beginof(I1[V] \cap I2[V])} I_2$

EMP	SAL	RANK	Vs	Ve	Ts	Te
1111	5000	P	3	3	3	5

3.2 Temporal Relational Algebra and TempSQL

Due to its simplicity, the temporal relational algebra can be seen as a underlying language for a temporal relational database management system. Other languages over the temporal relational data model can be translated to the algebra. TempSQL [Chat 1990] is a query language based on the relational query language SQL. We will now consider translating the SELECT statement in TempSQL to the temporal relational algebra.

The SELECT statement (restricted to two temporal relations) in TempSQL has the form:

```
SELECT (R_m.A_n)^r
FROM R_1,R_2,
WHERE p
WHEN  q
VALID FROM a to b
AS-OF c
```

where

$R_m.A_n$ represents the n-th attribute of relation R_m,

p is a predicate on the non-time attributes,

q is a predicate on the valid time attributes,

a, b are expressions evaluating to valid times, and

c is an expression evaluating to a transaction time.

The above query expressed in the algebra will take the form:

$$\pi_{(R_m.A_n)_r}(R_1 *_{p,q,[a,b)} R_2)$$

SELECT statements for three or more relations do not translate directly into the algebra. This is due to the fact that θ-joins for three or more temporal relations have not been defined, and it is doubt that if such joins are useful to users at all.

Chapter 4

Classical Data Dependencies in Temporal Relations

In this chapter we will extend the concept of *functional dependency* (FD) and *multivalued dependency* (MVD) to the temporal relational model. Then in the next chapter we will discuss a new type of data dependencies present in this data model. Detailed descriptions of the concepts of FD and MVD in the snapshot relational model can be found in various sources such as [Ullman 1982] or [Maier 1983] and would not be repeated here.

4.1 Functional Dependency in the Temporal Relational Model

Classical functional dependency requires that one attribute set is single-valued with respect to another attribute set. This concept is extended to temporal relational model, by a new constraint called *temporal functional dependency* (TFD). A TFD is a constraint which requires that one attribute set if single-valued with respect to another for every combination of V-value and T-value.

Formally, the TFD $X \rightarrow Y$ is satisfied by a temporal relation I on R if $XY \subseteq R$ and for all tuples u, $v \in I$, if $u[X]=v[X]$ and both $u[V] \cap v[V]$ and $u[T] \cap v[T]$ are not empty, then $u[Y]=v[Y]$. Fig.4.1 illustrate a relation satisfying TFD Course→Teacher.

The existence of TFDs not related to the keys in a temporal relation scheme may cause update anomalies. The details are the same as in snapshot databases and will not be

repeated here. These anomalies can be avoided by designing the database scheme according to a temporal version of the *Boyce-Codd normal form* (BCNF) [Codd 1972a].

The definition of BCNF in the temporal relational context reflects that in the relational model. A temporal relation scheme R is said to be in *temporal BCNF* if whenever TFD X→A holds in R, and A is not subset of X, then X is a key of R.

Fig.4.1 TFD Course→Teacher satisfied by temporal relation

Course	Teacher	Class	Vs	Ve	Ts	Te
Eng	Chan	1	1	10	1	10
Eng	Chan	2	2	10	2	10
Eng	Yang	1	11	12	1	12
Phy	Yang	1	1	10	1	10
Phy	Yang	2	1	5	1	-

4.2 Multivalued Dependency in the Temporal Relational Model

Informally, a *temporal multivalued dependency* (TMVD) is a constraint in a relation scheme which means that some distinct entities are being mixed in the same relation.

Formally, the TMVD X↠Y is satisfied by a temporal relation I on R if XY⊆R and for all tuples u, v ∈ I, if u[X]=v[X] and both u[V]∩v[V] and u[T]∩v[T] are non-empty, then there exists a tuple w ∈ I such that w[X]=u[X]=v[X], w[Y]=u[Y], w[R-XY]=v[R-XY], w[V]⊇u[V]∩v[V] and w[T]⊇u[T]∩v[T]. Fig.4.2 illustrate a relation satisfying MVD Course↠Teacher and Course↠Class.

Fig.4.2 TMVD Course↠Teacher and Course↠Class in a temporal relation

I

Course	Teacher	Class	Vs	Ve	Ts	Te
Eng	Chan	1	1	10	1	10
Eng	Chan	2	2	10	2	10
Eng	Li	1	3	14	4	18
Eng	Li	2	3	15	4	16
Phy	Li	1	1	10	1	10
Phy	Li	2	1	5	1	-

$Snapshot_{V=2;T=2}(I)$

(which satisfies classical MVD Course↠Teacher and Course↠Class)

Course	Teacher	Class
Eng	Chan	1
Eng	Chan	2
Phy	Li	1
Phy	Li	2

4.3 Relationship with Snapshot Data Dependencies

We claim that TFD and TMVD are natural extension of the concepts of FD and MVD to temporal relations. We can see that if a temporal relation satisfies C, where C is either a TFD or TMVD, then every snapshot of the temporal relation satisfies the snapshot version of C. That is, the temporal dependencies reduce to the snapshot dependencies if we take snapshots out of the temporal relations.

Let I be a temporal relation on scheme R, we define the snapshot of I at valid time a and transaction time b, written $\text{Snapshot}_{V=a;T=b}(I)$, as

$$\text{Snapshot}_{V=a;T=b}(I) = \{t[R]: t \in I \text{ and } a \in t[V] \text{ and } b \in t[T]\}$$

An example of a snapshot of I is shown in fig.4.2.

We then have these two theorems:

Theorem 1. If I satisfies TFD X→Y, then the snapshot of I at V=a and T=b satisfies the classical FD X→Y for all a and b.

Theorem 2. If I satisfies TMVD X↠Y, then the snapshot of I at V=a and T=b satisfies the classical MVD X↠Y for all a and b.

The theorems are proved in the appendix. An example of a snapshot of a temporal relation satisfying the corresponding classical dependencies is shown in fig.4.2.

4.4 Lossless Decomposition

The simultaneous projection of a relation I on $X_1, ..., X_m$ is called a decomposition. If the natural join of the projections gives back the original I, that is, if $I = \overset{m}{\underset{j-1}{*}} \pi_{X_j}(I)$,

then we say that the decomposition of I is lossless, since these projections can be joined to recover the original relation with no loss of information. However, if $I \subset \overset{m}{\underset{j-1}{*}} \pi_{X_j}(I)$,

then we say that the decomposition is lossy. An example of a lossless decomposition of a temporal relation is shown in fig.4.3, accompanied by an example of a lossy decomposition in fig.4.4.

We can see that lossless decomposition is linked to the concept of TMVD:

Theorem 3. Let I be a temporal relation on scheme R, and X↠Y an TMVD, and let Z = R-XY, the decomposition of I into $\pi_{XY}(I)$ and $\pi_{XZ}(I)$ is lossless if and only if I satisfies TMVD X↠Y.

The proof is shown in the appendix.

Fig.4.3 Lossless decomposition of a temporal relation which satisfies Course↠Teacher and Course↠Class

I (satisfying Course↠Teacher)

$= \pi_{\{Course,\ Teacher\}}(I) * \pi_{\{Course,Class\}}(I)$

Course	Teacher	Class	Vs	Ve	Ts	Te
Eng	Chan	1	1	10	1	10
Eng	Chan	2	2	10	2	10
Eng	Li	1	3	14	4	18
Eng	Li	2	3	15	4	16
Phy	Li	1	1	10	1	10
Phy	Li	2	1	5	1	-

$\pi_{\{Course,\ Teacher\}}(I)$

Course	Teacher	Vs	Ve	Ts	Te
Eng	Chan	1	10	1	10
Eng	Li	3	15	4	18
Phy	Li	1	10	1	-

$\pi_{\{Course,\ Class\}}(I)$

Course	Class	Vs	Ve	Ts	Te
Eng	1	1	14	1	18
Eng	2	2	15	2	16
Phy	1	1	10	1	10
Phy	2	1	5	1	-

Fig.4.4 Lossy decomposition of a temporal relation (using I in fig.4.3)

$\pi_{\{Course,\ Teacher\}}(I)$

Course	Teacher	Vs	Ve	Ts	Te
Eng	Chan	1	10	1	10
Eng	Li	3	15	4	18
Phy	Li	1	10	1	-

$\pi_{\{Teacher,\ Class\}}(I)$

Teacher	Class	Vs	Ve	Ts	Te
Chan	1	1	10	1	10
Chan	2	2	10	2	10
Li	1	1	14	1	18
Li	2	1	15	1	-

$\pi_{\{Course,\ Teacher\}}(I)\ *\ \pi_{\{Teacher,Class\}}(I) \neq I$

Course	Teacher	Class	Vs	Ve	Ts	Te
Eng	Chan	1	1	10	1	10
Eng	Chan	2	2	10	2	10
Eng	Li	1	3	14	4	18
Eng	Li	2	3	15	4	*18*
Phy	Li	1	1	10	1	*18*
Phy	Li	2	1	5	1	-

Chapter 5

Asynchronous Dependency

In classical relations, no past data are stored. Therefore, data dependencies among past data will not cause problems. However, temporal relations may exhibit anomalies due to the presence of past data and their dependencies.

Asynchronous attributes are known to be one of the sources of anomalies. Two attributes are said to be synchronous to each other if the values of the two attributes for an entity always change values at the same (valid) time, otherwise they are said to be asynchronous.

For example, in fig.5.1 the attributes SAL and RANK are asynchronous to each other - when salary of employee 1010 change from 4000 to 4500 at valid time 3, the rank remains unchanged.

Update anomalies may arise in this temporal relation. If the rank AP (at valid time 1 to 5) of employee 1010 should be updated to rank P1 (maybe due an error in the past resulting in the wrong rank of AP stored), two tuples have to be changed, instead of just one.

Fig.5.1 Asynchronous dependency

SAL~RANK satisfied

EMP	SAL	RANK	Vs	Ve	Ts	Te
1010	4000	AP	1	3	1	-
1010	4500	AP	3	5	3	-
1010	5000	SA	5	NOW	5	-
2020	7000	MGR	1	5	1	-

5.1 Asynchronous Dependency

The existence of asynchronous attributes in a temporal relation can be viewed as a semantic constraint over the relation. This type of constraint is called *asynchronous dependency* (AD). AD is defined below.

In a temporal relation I on scheme R with key K, $XY \subseteq R$ and X, Y disjoint, asynchronous dependency $X \sim Y$ is said to be satisfied by I if there exist two tuples $u, v \in I$ such that

i. $u[K] = v[K]$

and ii. u[V], v[V] are adjacent

and iii. u[T] and v[T] not disjoint

and iv. $(u[X]=v[X]$ and $u[Y] \neq v[Y])$

or $(u[X] \neq v[X]$ and $u[Y]=v[Y])$

The definition is essentially the definition of asynchronous attributes, and an AD is said to be satisfied by each pair of asynchronous attributes which exists in the same temporal

relation. In fig.5.1, SAL and RANK are asynchronous to each other, we say that SAL~RANK is satisfied by the relation.

5.2 Asynchronous Normal Form

A temporal relation is said to be in Asynchronous Normal Form (ANF) if and only if no AD among the non-key attributes is satisfied by the relation.

We can losslessly decompose a relation with an AD into a set of relations each containing synchronous attributes only. The resulting set of relations will be each in ANF.

For example, a relation I on scheme R with key K satisfying AD X~Y can be decomposed into $\pi_{KX}(I)$ and $\pi_{R-X}(I)$. Fig.5.2 shows an example of the decomposition.

ANF is achieved by lossless decomposition of a temporal relation into a number of relations containing synchronous attributes only. But we can see that, by the result in section 4.4, that a decomposition is lossless is always guaranteed by the key of the relation (since the key K always obey the TMVD K↠A where A is any attribute). The AD that is present in the temporal relation, which is a stronger condition, is not directly responsible for ensuring the decomposition being lossless.

Fig.5.2 Decomposition into ANF

I (with key EMP and AD SAL~RANK)

EMP	SAL	RANK	Vs	Ve	Ts	Te
1010	4000	AP	1	3	1	-
1010	4500	AP	3	5	3	-
1010	5000	SA	5	NOW	5	-
2020	7000	MGR	1	5	1	-

$\pi_{\{EMP,\ SAL\}}(I)$

EMP	SAL	Vs	Ve	Ts	Te
1010	4000	1	3	1	-
1010	4500	3	5	3	-
1010	5000	5	NOW	5	-
2020	7000	1	5	1	-

$\pi_{\{EMP,\ RANK\}}(I)$

EMP	RANK	Vs	Ve	Ts	Te
1010	AP	1	5	1	-
1010	SA	5	NOW	5	-
2020	MGR	1	5	1	-

5.3 Generalized Form of Data Dependency

In the preceding sections we have introduced the asynchronous dependency in the temporal relational model. In the remaining sections we will give definitions of what we think generalized data dependencies are and finally compare these forms of data dependency with the asynchronous dependency.

In the literature, several types of data dependencies have been studied. The most classical ones are the functional dependencies (FD), the multivalued dependencies (MVD), and the join dependencies (JD). In the early eighties there have been attempts to define a single general class of data dependencies and to study its properties. The *embedded implicational dependency* (EID) proposed by Fagin [1982] and the *algebraic dependency* defined by Yannakakis and Papadimitriou [1982] are shown to generalize most known dependencies. Furthermore these two classes are shown to be identical. Thus the EID or the algebraic dependency can be used as a yardstick to measure how an arbitrary constraint resembles what is informally known as a data dependency.

5.3.1 Embedded Implicational Dependency [Fagin 1982]

Before we defined the EID we will first introduce some preliminary concepts. Let P a relation symbol. The relational formulas are logical formulas of the form $Pz_1...z_d$. Equalities are of the form x = y. Formulas without free variables are called sentences. We will abbreviate $\forall x_1 \ldots \forall x_n \phi$ where each x_i is universally quantified, by $(\forall x_1 \ldots x_n)\,\phi$. Similarly, $\exists y_1 \ldots \exists y_r \phi$ is abbreviated by $(\exists y_1 \ldots y_r)\,\phi$.

A formula is said to be typed if there are d disjoint types of variables, so that (a) if the relational formula $Pz_1...z_d$ appears in the formula, then z_i is of type i, and (b) if x=y appears in the formula, then x and y have the same type. The result is that in a typed formula no variable can represent an entry in two distinct columns.

An EID is a typed sentence of the form

$$(\forall x_1 \ldots x_m)\ ((A_1 \wedge A_2 \ldots \wedge A_n) \rightarrow (\exists y_1 \ldots y_k)\ (B_1 \wedge \ldots \wedge B_s))\ .$$

where each A_i is a relational formula and each B_i is either a relational formula or an equality. We assumed also that each of the x_j's appears in at least one of the A_i's and that $n \geq 1$. Also, assumed that $k \geq 0$ and that $s \geq 1$.

Example 1

EID can be thought of as a constraint which says that if certain tuples can be found in a relation, then some other tuples can be found or some of the entries are equal in value. We will shown that FD, MVD and JD are EID's.

Assumed that we are dealing with a 4-ary relation P with attributes ABCD. The FD A→B can be expressed as the EID

$$(\forall a b_1 b_2 c_1 c_2 d_1 d_2)\ ((P a b_1 c_1 d_1 \wedge P a b_2 c_2 d_2) \rightarrow (b_1 = b_2))\ .$$

The MVD A↠B can be expressed as the EID

$$(\forall a b_1 b_2 c_1 c_2 d_1 d_2)\ ((P a b_1 c_1 d_1 \wedge P a b_2 c_2 d_2) \rightarrow (P a b_1 c_2 d_2))\ .$$

JD(AB, ACD, BCD) can be expressed as EID

$$(\forall a_* a_2 b_* b_3 c_1 c_* d_1 d_*)\,((Pa_* b_* c_1 d_1 \wedge Pa_2 b_* c_* d_* \wedge Pa_* b_3 c_* d_*) \rightarrow (Pa_* b_* c_* d_*))\,.$$

5.3.2 Algebraic Dependency [Yannakakis and Papadimitriou 1982]

First lets introduce some preliminary concepts. Let R be a relation with attributes U={A, B, ..., Z}. The extension R′ of R is a relation with attributes U′ ={A_1, B_1, ..., Z_1, A_2, B_2, ..., Z_2, A_3, B_3, ..., Z_3, ... }, and that R′ ={(t,t,...): t∈R}. R′ is therefore, an infinite collection of copies of R.

An algebraic dependency is a assertion of the form

$$\phi_1(R') \subseteq \phi_2(R')$$

where ϕ_1 and ϕ_2 are expression involving only the projection and join operations

(called a project-join expression). That is, an algebraic dependency is a constraint stating that two project-join expression involving a single relation satisfy an set inequality.

Example 2

Using the 4-ary relation of example 1, the FD A→B can be expressed as the algebraic dependency

$$\pi_{AB_1}(R') * \pi_{AB_2}(R') \subseteq \pi_{AB_1B_2}(R')$$

and the MVD A↠B can be expressed as

$$\pi_{AB}(R') * \pi_{ACD}(R') \subseteq \pi_{ABCD}(R')$$

the JD(AB, BCD, ACD) can be expressed as

$$\pi_{AB}(R') * \pi_{ACD}(R') * \pi_{BCD}(R') \subseteq \pi_{ABCD}(R')$$

It has been proved that the class of EID's and algebraic dependencies are identical [Yannakakis and Papadimitriou 1982]. That is, every EID can be expressed as an algebraic dependency, i.e. an equation between two project-join expression.

5.4 Asynchronous Dependency versus Synchronous Dependency

One problem regarding the definition of asynchronous dependency is that it does not resemble the familiar types of data dependency, i.e. EID's. In the EID's, we have to see if all tuples in a relation satisfy a certain property, i.e. we use a "for all" (universal) quantifier in the logical formula expressing the dependency.

The asynchronous dependency as defined in the above discussion differs from the usual type. It is based on a formula with a "there exist" (existential) quantifier.

Based on this, one may wish to define the Synchronous Dependency (SD), which is the negation of AD, and use SD to construct the theory for asynchronous attributes. In a temporal relation I on scheme R with key K, XY⊆R and X, Y disjoint, synchronous dependency *X sync Y* is said to be satisfied by I if for all tuples u, v ∈ I we have

i. u[K] = v[K]

and ii. u[V], v[V] are adjacent

and iii. u[T] and v[T] not disjoint

implies (u[X]≠v[X] and u[Y]≠v[Y])

or (u[X]=v[X] and u[Y]=v[Y])

SD resemble EID in that the universal quantifier is used. However, SD differs from EID in that in the consequent of an SD, it does not only say that some values must be equal, it also say that some values must not be equal. Therefore, we can see that SD is not an EID.

The above analysis is somewhat imprecise because EID is not defined for the temporal relational model but for the snapshot relational model. The exact analysis should use a EID format for temporal relations. But to construct the required EID format we have to solve a lot of problems, especially the reconstruction of the axioms for projection and join [Yannakakis and Papadimitriou 1982] for the temporal relational model.

On the other hand, that whether SD (or even an AD) can be expressed as a equation of project-join expression remains unanswered. A positive answer to the this question implies that SD is actually a subclass of a temporal version of algebraic dependency, while a negative answer to the question will mean that the properties of SD or AD have to be derived anew.

Chapter 6

Conclusions

In this chapter, a summary of the thesis will be presented, followed by a discussion of some unsolved problems and research directions in the context of temporal database theory.

6.1 Summary of the Thesis

As the literature of temporal database research has not provide a solid foundation of the theoretical and design aspects of the topic, this research attempts to construct a theoretical framework for temporal databases based on the temporal relational model. In the introduction, the concept of temporal databases, the concept of valid time versus transaction time have been discussed. We have differentiated the four types of databases according to whether their associations with the valid time or transaction time or both.

The temporal relational data model is the basis for the discussions and analysis in subsequent chapters. We define the structural elements in chapter 2, while a formal query language, the temporal relational algebra, is defined in chapter 3.

In chapter 4, this research extends the two classical data dependencies to the temporal relational model, resulting in the temporal functional dependency (TFD) and the temporal multivalued dependency (TMVD). TMVD is shown to be the necessary and sufficient condition for the lossless decomposition of two temporal relations.

A new type of data dependency is introduced in chapter 5. It is based on the concept of asynchronous attributes and is termed the asynchronous dependency (AD). It is shown to be the cause of update anomalies. A normal form, the asynchronous normal form is introduced as a condition that no anomaly is aroused by AD. It can be achieve by decomposing temporal relations. Finally, we have a discussion of the possibility of incorporating AD in some generalized forms of data dependencies - the embedded implicational dependency or the algebraic dependency.

6.2 Unsolved Problems and Research Directions

There are a number of unsolved problems and further research directions which the author found promising. They concern mainly with the theoretical aspects of temporal databases.

6.2.1 Equivalent Representations in the Temporal Relational Model

Observe that the two temporal relations shown in fig.6.1 conveyed the same information. The fact that two different relations can represent the same information is not a property inherited from classical relational model but is introduced by the addition of the time attributes.

In view of this, for all operations defined on temporal relations we have to ensure that the result is in some kind of "compressed" form as in relation J. A compress operation needs to be defined for temporal relations.

Fig.6.1 Example of equivalent representation

I

X	Y	Vs	Ve	Ts	Te
a	b	3	4	3	-
a	b	4	6	3	-

J

X	Y	Vs	Ve	Ts	Te
a	b	3	6	3	-

6.2.2 The Notion of "Completeness" of Temporal Query Languages

Historically, Codd has selected a version of the relational calculus as a standard for evaluating the expressive power of relational query languages [Codd 1972b]. Languages that can express all the queries in the relational calculus is said to be *complete*, which have a meaning that all "reasonable" queries can be expressed in the language (assumed the set of queries expressible in relational calculus to be reasonable). Later, Chandra and Harel define a different view of completeness where a complete language is one which expressed exactly the set of queries that are computable [Chandra and Harel 1980].

Although a number of temporal query languages have been defined, there is no accepted notion of a *temporal complete* language. We do not know what the set of reasonable temporal queries includes, and we do not have an idea of the set of computable temporal

queries. But for the evaluation and comparison of the various temporal query languages, there is a need to define the notion of temporal complete language.

6.2.3 Logical Basis for Temporal Data Models and Languages

The relational model has a strong relationship with predicate logic. Relational databases can be seen as interpretations of a first-order theory or as a first-order theory itself [Galliare et al. 1984]. Relational calculus is based upon the syntax and semantics of logic. These connection of databases with logic enables the development of the field of deductive or logic databases.

There is a lack of study on the relationship of logic and temporal databases. The relationship identified will be instrumental to the development of temporal deductive databases and may help to relate the currently disparate fields of temporal databases on one hand and temporal reasoning and temporal logics in artificial intelligence [Shoham 1988] on the other.

6.2.4 Other Temporal Dependencies

This thesis have identified a number of data dependencies in the temporal relational model. These dependencies are considered because they will be present in temporal relations of diverse applications and they each have important implications in logical design of databases.

The author does not claim to have identified all of these interesting data dependencies. Other dependencies may exist which also have implications in logical design. A further research direction then is to identify these dependencies and to construct a general theory of data dependencies in temporal data models.

6.2.5 Research Directions in Topics other than Theory

The research directions suggested above concentrate on the theoretical aspects of temporal databases. There are many other important issues, which include topics in physical data structures, physical design of databases, implementation issues, and the relationship of temporal databases to other forms of databases such as object-oriented or active databases [Snodgrass 1990].

Appendix

Proofs of Theorems

Theorem 1 to 3 in the main text is proved as follows:

Theorem 1

If a temporal relation r satisfies the TFD X→Y, then the snapshot of r at V=a and T=b satisfies the classical FD X→Y for all a and b.

Proof

Let s = $Snapshot_{V=a;T=b}(r)$ and t1, t2 be two tuples of s with t1[X] = t2[X] = x. But for all tuples u1, u2 in r with u1[X] = u2[X] = x, a ∈ u1[V], a ∈ u2[V], b ∈ u1[T] and b ∈ u2[T], by TFD X→Y, we have u1[Y] = u2[Y]. Therefore t1[Y] = t2[Y].

Theorem 2

Let r be a temporal relation on scheme R, and let X, Y, Z be subsets of R such that Z = R-XY. If r satisfies the TMVD X↠Y, then the snapshot of r at V=a and T=b satisfies the classical MVD X↠Y for all a and b.

Proof

Let s = $Snapshot_{V=a;T=b}(r)$ and t1, t2 be two tuples of s with t1[X] = t2[X] = x. Then there exists two tuples u1, u2 in r with u1[X] = u2[X] = x, u1[Y] = t1[Y], u1[Z] = t1[Z], a ∈ u1[V], b ∈ u1[T], u2[Y] = t2[Y], u2[Z] = t2[Z],a ∈ u2[V], and b ∈ u2[T]. Then by TMVD X↠Y, there exists a tuple u in r with u[X] = u1[X] = x, u[Y] = u1[Y] = t1[Y],

u[Z] = u2[Z] = t2[Z], u[V]⊇u1[V]∩u2[V], u[T]⊇u1[T]∩u2[T]. Therefore, a ∈ u[V] and b ∈ u[T]. Therefore by u, there exists tuple t in s with t[X] = u[X] = x, t[Y] = u[Y] = t1[Y], t[Z] = u[Z] = t2[Z]. So classical MVD X↠Y is satisfied by s.

Theorem 3

Let r be a temporal relation on scheme R, and let X, Y, Z be subsets of R such that Z = R-XY. Relation r satisfies the TMVD X↠Y if and only if the decomposition of r into $\pi_{XY}(r)$ and $\pi_{XZ}(r)$ is lossless.

Proof

(Only if part) Suppose the TMVD holds. Let m = $\pi_{XY}(r)$ and n = $\pi_{XZ}(r)$. Let t be a tuple in m * n. Then there must be a tuple t1 ∈ m and a tuple t2 ∈ n such that t[X] = t1[X] = t2[X], t[Y] = t1[Y], t[Z] = t2[Z], t[V] = t1[V]∩t2[V] and t[T] = t1[T]∩t2[T]. Since m and n are projections of r, there must be tuples t3, t4 in r with t1[XY] = t3[XY], t1[VT] = t3[VT] and t2[XZ] = t4[XZ], t2[VT] = t4[VT]. By TMVD X↠Y, induced by t3 and t4, there must be a tuple t5 in r, with t5[X] = t3[X] = t4[X], t5[Y] = t3[Y], t5[Z] = t4[Z], t5[V]⊇t3[V]∩t4[V] and t5[T]⊇t3[T]∩t4[T]. But t5[X] = t3[X] = t1[X] = t[X], t5[Y] = t3[Y] = t1[Y] = t[Y], t5[Z] = t4[Z] = t2[Z] = t[Z], t5[V]⊇t3[V]∩t4[V] = t1[V]∩t2[V] = t[V] and t5[T]⊇t3[T]∩t4[T] = t1[T]∩t2[T] = t[T]. Therefore t5 includes the description of t. But r⊆m * n: since if u ∈ r, then u[XYVT] ∈ m and u[XZVT] ∈ n, so u[XYZVT] = u ∈ m * n. Therefore r=m * n.

(If part) Let m and n be defined as above. Suppose now that r decomposes losslessly into m and n. Let u1 and u2 be tuples in r such that u1[X] = u2[X] and both

u1[V]∩u2[V] and u1[T]∩u2[T] are non-empty. Then m must contain tuple u3 with u3 = u1[XYVT] and n must contain a tuple u4 = u2[XZVT]. Since r = m * n, r contains a tuple u such that u[XY] = u3[XY] = u1[XY], u[XZ] = u4[XZ] = u2[XZ], u[V] = u3[V]∩u4[V] = u1[V]∩u2[V] and u[T] = u3[T]∩u4[T] = u1[T]∩u2[T]. Therefore, u satisfies the TMVD X↦Y requirements induced by u1 and u2.

Bibliography

Abiteboul, S., and Bidiot, N. (1986). Non first normal form relations: an algebra allowing data restructuring. *Journal of Computer and System Sciences*, 33, pp.361-393.

Ahn, I., and Snodgrass, R. (1986). Performance evaluation of a temporal database management system. In *Proc. SIGMOD '86 Intl. Conf. on Mgt. of Data* (Washington DC, May). ACM, New York, pp.96-107.

Ahn, I., and Snodgrass, R. (1988). Partitioned storage for temporal databases. *Information Systems*, 13(4), pp.369-391.

Allen, J.F. (1983). Maintaining knowledge about temporal intervals. *Comm. ACM*, 26(11), pp.832-843.

Ariav, G. (1986). A temporally oriented data model. *ACM Trans. Database Systems*, 11(4), pp.499-527.

Bancilhon, F. (1978). On the completeness of query languages for relational data bases. In *Mathematical Foundations of Computer Science 1978 (MFCS'78), Proc., 7th Symp.* (Zakopane, Poland). Lecture Notes in Comp. Sci. 64, Springer, Berlin, pp.112-124.

Bolour, A., et al. (1982). The role of time in information processing: a survey. *ACM SIGMOD RECORD*, 12(3), pp.27-50.

Chandra, A.K., and Harel, D. (1980). Computable queries for relational data bases. *Journal of Computer and System Sciences*, 21, pp.156-178.

Chat, G.S.-W. (1990). *An Extended Query Language for a Temporal Relational Database*. Master's Thesis, Department of Computer Science, The Chinese University of Hong Kong.

Chen, P.P.-S. (1976). The entity-relationship model - toward a unified view of data. ACM Trans. Database Systems, 1(1), pp.9-36.

Clifford, J., and Ariav, G. (1986). Temporal data management: models and systems. In *New Directions for Database Systems*, G. Ariav and J. Clifford, eds. Ablex, Norwood, New Jersey, pp.169-185.

Clifford, J., and Tansel, A.U. (1985). On an algebra for historical relational databases: two views. In *Proc. ACM SIGMOD Intl. Conf. on Mgt. of Data* (Austin, Texas, May). ACM, New York, pp.247-265.

Clifford, J., and Warren, D.S. (1983). Formal semantics of time in databases. *ACM Trans. Database Systems*, 8(2), pp.214-254.

Codd, E.F. (1970). A relational model of data for large shared data banks. *Comm. ACM*, 13(6), pp.377-387.

Codd, E.F. (1972a). Further normalization of the data base relational model. In *Data Base Systems*, R. Rustin, ed. Prentice-Hall, Englewood Cliffs, New Jersey, pp.33-64.

Codd, E.F. (1972b). Relational completeness of data base sublanguages. In *Data Base Systems*, R. Rustin, ed. Prentice-Hall, Englewood Cliffs, New Jersey, pp.65-98.

Copeland, G. (1982). What if mass storage were free? *IEEE Computer*, 15(7), pp.27-35.

Copeland, G., and Maier, D. (1984). Making Smalltalk a database system. In *Proc. ACM SIGMOD Intl. Conf. on Mgt. of Data* (Boston, June), pp.316-325.

Dadam, P. et al. (1984). Integration of time versions into a relational database system. In *Proc. 10th Intl. Conf. on Very Large Data Bases* (Singapore, Aug.). VLDB Endowment, pp.509-522.

Date, C.J. (1990). *An Introduction to Database Systems*, volume I, 5th edition. Addison-Wesley, Reading, Massachusetts.

Fagin, R. (1981). A normal form for relational databases that is based on domains and keys. *ACM Trans. Database Systems*, 6(3), pp.387-415.

Fagin, R. (1982). Horn clauses and database dependencies. *J. ACM*, 29, pp.952-985.

Fagin, R., and Vardi, M.Y. (1986). The theory of data dependencies - a survey. In *Mathematics of Information Processing*, M. Ansel and W. Gerwirtz, eds. American Mathematical Society, Providence, Rhode Island, pp.19-71.

Gadia, S.K., and Yeung, C.-S. (1988). A generalized model for a relational temporal database. In *Proc. ACM SIGMOD Intl. Conf. on Mgt. of Data* (Chicago, Illinois, June). ACM, New York, pp.251-259.

Gallaire, H. et al. (1984). Logic and databases: a deductive approach. *ACM Computing Surveys*, 16(2), pp.153-185.

Kambayashi, Y. et al. (1983). Synthesis of unnormalized relations incorporating more meaning. *Information Sciences*, 29, pp.201-247.

Klopprogge, M.R. (1983). Term: an approach to include the time dimension in the entity-relationship model. In *Entity-Relationship Approach to Information Modeling and Analysis*, P.P. Chen, ed. Elsevier North-Holland, pp.473-508.

Klopprogge, M.R., and Lockemann, P.C. (1983). Modelling information preserving databases: consequences of the concept of time. In *Proc. 9th Intl. Conf. on Very Large Data Bases* (Florence, Italy). VLDB Endowment, pp.399-416.

Kobayashi, I. (1985). Temporal aspects of databases: interaction between state and event relations. In *Database Semantics (DS-1)*, T.B. Steel, Jr. and R. Meersman, eds. North-Holland, Amsterdam, pp.223-232.

Lorentzos, N.A., and Johnson, R.G. (1988). Extending relational algebra to manipulate temporal data. *Information Systems*, 13(3), pp.289-296.

Maier, D. (1983). *The Theory of Relational Databases.* Computer Science Press, Rockville, Maryland.

McKenzie, E. (1986). Bibliography: temporal databases. *ACM SIGMOD RECORD*, 15(4), pp.40-52.

McKenzie, E., and Snodgrass, R. (1987). Extending the relational algebra to support transaction time. In *Proc. ACM SIGMOD 1987 Annual Conf.* (San Francisco, Calif., May). ACM, New York, pp.467-478.

Navathe, S.B., and Ahmed, R. (1987a). *A Temporal Relational Model and a Query Language*. Technical Report, Database Systems Research and Development Center, University of Florida, Gainesville.

Navathe, S.B., and Ahmed, R. (1987b). TSQL: a language interface for history databases. In [Rolland 1987].

Paredaens, J. et al. (1989). *The Structure of the Relational Database Model.* Springer, Berlin.

Peckman, J., and Maryanski, F. (1988). Semantics data models. *ACM Computing Surveys*, 20(3), pp.153-189.

Rolland, C. et al., eds. (1987). *Temporal Aspects in Information Systems*. North-Holland, Amsterdam.

Roth, M.A., Korth, H.F. (1987). The design of ¬1NF relational databases into nested normal form. In *Proc. ACM SIGMOD 1987 Annual Conf.* (San Francisco, May). ACM, New York, pp.143-159.

Sadra, N.L. (1990). Extensions to SQL for historical databases. *IEEE Trans. Knowledge and Data Engg.*, 2(2), pp.220-230.

Segev, A., and Shohani, A. (1987). Logical modeling of temporal data. In *Proc. ACM SIGMOD 1987 Annual Conf.* (San Francisco, Calif., May). ACM, New York, pp.454-466.

Shoham, Y. (1988). *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, Cambridge, Massachusetts.

Shoshani, A., and Kawagoe, K. (1986). Temporal data management. In *Proc. 12th Intl. Conf. on Very Large Data Bases* (Kyoto, Aug.). VLDB Endowment.

Snodgrass, R., ed. (1986). Research concerning time in databases: project summaries. *ACM SIGMOD RECORD*, 15(4), pp.19-39.

Snodgrass, R. (1987). The temporal query language TQuel. *ACM Trans. Database Systems*, 12(2), pp.247-298.

Snodgrass, R. (1990). Temporal databases: status and research directions. *ACM SIGMOD RECORD*, 19(4), pp.83-89.

Snodgrass, R., and Ahn, I. (1985). A taxonomy of time in databases. In *Proc. ACM SIGMOD Intl. Conf. on Mgt. of Data* (Austin, Texas, May). ACM, New York, pp.236-246.

Snodgrass, R., and Ahn, I. (1986). Temporal databases. *IEEE Computer*, 19(9), pp.35-42.

Soo, M.D. (1991). Bibliography on temporal databases. *ACM SIGMOD RECORD*, 20(1), pp.14-23.

Tansel, A.U. (1986). Adding time dimension to relational model and extending relational algebra. *Information Systems*, 11(4), pp.343-355.

Tansel, A.U., and Garnett, L. (1989). Nested historical relations. In *Proc. 1989 ACM SIGMOD Intl. Conf. on Mgt. of Data* (Portland, Oregon, May). ACM, New York, pp.284-293.

Tu, S.W. et al. (1989). Episodic skeletal-plan refinement based on temporal data. *Comm. ACM*, 32(12), pp.1439-1455.

Ullman, J.D. (1981). A view of directions in relational database theory. In *Automata Languages and Programming (ICALP'81), 8th International Colloquium on* (Acre, Israel, June). Lecture Notes in Comp. Sci. 115, Springer, Berlin, pp.165-176.

Ullman, J.D. (1982). *Principles of Database Systems*, 2nd edition. Computer Science Press, Rockville, Maryland.

Vardi, M.Y. (1988). Fundamentals of dependency theory. In *Trends in Theoretical Computer Science*, E. Borger, ed. Computer Science Press, Rockville, Maryland, pp.171-224.

Yannakakis, M., and Papadimitriou, C.H. (1982). Algebraic dependencies. *Journal of Computer and System Sciences*, 25, pp.2-41.