MATHEMATICAL FOUNDATIONS FOR RELATIONAL DATA BASES

Dissertation for the Degree of Ph. D.
MICHIGAN STATE UNIVERSITY
RAYMOND YOUSSEF FADOUS
1975

K 4-5-75
pd $32.00

ABSTRACT

MATHEMATICAL FOUNDATIONS FOR RELATIONAL DATA BASES

By

Raymond Youssef Fadous

A new approach to data management systems has been the introduction of a relation or table as a model for a data base. Large sets of data can be represented in a few large tables, but such a representation often leads to certain anomalies whenever data items in the data base are added, deleted, or changed. To reduce the effect of these anomalies, previous research identified functional relations or dependencies between attributes and defined second and third normal forms. These normal forms are dependent on minimal subsets of attributes, called candidate keys or simply keys, which uniquely identify each row of a table whenever the usual operations of retrieval, deletion, and update are performed. The research reported in this thesis considers the problem of constructing algorithms for finding keys for relational data bases and for determining whether a relation is in second or third normal form. The thesis presents a new algorithm which starts with the functional relations and finds all keys of a normalized relation.

The mathematical properties of a relation in second and third normal forms are studied in detail along with the properties of prime and non-prime attributes and algorithms are given for determining whether a relation is in either second or third normal form.

Finally, this thesis points to a weakness in the definitions of a relation in third normal form, as proposed by Codd and Kent, and advances the concept of a canonical normal form to overcome the disclosed weakness.

MATHEMATICAL FOUNDATIONS FOR RELATIONAL DATA BASES

By

Raymond Youssef Fadous

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1975

ACKNOWLEDGEMENTS

I am very grateful to Dr. John J. Forsyth, my thesis adviser and the co-chairman of my committee, for his encouragement, suggestions, and help in writing this thesis in a clear and concise mathematical form. I would like to thank Dr. Carl V. Page, the co-chairman of my committee, for introducing me to this area of research. My thanks to Dr. J.S. Frame and Dr. Richard C. Dubes for serving on my committee and for the helpful discussions I had with them while writing this thesis. Thanks go to the Division of Engineering Research for providing me with financial assistance to do research. My thanks to Dr. E.F. Codd of IBM for generously providing the necessary references in this area of research.

Special thanks are due to my parents, my brothers, my wife Dola, and to my children Sandy and Joe. Their love and understanding helped me sustain the hard work of the doctoral program.

TABLE OF CONTENTS

Chapter | Page

GLOSSARIAL CROSS REFERENCE

Term	Definition Number	Page
Adjacent	3.2.1	35
Algorithm A1	-	27
Algorithm A2	-	63
Algorithm A3	-	71
Attribute	-	5
Attribute Value	-	5
Candidate Key	1.3.4	8
Canonical Normal Form (CNF)	5.2.3	78
Connected	3.2.2	35
Dependent	1.3.2	7
First Normal Form (FNF)	1.3.7	9
Full Dependence	1.3.3	8
Functional Dependence	1.3.2	7
Functional Relation	1.3.2	7
Identity Partition	3.2.3	37
Implication Matrix	-	19
Implies	1.3.2	7
Key	1.3.4	8
KSNF	5.2.1	75
KTNF	5.2.2	76

CHAPTER 1

SURVEY OF RELATIONAL DATA BASE THEORY

The trend in computer applications is to make the computer accessible to a wide range of users, especially casual users, who have little or no training in programming. It is estimated, by IBM researchers Codd, et. al [28], that in the 1990's the growth in on-line interaction by casual users will exceed that for all other users by a large factor. Such users need a simple logical notion of the data organization in order to form queries in a sensible way. One of the key developments in data base design in recent years has been the introduction of a relation or table as a model for a data base. The tabular representation of data is the simplest and most universally understood data structure. The initial structuring of the data can often be described by a few large tables with many columns. However, such a structure is usually inefficient and contains redundant information which is not suitable for direct storage. Codd [22] discussed the possibility of breaking up a table into smaller ones, in what he termed second and third normal forms, so as to remove these shortcomings. For this reason, Codd introduced the concept of a functional relation between attributes and the important notion of a key to help project a relation into subrelations in second and third normal forms, and also to recover the original table from the projections by a special operation called the natural join.

Existing methods for finding keys are conceptually elegant but computationally intimidating; these methods suffer from the curse of dimensionality or combinatorial explosion associated with finding prime implicants of Boolean functions having a large number of variables. The algorithm in this thesis, in contrast, proceeds by identifying subsets which can lead to keys and then carefully selecting the appropriate intersections of these subsets which produce the keys. Further, Codd gives only the definition as a basis for a relation to be in a normal form. But, how can the date base administrator know whether the relations are in second or third normal form? This thesis provides the answer to this question.

Kent [48] examined the underlying definitions of normal forms, as proposed by Codd, and suggested some improvements. However, in this thesis, further examination of Kent's alternative definitions reveals that the objectives of the normal forms were not totally met, and this led to the introduction of a new canonical normal form that better meets the criteria for normal forms in relational data bases.

1.1 Background

One of the main objectives in the design of data base systems is the concept of data independence. Briefly stated, this means that application programs are not affected by the way data is stored. The data base is the data as physically stored, also referred to as the storage structure. The user's view of the data base is called the data model, also known as the logical structure. That is, to the user the data model is the data base. Date, et. al [29, 30] give many examples and explain these terms. Engles [40] and Meltzer [53] provide a good background in the design of data independent accessing models.

Many different data models that claim data independence have appeared in the literature since 1960. Two of these, the CODASYL Data Base Task Group (DBTG) network model [17-20] and the relational model, as proposed by E.F. Codd [21], have each attracted a large number of followers. Bachman [4, 5], one of the main contributors to the DBTG approach, clearly presents the goals desired in a data independent model. The string model, as proposed by Senko, et. al [1, 3, 63], is also a data independent accessing model, based on the work of Davies [35], Engles [40], and Meltzer [53], but is overshadowed by the two previously mentioned models.

Date [32] gives a good introduction to the DBTG approach. Engles [41] presents a thoughtful criticism of the DBTG model and lists the IBM objections to it. Date, et. al [33] and Codd, et. al [28] explain the main differences between the network and relational approaches to data base design. The main differences and, in Codd's view, the advantages of the relational model are simplicity, uniformity, completeness, and data independence. To a large extent, simplicity may be considered the justification of the relational approach. All the records in a file are stored as n-tuples in a relation -- that is as a table. Also, the data sublanguages that operate on the relational data base are easier to learn than the data manipulation language of the DBTG approach. The relational model is uniform in the sense that any relationship between entities or relations is also expressed as a relation. It is complete in the sense that all data structures commonly used in data base systems can be expressed in a relational form. As for data independence, neither the data model nor the languages contain any reference to storage structure or to access methods. Date [31] summarizes, in a tutorial form, the main concepts of these two as well as other models.

This thesis deals exclusively with the relational model as originally proposed by Codd [21]. The relational approach is motivated primarily, but not exclusively, by the desire to attract the casual user. The data structure used is in the simplest form possible, the tabular form. Codd [27] states the steps necessary to help the user interact with the computer in a natural dialogue with the objective of attaining an agreement between the user and the system as to the user's needs. One of these steps is the development of the language for manipulating the data model, which is known as the data sublanguage. Codd [24] defined the data sublanguage ALPHA (DSL-ALPHA), based on the first-order predicate calculus, for expressing the user's queries. Queries can be expressed in terms of a collection of operations on the relations [21, 25], and this collection is called the relational algebra. An earlier information algebra, with a different set of operations, was defined by Bosak [9]. Codd [25] proves that his DSL-ALPHA is complete in the sense that any query expressable in the relational algebra is also expressable in the relational calculus. Other languages have been defined for accessing data in a relational data base. Boyce, et. al [10] proposed a data sublanguage called SQUARE. It uses a mathematical notation for expressing queries, but is less sophisticated mathematically than DSL-ALPHA; hence, SQUARE is easier to use by the casual user. They have also shown that SQUARE is a complete sublanguage. In later papers, Boyce, et. al [11] and Chamberlin, et. al [15] presented another sublanguage called SEQUEL that has the functional capabilities of SQUARE but different syntax. SQUARE is a concise mathematical APL-like notation, while SEQUEL is a block structured English keyword language.

There has been much research done on the relational model and Codd [26] reviews briefly the latest and the current areas of investigations that have been undertaken.

1.2 The Relational Model

Given sets, $D_1, D_2, \ldots, D_n$, called domains, not necessarily distinct, a relation $\mathfrak{R}$ of degree n, defined over $D_1, D_2, \ldots D_n$, is a subset of the cartesian product $D_1 \times D_2 \times \ldots \times D_n$. That is, $\mathfrak{R}$ is a set of elements each of the form $(d_1, d_2, \ldots, d_n)$ where each d_i is an element of D_i. The set D_i is called the i-th domain of $\mathfrak{R}$. Each element $(d_1, d_2, \ldots, d_n)$ is called an n-tuple or, simply, tuple of $\mathfrak{R}$. Each d_i is the i-th component of a tuple. An attribute is a name assigned to a domain of a relation. Any value associated with an attribute is called an attribute value. While the domains of a relation need not be distinct, the attribute names assigned to them must all be distinct. The relations are time-varying relations; tuples may be updated, deleted, and inserted in a relation. Throughout this thesis, the term relation means a time-varying relation. The logical view of a relation of degree n is a rectangular array or table with n columns and m rows such that m varies from one update to another. This work deals only with normalized relations. These are relations whose attribute values are simple and are not themselves relations. Codd [21, 23] lists five properties that are satisfied by any normalized relation $\Re$. These properties are:

1. $\Re$ is column homogeneous, that is in any one column all the attribute values are of the same kind, whereas items in different columns need not be of the same kind.
2. Each attribute value is a simple number or a character string and not a set of numbers or a repeating group.
3. All rows of a relation must be distinct.
4. The ordering of rows within a relation is immaterial.
5. Since the attributes are distinct, the ordering of columns within a relation is immaterial.

With property 5 added, the exact mathematical term is relationship and not a relation, but here this distinction will be ignored. A relation which satisfies property 2 is said to be normalized. Date, et. al [33] give a good exposition of the logical structure of the data base. In the relational approach, the data model definition (DMD) defines the relations and the underlying attributes that together constitute the relational model.

1.3 Normal Forms

The notion of functional dependence, as defined by Codd [22], plays a fundamental role in the theory which governs the decomposition of relations into subrelations in normal forms. Codd [23] listed six aims of normalization of relations. The two most important are:

1. To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the life span of application programs.
2. To reduce the incidence of undesirable insertion, update, and delition anomalies.

An example will be given later, in this section, to explain these different anomalies. Delobel, et. al [36] use the term "functional relation" to mean functional dependence. A whole theory of relations can be built around this simple concept. First, one needs to define the term projection of a relation $\mathfrak{R}$ on a subset of attributes of $\mathfrak{R}$. The notation used in the following definition is explained in Chapter 2.

Definition 1.3.1

Let $\mathfrak{R}$ be a relation defined on the set of attributes $\Omega = A_1A_2A_3 \ldots A_n$. For any $\alpha = A_1A_2 \ldots A_m$, a subset of Ω, the projection of $\mathfrak{R}$ on α is defined as:

$$\mathfrak{R}_\alpha = \{(a_1,a_2,\ldots,a_m) \mid (a_1,a_2,\ldots,a_n) \in \mathfrak{R}\} ,$$

also written as $\mathfrak{R}[A_1A_2 \ldots A_m]$.

Definition 1.3.2

The set of attributes B in a relation $\mathfrak{R}$ of a degree n is functionally dependent or just dependent on the set of attributes A in $\mathfrak{R}$, if, at any instant of time, there exists a function, called functional relation, $F : \mathfrak{R}[A] \rightarrow \mathfrak{R}[B]$ or simply $A \rightarrow B$. The term A implies B is also used whenever $A \rightarrow B$.

The notation $A \not\rightarrow B$ indicates that B is not dependent on A, while $A \leftrightarrow B$ says that A and B are dependent on each other.

Definition 1.3.3

The set of attributes B in a relation $\mathfrak{R}$ is fully dependent on the set of attributes A in $\mathfrak{R}$, written as $A \Rightarrow B$, if

1. B is dependent on A and
2. B is not dependent on any proper subset of A.

If B is not fully dependent on A, one writes $A \not\Rightarrow B$, while $A \Leftrightarrow B$ indicates that A and B are fully dependent on each other.

The concept of a key, that will be defined next, is the mathematical basis of the relational model. The normal forms are defined in terms of the candidate keys or simply keys of the relation. Also, the search algorithms use the keys of a relation to retrieve or insert information in the data base. The remainder of this thesis uses the term key instead of candidate key.

Definition 1.3.4

Each key of a relation $\mathfrak{R}$ is a non-empty subset, K, of Ω, the set of all attributes in $\mathfrak{R}$, such that Ω is fully dependent on K; or in symbols $K \Rightarrow \Omega$.

Definition 1.3.5

An attribute in $\mathfrak{R}$ is called a prime attribute if it is a member of any key of $\mathfrak{R}$. Otherwise, it is non-prime.

Definition 1.3.6

Every relation $\mathfrak{R}$ has a primary key which is chosen arbitrarily from the set of all keys of $\Re$. Also, no attribute in a primary key is allowed to have an undefined value.

Definition 1.3.7

Every normalized relation is said to be in first normal form (FNF).

Example 1.3.1

Consider the teaching schedule relation, T1, defined on the attributes P = Professor name, C = Course number, R = Room number, H = Hour of the course meeting, and O = Office number of a professor. Assume that a course may be taught by many professors and that a professor may teach many courses, but that a professor can only be in one room at any one time. Also, assume that each professor is assigned exactly one office number, and that many professors can share the same office. The above statement of the problem provides the following functional relations:

$$PH \rightarrow RC;\ RH \rightarrow PC;\ P \rightarrow O\ .$$

At some instant of time T1 might look like this:

T1 :

P	C	R	H	O
Forsyth	817	1	9	400
Forsyth	818	1	10	400
Dubes	805	2	9	400
Dubes	817	2	10	400
Frame	851	3	9	243
Frame	831	4	11	243
Page	841	2	11	404
Page	841	2	12	404

The relation T1 is in FNF because all the entries in T1 are simple. The combinations of attributes RH and PH are the only keys of T1 based only on the functional relations given above. Then, P, R, and H are prime attributes, while C and O are non-prime. Choose PH, say, as the primary key.

Observe that if Professor Forsyth moved to office number 402, then more than one tuple has to be updated. This is called an update anomaly. If Professor Page is not teaching a course this year but he will next year, then the information about his office number should be retained; but, if the Page tuples are deleted, that information is lost. This is an example of a deletion anomaly. Finally, suppose one wishes to record the office number of visiting Professor Jones who has not been yet assigned a course number to teach. Since PH is the primary key, one must fabricate a fictitious hour number in order to store this information. This is called an insertion anomaly.

To reduce the effect of these anomalies, transitive dependence of attributes upon one another were introduced and the second and third normal forms were defined by Codd [22].

<u>Definition 1.3.8</u>

A relation $\mathfrak{R}$ is in <u>second normal form</u> (SNF) if

1. $\mathfrak{R}$ is in first normal form and
2. Every non-prime attribute in $\mathfrak{R}$ is fully dependent on every key of $\mathfrak{R}$.

<u>Example 1.3.2</u>

If relation T1, in Example 1.3.1, were projected into two subrelations, T2[PCRH] and T3[PO], then, T3, for example, looks like this:

T3 :	P	O
	Forsyth	400
	Dubes	400
	Frame	243
	Page	404

The only key of T3 is P, hence the primary key, while T2 has the same keys as T1. Note that T2 and T3 are each in second normal form. This is so, since T3 has a simple key, that is a key with just one attribute, and C, the non-prime attribute in T2, is not dependent on any proper subsets of the keys PH and RH. The anomalies previously mentioned have disappeared.

<u>Definition 1.3.9</u>

The set of attributes C in a relation $\mathfrak{R}$ is <u>transitively dependent</u> on the set of attributes A in $\mathfrak{R}$, written as $A \dashrightarrow C$ if

1. A and C are disjoint and
2. There exists a set of attributes B in $\mathfrak{R}$, disjoint from both A and C, such that $A \rightarrow B$, $B \not\rightarrow A$, and $B \rightarrow C$. Otherwise, C is <u>non-transitively dependent</u> on A.

<u>Definition 1.3.10</u>

A relation $\mathfrak{R}$ in second normal form is in <u>third normal form</u> (TNF) if every non-prime attribute in $\mathfrak{R}$ is non-transitively dependent on each key of $\mathfrak{R}$.

Example 1.3.3

Given the following data base with relation T4 defined on the attributes P = Professor name, C = City of residence, and Z = Zip code. At some instant of time T4 might look like this:

T4 :

P	C	Z
Page	E.L.	48823
Frame	E.L.	48823
Forsyth	E.L.	48823
Dubes	E.L.	48823

The statement of the problem provides the following functional relations:

$$P \rightarrow C;\ C \rightarrow Z\ .$$

The only key of T4 is P, hence the primary key, and so, by definition, T4 is in second normal form; but T4 is not in third normal form since Z is transitively dependent on P. Suppose that the Zip Code of E.L. is changed to 48824, then many tuples would have to be updated. This is a consequence of the transitive dependence. To get rid of this anomaly, one can project T4 into T5[PC] and T6[CZ].

T5 :

P	C
Page	E.L.
Frame	E.L.
Dubes	E.L.
Forsyth	E.L.

and T6 :

C	Z
E.L.	48823

Now, T5 and T6 are each in third normal form, as were T2 and T3.

Definition 1.3.11

Let $\mathfrak{R}_1$ and $\mathfrak{R}_2$ be any relations over the set of attributes $\Omega_1 = \{a,b\}$ and $\Omega_2 = \{b,c\}$ respectively such that $\alpha = \Omega_1 \cap \Omega_2 \neq \phi$, then the natural join, denoted $\mathfrak{R}_1 * \mathfrak{R}_2$, of $\mathfrak{R}_1$ and $\mathfrak{R}_2$ over α is defined by $\mathfrak{R}_1 * \mathfrak{R}_2 = \{(a,b,c) \mid (a,b) \in \mathfrak{R}_1 \text{ and } (b,c) \in \mathfrak{R}_2\}$. This definition can be extended to any sets Ω_1 and Ω_2 such that $\alpha = \Omega_1 \cap \Omega_2 \neq \phi$.

Example 1.3.4

Consider the relations T4, T5, and T6 defined in Example 1.3.3. Then T4 = T5 * T6. The natural join is only one of the relational operations introduced by Codd [21, 24]. In projecting a relation $\mathfrak{R}$ into subrelations of $\mathfrak{R}$, it is important to be able to recover the original relation $\mathfrak{R}$ from the subrelations. This idea is emphasized later in the definitions of optimal normal forms. This recovery can be done with the natural join operation. Suppose one projects a relation $\mathfrak{R}[ABC]$ into $\mathfrak{R}_1[AB]$ and $\mathfrak{R}_2[BC]$; it is not always true that $\mathfrak{R} = \mathfrak{R}_1 * \mathfrak{R}_2$. Delobel, et. al [36] gave the sufficient condition that, if $B \rightarrow C$, then $\mathfrak{R}[ABC] = \mathfrak{R}_1[AB] * \mathfrak{R}_2[BC]$.

Although this thesis does not deal with optimization, the following definitions are given here to complete the presentation of the relational model as originally proposed by Codd.

Definition 1.3.12

The transitive dependence $A \dashrightarrow C$, in a relation $\mathfrak{R}$, is a strict transitive dependence if there exists a set of attributes B in $\Re$ disjoint from A and C such that:

1. $A \rightarrow B$, $B \not\rightarrow A$ and
2. $B \rightarrow C$, $C \not\rightarrow B$.

<u>Definition 1.3.13</u>

A collection of relations C, which are projections in a relation $\Re$, is in <u>optimal second normal form</u> if the following three conditions hold:

1. All the relations in C are in second normal form.
2. $\Re$ is the natural join of the relations in C.
3. No smaller collection of relations has these properties.

<u>Definition 1.3.14</u>

Let C_2 be a collection of relations in optimal second normal form, and C_3, a collection of relations in third normal form, which are projections from the relations in C_2. Then, the collection C_3 is in <u>optimal third normal form</u> if the following 4 conditions hold:

1. All the relations in C_3 are in third normal form.
2. C_2 can be recovered with the natural join of the relations in C_3.
3. No relation in C_3 contains any pair of attributes that are strictly transitively dependent on any relation in C_2.
4. No smaller collection of relations has these properties.

The relational model, as proposed by Codd, lacks simple and efficient procedures to find all keys, and to determine whether a relation is in second or third normal form. This thesis reports efforts made to take constructive steps in this direction.

attributes B in $\Re$ disjoint from A and C such that:

1. $A \rightarrow B$, $B \not\rightarrow A$ and
2. $B \rightarrow C$, $C \not\rightarrow B$.

Definition 1.3.13

A collection of relations C, which are projections in a relation $\Re$, is in optimal second normal form if the following three conditions hold:

1. All the relations in C are in second normal form.
2. $\Re$ is the natural join of the relations in C.
3. No smaller collection of relations has these properties.

Definition 1.3.14

Let C_2 be a collection of relations in optimal second normal form, and C_3, a collection of relations in third normal form, which are projections from the relations in C_2. Then, the collection C_3 is in optimal third normal form if the following 4 conditions hold:

1. All the relations in C_3 are in third normal form.
2. C_2 can be recovered with the natural join of the relations in C_3.
3. No relation in C_3 contains any pair of attributes that are strictly transitively dependent on any relation in C_2.
4. No smaller collection of relations has these properties.

The relational model, as proposed by Codd, lacks simple and efficient procedures to find all keys, and to determine whether a relation is in second or third normal form. This thesis reports efforts made to take constructive steps in this direction.

1.4 Contributions and Organization of the Thesis

Functional relations are shown to enjoy a rich algebraic and set theoretic structure, and that the normal forms can be studied within this framework.

This mathematical structure leads to a new approach for starting with the functional relations and finding all of the keys in a normalized relation. The algorithm uses an implication matrix, its transitive closure and a systematic method for introducing attributes to form keys.

A detailed mathematical characterization of prime and non-prime attributes leads to the concepts of functional partition, reduction, and deletion. Also, it is shown that, under certain conditions, many computational steps can be saved in the algorithm for finding the keys.

The mathematical properties of a relation in second and third normal forms are studied in detail and algorithms are defined for determining whether a relation is in any of these forms. Then, this thesis proceeds to point to a weakness in the definitions of third normal form, as proposed by Codd and Kent [48], and a new canonical normal form (CNF), an improvement in terms of reducing the update anomaly, is suggested.

The basic definitions of the relational model, as given by Codd, are all included in Chapter 1. Chapter 2 defines the implication matrix and its transitive closure which form the basis of the new algebraic approach, taken in this thesis, in the study of the normal forms. Also, in Chapter 2, the algorithm for finding all keys in a relation is presented. Chapter 3 deals with the mathematical properties of prime and non-prime attributes, and with the new concepts of functional partition, reduction, and deletion. In Chapter 4, the mathematical properties, and algorithms for the normal forms, are given. Conclusions and suggestions for future research are offered in Chapter 5.

All definitions, theorems and lemmas in the remainder of this thesis are the original work of the author unless otherwise indicated.

CHAPTER 2

FINDING KEYS FOR RELATIONAL DATA BASES

2.1 Introduction

E.F. Codd [21] proposed the relational data base model in an effort to provide "data independence" for applications programs. An applications programmer working with a relational data base may view the data as being stored in tabular form; each unique collection of data items which describes a single entity occupies a unique row of some table. A number of different collections of tables may be used to represent any given data base. Certain collections might be preferable to others if they eliminate exceptional conditions, or anomalies, which can occur when items in the data base are added, deleted, or changed. Several researchers have investigated normal forms for relational data bases and characterized their role in eliminating addition, deletion, and insertion anomalies. A good overview of these concepts can be obtained by reading Codd [22-23] and Kent [48].

In order to establish an appropriate normal form for a relational data base, one must identify minimal subsets of attributes, called <u>keys</u>, which uniquely determine the values of the remaining attributes, so that the rows of the tables in the data base have the required property of uniqueness. Delobel and Casey [36] developed an algorithm for finding all keys in a relational data base, given the set of fundamental functional relations defined on the data base. Their method maps the functional relations to a Boolean function and produces the keys from those prime implicants of the Boolean function which cover the implicant which consists of all of the uncomplemented variables of the function. This Chapter describes an alternative algorithm, whose correctness is shown to derive directly from the fundamental properties of the functional relations as given by Armstrong [2].

2.2 Basic Concepts and Notation

In the discussion which follows, the symbol Ω will be used to denote the set of all n attributes $A_1, A_2, \ldots, A_n$ on which a relation $\mathfrak{R}$ of degree n is defined. The empty set is denoted by ϕ. The union and intersection operations between sets will be denoted by $\cup$ and $\cap$, respectively. If A is a set, then $|A|$ is the cardinality of A. The usual set notation for a set Ω with n elements $A_1, A_2, \ldots, A_n$ is $\Omega = \{A_1, A_2, \ldots, A_n\}$; while here the notation $\Omega = A_1 A_2 \ldots A_n$ is used. Two sets X and Y are <u>disjoint</u> if $X \cap Y = \phi$. A <u>partition</u> on a set Ω is a collection of non-empty disjoint subsets of Ω whose union is Ω.

<u>Definition 2.2.1</u>

Let L_i and R_i, $i = 1,2,\ldots,m$, be non-empty subsets in Ω, then the $\{L_i \rightarrow R_i\}$ is called the <u>set of functional relations in Ω</u>.

Armstrong [2] has shown that the following properties, P1 - P5, are sufficient to find the set of all functional relations that can be derived from a given set of functional relations. Let A, B, C, and D be any non-empty subsets of Ω then,

P1. Reflexivity: $A \rightarrow A$;

P2. Transitivity: If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$;

P3. Augmentation: If $A \rightarrow B$, then $A' \rightarrow B$ for any $A' \supseteq A$;

P4. Projectivity: If $A \rightarrow B$, then $A \rightarrow B'$ for any $B' \subseteq B$ and $B' \neq \emptyset$;

P5. Additivity: If $A \rightarrow B$ and $C \rightarrow D$, then $A \cup C \rightarrow B \cup D$.

The purpose of this chapter is to exhibit an algorithm which generates all keys which can be found using only the given functional relations and properties P1 - P5. The starting point is a set of functional relations specified by the data base administrator or the user. Let $\{L_i \rightarrow R_i\}$, $i = 1,2,\ldots,m$, be the set of functional relations in Ω. Without loss of generality, one can impose the restrictions that $L_i \neq L_j$ for $i \neq j$ and that the intersection $L_i \cap R_i$, $i = 1,2,\ldots,m$, is empty. Note that if $AB \rightarrow BC$, then it is true that $AB \rightarrow C$, but it is not necessarily true that $A \rightarrow BC$ or that $A \rightarrow C$.

The <u>implication matrix</u> P of a relation $\Re$ of degree n, defined on $\Omega = A_1A_2 \ldots A_n$, is an $m \times n$ matrix whose rows are labeled $L_1, L_2, \ldots, L_m$ and whose columns are labeled $A_1, A_2, \ldots, A_n$ such that:

$$L_iA_j = \begin{cases} 1 & \text{if} \quad A_j \in (L_i \cup R_i) \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,n$.

Example 2.2.1

Consider the following functional relations where Ω = ABCDEFG.

$$ABC \rightarrow DEG;\ AB \rightarrow CF;\ CD \rightarrow EF;\ EG \rightarrow AC\ .$$

Then, the implication matrix P is

$$P = \begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ ABC & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ AB & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ CD & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ EG & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}$$

The fact that row labels of P are not isomorphic to subsets of the column labels requires an extension of the usual method of finding the transitive closure of an implication matrix. The transitive closure, P^*, of P is defined as follows:

1. Put $P^* = P$
2. For every two distinct rows L_i and L_j in P^*, if for every attribute A_k in L_j, the entry $L_iA_k = 1$, then copy all 1 entries in row L_j into the corresponding entries in row L_i. The 0 entries in row L_i change to 1 if the corresponding entries in row L_j are 1 and the original 1 entries in row L_i will remain. This changes P^*.
3. Repeat part 2 above until P^* cannot be changed any further.

Example 2.2.2

Starting with P in Example 2.2.1, then

$$P^* = \begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ ABC & 1 & 1 & 1 & 1 & 1 & \underline{1} & 1 \\ AB & 1 & 1 & 1 & \underline{1} & \underline{1} & 1 & \underline{1} \\ CD & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ EG & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}$$

where $\underline{1}$ means a 0 has been changed to 1 in the process of taking the transitive closure.

Before proceeding with the proof of the basic theorems that support the algorithm, the following notation is needed. From the definition of P^*, P^* might effectively indicate a new set of functional relations. These new functional relations are denoted $L_i \to T_i$, $i = 1,2,\ldots,m$, where $T_i \supseteq R_i$, $L_i \cap T_i = \phi$. In P^*, it might be the case that some rows are not all 1. The set of attributes that correspond to 0 entries in row L_i will be denoted by T_i'. Note that $T_i' = \phi$ if row L_i is all 1 in P^* and that $\pi = L_i T_i T_i'$ is a partition of Ω, if $T_i' \neq \phi$. It is also assumed that L_i and T_i are not empty.

2.3 Mathematical Preliminaries

The mathematical properties of the functional relations are stated in Lemmas 2.3.1 - 2.3.7, and the theoretical foundations for the steps taken in the algorithm for finding the keys are presented in Theorems 2.3.1 - 2.3.3.

Lemma 2.3.1

In P^*, if $L_i \to T_i$ and $L_i \cup T_i \neq \Omega$, then L_i can be extended by the subset T_i' such that $L_i \cup T_i' \to \Omega$.

Proof:

Since $L_i \to T_i$ and $L_i \to L_i$, then $L_i \to L_i \cup T_i$ (by P5). Also $T_i' \to T_i'$ (by P1), hence $L_i \cup T_i' \to L_i \cup T_i \cup T_i' = \Omega$ (by P5).

Lemma 2.3.2

In P^*, if row L_j is all 1 and $L_j \subseteq L_i \cup T_i$, then row L_i is also all 1.

Proof:

Since row L_j is all 1, this implies that $L_j \to \Omega$. Also since $L_i \to T_i$, then $L_i \to L_i \cup T_i$ (by P5). But $L_i \cup T_i \to L_j$ (by P4) and so $L_i \to L_j \to \Omega$ (by P2).

Lemma 2.3.3

In P^*, if L_j is a subset of $L_i \cup T_i$ and if row L_i is not all 1, hence row L_j is not all 1, then, for T_i' and T_j' such that $L_i \cup T_i' \to \Omega$ and $L_j \cup T_j' \to \Omega$, the intersection of T_i' and T_j' is not empty.

Proof:

It follows from the transitive property P2, that any 1 entry in row L_j is a 1 entry in row L_i and hence any 0 entry in row L_i is a 0 entry in row L_j. But T_i' corresponds to the 0 entries in row L_i, hence $T_i' \cap T_j' \neq \phi$.

Example 2.3.1

$\Omega = ABCDE$

1. $AB \to C$
2. $AC \to D$

$$P^* = \begin{array}{c|ccccc} & A & B & C & D & E \\ AB & 1 & 1 & 1 & \underline{1} & 0 \\ AC & 1 & 0 & 1 & 1 & 0 \end{array}$$

then $T_1' = E$, $T_2' = BE$ and $T_1' \cap T_2' \neq \phi$.

Lemma 2.3.4

In P^*, if L_j is a subset of $L_i \cup T_i$ and if row L_i is not all 1, then for any $\alpha_j \subseteq T_j'$ such that $L_j \cup \alpha_j \to \Omega$, the intersection of T_i' and α_j is not empty.

Proof:

The proof is by contradiction. If $\alpha_j \cap T_i' = \emptyset$, then $\alpha_j \subseteq L_i \cup T_i$ since $L_i T_i T_i'$ is a partition on Ω. Also $L_j \subseteq L_i \cup T_i$, hence $L_j \cup \alpha_j \subseteq L_i \cup T_i$. But this would imply that $L_i \to L_i \cup T_i \to L_j \cup \alpha_j \to \Omega$, which is a contradiction since $L_i \not\to \Omega$. Therefore, $\alpha_j \cap T_i' \neq \emptyset$.

Lemma 2.3.5

In P^*, if L_j is not a subset of $L_i \cup T_i$ and if row L_j is all 1 and row L_i is not all 1, then for T_i' such that $L_i \cup T_i' \to \Omega$, the intersection of T_i' and L_j is not empty.

Proof:

If the intersection of T_i' and L_j is empty, this implies that the 1 entries in row L_i include L_j and hence L_j would be a subset of $L_i \cup T_i$ which contradicts the hypothesis. Therefore, $T_i' \cap L_j \neq \emptyset$.

Example 2.3.2

Ω = ABCDE

1. A $\to$ BC
2. AD $\to$ E

$$P^* = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 1 & 1 & 1 & 0 & 0 \\ AD & 1 & \underline{1} & \underline{1} & 1 & 1 \end{array}$$

then $T_1' = DE$, $L_2 = AD$ and $T_1' \cap L_2 \neq \emptyset$.

Lemma 2.3.6

In P^*, if L_j is not a subset of $L_i \cup T_i$ and if row

L_j is not all 1 and row L_i is not all 1, then for T_i' such that $L_i \cup T_i' \rightarrow \Omega$, the intersection of T_i' and L_j is not empty.

Proof:

The proof is exactly the same as in Lemma 2.3.5.

Example 2.3.3

Ω = ABCDE

1. A → BC

2. BD → A

$$P^* = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 1 & 1 & 1 & 0 & 0 \\ BD & 1 & 1 & \underline{1} & 1 & 0 \end{array}$$

then T_1' = DE, L_2 = BD and $T_1' \cap L_2 \neq \emptyset$.

Lemma 2.3.7

In P^*, if L_j is a subset of $L_i \cup T_i$, then L_i and $L_i \cup L_j$ imply the same subset in Ω.

Proof:

In P^*, $L_i \rightarrow L_i \cup T_i$ and so, by definition of T_i, if $Z_i \subseteq \Omega$ such that $Z_i \supset L_i \cup T_i$, then $L_i \not\rightarrow Z_i$. Similarly, $L_j \rightarrow L_j \cup T_j$. But, by hypothesis, $L_j \subseteq L_i \cup T_i$ and so $L_i \rightarrow L_i \cup T_i \rightarrow L_j \rightarrow L_j \cup T_j$ by projectivity and transitivity respectively which implies that $L_j \cup T_j \subseteq L_i \cup T_i$. Also $L_i \rightarrow T_i$ and $L_j \rightarrow T_j$, and so

$L_i \cup L_j \rightarrow L_i \cup T_i \cup L_j \cup T_j = L_i \cup T_i$ since

$L_j \cup T_j \subseteq L_i \cup T_i$. Therefore, L_i and $L_i \cup L_j$ imply the same subset, $L_i \cup T_i$, in Ω.

Example 2.3.4

Refer back to Example 2.3.1. Note that L_1 = AB → ABCD and $L_1 \cup L_2$ = ABC → ABCD.

Theorem 2.3.1

In P^*, if L_j is not a subset of $L_i \cup T_i$, then there exists a subset $\alpha \subseteq L_j$, possibly $\alpha = \emptyset$ if row L_i is all 1, such that $L_i \cup \alpha$ and $L_i \cup L_j$ imply the same subset in Ω.

Proof:

If $L_i \rightarrow \Omega$, then $\alpha = \emptyset$ and hence it is always true that $L_i \cup L_j \rightarrow \Omega$.

If $L_i \not\rightarrow \Omega$, then by Lemmas 2.3.5 and 2.3.6, for T_i' such that $L_i \cup T_i' \rightarrow \Omega$, it follows that $\alpha = T_i' \cap L_j \neq \emptyset$. So $\alpha \subseteq L_j$. Since $L_i T_i T_i'$ is a partition on Ω, then either $L_j \subseteq T_i'$ or $L_j \cap (L_i \cup T_i) \neq \emptyset$. If $L_j \subseteq T_i'$, then $\alpha = L_j \cap T_i' = L_j$ and $L_i \cup \alpha = L_i \cup L_j$ and hence they imply the same subset in Ω. If $L_j \not\subseteq T_i'$, then $L_j \cap (L_i \cup T_i) \neq \emptyset$. But $L_j = (L_j \cap L_i) \cup (L_j \cap T_i) \cup (L_j \cap T_i')$ since $L_i T_i T_i'$ is a partition on Ω; hence $L_j = [L_j \cap (L_i \cup T_i)] \cup \alpha$. Also $L_j \cap (L_i \cup T_i) \subseteq L_i \cup T_i$, so $L_j = [L_j \cap (L_i \cup T_i)] \cup \alpha \subseteq L_i \cup T_i \cup \alpha$. But $L_i \rightarrow T_i$, so $L_i \cup \alpha \rightarrow T_i$. Therefore, using Lemma 2.3.7, if $L_j \subseteq (L_i \cup \alpha) \cup T_i$ where $L_i \cup \alpha \rightarrow T_i$, then the subset in Ω dependent on $L_i \cup \alpha$ is the same subset dependent on the union $(L_i \cup \alpha) \cup L_j = L_i \cup L_j$ since $\alpha \subseteq L_j$.

Example 2.3.5

In Example 2.3.3, note that $\alpha = D \subseteq L_2 = BD$ and that $A \cup D$ and $A \cup BD$ imply the same subset, ABCD, in Ω.

If $K = L_i \cup \alpha_i$ is a key, then K is said to be <u>derived</u> from L_i.

The following theorem shows that unions of L_i and L_j are not useful in deriving keys.

Theorem 2.3.2

Any key that can be derived from $L_i \cup L_j$ can also be derived from L_i or L_j separately.

Proof:

By Lemma 2.3.7 and Theorem 2.3.1, for some $\alpha \subseteq L_j$, where α could possibly be empty, $L_i \cup \alpha$ and $L_i \cup L_j$ imply the same subset in Ω. If $L_i \cup \alpha \to \Omega$, then $L_i \cup L_j \to \Omega$, but $L_i \cup L_j \supseteq L_i \cup \alpha$, hence $L_i \cup L_j$ cannot be a key whenever $L_i \cup \alpha$ is unless $L_i \cup L_j = L_i \cup \alpha$. If $L_i \cup \alpha \not\to \Omega$, then, by Lemma 2.3.1, $L_i \cup \alpha$ and $L_i \cup L_j$ can be extended by the same minimum subset β so that $L_i \cup (\alpha \cup \beta) \to \Omega$ and $(L_i \cup L_j) \cup \beta \to \Omega$. But $\alpha \subseteq L_j$ and for any β, $\alpha \cup \beta \subseteq L_j \cup \beta$, hence $L_i \cup (\alpha \cup \beta) \subseteq (L_i \cup L_j) \cup \beta$ and therefore $(L_i \cup L_j) \cup \beta$ can never be a key whenever $L_i \cup (\alpha \cup \beta)$ is unless $(L_i \cup L_j) \cup \beta = L_i \cup (\alpha \cup \beta)$.

The previous theorems and lemmas show that to find the keys of a relation $\mathfrak{R}$, one needs only consider the functional relations $L_i \to T_i$, $i = 1,2,\ldots,m$, and that taking the union $L_i \cup L_j$ of any two L_i, L_j will not result in any new key that cannot be found from L_i or L_j separately. So, the algorithm uses the transitive property repeatedly to find the minimal α_i such that $L_i \cup \alpha_i \to \Omega$. The following theorem asserts that for each $L_i \cup \alpha_i$ in T2, to be explained in the algorithm, $L_i \cup \alpha_i \to \Omega$.

Theorem 2.3.3

If L is any subset of Ω such that $L \to \Omega$, and if $T_i' \neq \phi$, then $L_i \cup (T_i' \cap L) \to \Omega$.

Proof:

By Lemmas 2.3.3 - 2.3.6, $T_i' \cap L \neq \phi$ whenever $T_i' \neq \phi$. Partition L into L_1 and L_2 such that $L_1 \subseteq L_i \cup T_i$ and $L_2 \subseteq T_i'$; this is possible since $L_i T_i T_i'$ is a partition on Ω and $L_2 = T_i' \cap L \neq \phi$. $L_i \to L_i \cup T_i \to L_1$ by projectivity (P4). $T_i' \cap L_2 = L_2 \to L_2$ by reflexivity (P1). Since $T_i' \cap L_2 \subseteq T_i' \cap L$, $T_i' \cap L \to L_2$ by augmentation (P3). Also by additivity (P5), $L_i \cup (T_i' \cap L) \to L_1 \cup L_2 = L \to \Omega$. So, $L_i \cup (T_i' \cap L) \to \Omega$ by transitivity (P2).

2.4 Finding the Keys

Algorithm A1

1. Form P^* with row labels L_i, $i = 1,2,\ldots,m$.
2. $T1 = \phi$ and $T2 = \phi$.
3. For $i = 1$ to m enter into T1 each $L_i \cup T_i'$ where $|T_i'| \geq 0$.
4. If $L_i \cup T_i' \subseteq L_j \cup T_j'$ and if $|T_j'| \leq 1$ and $|T_i'| \leq 1$, then delete $L_j \cup T_j'$ from T1.
5. If $|T_i'| \leq 1$ for all remaining entries in T1 then terminate the algorithm. T1 contains all keys.
   a. Else for all i in T1 such that $|T_i'| \geq 2$ form $\alpha_{ij} = T_i' \cap (L_j \cup T_j')$ for all $j \neq i$ where $|T_j'| \geq 0$.
   b. For each i, delete any $\alpha_{ij'}$ such that $\alpha_{ij} \subseteq \alpha_{ij'}$, $j \neq j'$.

c. For each remaining α_{ij} enter $L_i \cup \alpha_{ij}$ into T2. Then delete from T2 any set which is a superset of any other set in T2.

d. For all i in T1 such that $|T_i'| \geq 2$ and all L_j in T2 for which $L_i \neq L_j$ form $\alpha_{ik} = T_i' \cap (L_j \cup \alpha_{jk})$.

e. For each i, delete any $\alpha_{ik'}$ such that $\alpha_{ik} \subseteq \alpha_{ik'}$, $k \neq k'$.

f. Enter into T2 any $L_i \cup \alpha_{ik}$ which is not a superset of a set already in T2. Then delete from T2 any set which is a superset of any other set in T2. If any new entries are thus created in T2, repeat from step 5_d. Otherwise go to step 6.

6. Copy from T1 into T2 any sets $L_i \cup T_i'$ in T1 where $|T_i'| \leq 1$.

7. Delete from T2 any set which is a superset of another set in T2. The remaining sets in T2 are all of the keys and the algorithm terminates.

By construction, the entries in T1 satisfy $L_i \cup T_i' \rightarrow \Omega$ and, by Theorem 2.3.3, all entries in T2 satisfy $L_i \cup \alpha_{ij} \rightarrow \Omega$.

It remains only to show that the algorithm is complete, in the sense that it finds all keys. If $L_i \cup \alpha_i$, where $|\alpha_i| \geq 0$, is a key, then, for $|\alpha_i| = 0$, L_i is a key only if, in P^*, $L_i \rightarrow \Omega$ and no subset of L_i implies Ω. Therefore, to show completeness it remains only to show that if $|\alpha_i| \geq 1$, then $L_i \cup \alpha_i$ is found by the algorithm. The following completeness theorem will show this. Assume that the set of functional relations has more than one element in it, otherwise it is a trivial case.

Theorem 2.4.1

If $L_i \cup \alpha_i$, $|\alpha_i| \geq 1$, is a key, then $\alpha_i = T_i' \cap (L_j \cup \beta_j)$, $|\beta_j| \geq 0$, for some $j \neq i$.

Proof:

If $L_i \cup \alpha_i$ is a key, then $L_i \cup \alpha_i \to \Omega$. But since $L_i \not\to \Omega$, this implies that there exists a subset $Z_i \subseteq L_i \cup T_i \cup \alpha_i$ such that $Z_i \to T_i'$ where $Z_i = \alpha_i \cup [Z_i \cap (L_i \cup T_i)]$ and $Z_i = \alpha_i$ only if $Z_i \cap (L_i \cup T_i) = \emptyset$. There is always one such Z_i, namely $Z_i = L_i \cup \alpha_i \to \Omega \to T_i'$. Also $\alpha_i = Z_i \cap T_i'$. So, if $Z_i \to T_i'$, one must show that there exists a $j \neq i$, such that $Z_i = L_j \cup \beta_j \to T_i'$, except in the trivial case when there is only one functional relation which would give $\alpha_i = T_i'$ and $Z_i = L_i \cup \alpha_i \to \Omega \to T_i'$. If $L_j \to T_i'$, then let $Z_i = L_j$. This is done for each L_j such that $L_j \to T_i'$, then choose the minimal α_i -- no superset of α_i is chosen, where $\alpha_i = T_i' \cap L_j$ and $L_i \cup \alpha_i \to \Omega$. If $L_j \not\to T_i'$, then for some β_j, $L_j \cup \beta_j \to T_i'$ by augmentation P3 and additivity P5 and for some β_j', $L_j \cup \beta_j \cup \beta_j' \to T_i' \cup L_j \cup \beta_j \cup \beta_j' = \Omega$, where $\beta_j' \subseteq L_i \cup T_i$, again by P3 and P5. Therefore $\beta_j' \cap T_i' = \emptyset$ since $L_i T_i T_i'$ is a partition on Ω. Let $Z_i = L_j \cup \beta_j$, then $T_i' \cap (L_j \cup \beta_j \cup \beta_j') = T_i' \cap (L_j \cup \beta_j) = T_i' \cap Z_i = \alpha_i$. Again this is done for each $L_j \cup \beta_j$ such that $L_j \cup \beta_j \to T_i'$. But, this is exactly how the algorithm iteratively finds the different α_i so that $L_i \cup \alpha_i \to \Omega$ and then chooses the minimal of these α_i.

When Algorithm A1 terminates, every set remaining in T2 is a key. Further, by Theorem 2.4.1, no keys are missed (not in T2). Therefore, the algorithm is complete.

2.5 Examples

Example 2.5.1

The process of finding the keys of the relation in Example 2.2.1, will be presented in a somewhat optimized manner, but the steps still agree with the algorithm.

T1: AB; ABC; CD $\cup$ABG; EG $\cup$ BDF

Delete ABC $\supseteq$ AB from T1, so

T1: AB; CD $\cup$ ABG; EG $\cup$ BDF.

Apply 5_a-5_f of the algorithm as follows.

EG $\cup$ [BDF $\cap$ (CD $\cup$ ABG)] = EG $\cup$ BD; CD $\cup$ [ABG $\cap$ (EG $\cup$ BDF)] = CD $\cup$ BG

EG $\cup$ (BDF $\cap$ AB) = EG $\cup$ B ; CD $\cup$ (ABG $\cap$ AB) = CD $\cup$ AB

Therefore, T2 holds:

T2: EG $\cup$ B; CD $\cup$ AB; CD $\cup$ BG.

Intersect every subset T_i', where $|T_i'| \geq 2$, in T1 with every subset in T2, then add this intersection to L_i of $L_i \cup T_i'$ and put the resulting subset in T2 if it is not a superset of a set already in T2.

BDF $\cap$ CDAB = BD gives EG $\cup$ BD (superset, do not put in T2)

BDF $\cap$ CDBG = BD gives EG $\cup$ BD (superset, do not put in T2)

ABG $\cap$ EGB = BG Gives CD $\cup$ BG (already in T2).

Every T_i', $|T_i'| \geq 2$, in T1 has been intersected with every subset in T2 and no new subsets resulted for T2. Proceed to step 6 of the algorithm to get:

T2: AB; EGB; CDAB; CDBG.

Applying step 7 leaves the keys:

T2: AB; EGB; CDBG.

Example 2.5.2

Consider the following functional relations:

$ABC \rightarrow DF$; $BCD \rightarrow G$; $CE \rightarrow AD$; $DG \rightarrow EF$; $BF \rightarrow CG$.

The transitive closure, P^*, is:

$$
P^* = \begin{array}{c|ccccccc}
 & A & B & C & D & E & F & G \\
ABC & 1 & 1 & 1 & 1 & \underline{1} & 1 & \underline{1} \\
BCD & \underline{1} & 1 & 1 & 1 & \underline{1} & \underline{1} & 1 \\
CE & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
DG & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
BF & 0 & 1 & 1 & 0 & 0 & 1 & 1
\end{array}
$$

So,

T1: ABC; BCD; $CE \cup BFG$; $DG \cup ABC$; $BF \cup ADE$.

Apply 5_a-5_c of Algorithm A1 as follows:

$BF \cup [ADE \cap (DG \cup ABC)] = BF \cup AD$; $BF \cup [ADE \cap (CE \cup BFG)] = BF \cup E$

$BF \cup (ADE \cap BCD) = BF \cup D$; $BF \cup (ADE \cap ABC) = BF \cup A$

$DG \cup [ABC \cap (BF \cup ADE)] = DG \cup AB$; $DG \cup [ABC \cap (CE \cup BFG)] = DG \cup BC$

$DG \cup (ABC \cap BCD) = DG \cup BC$; $DG \cup (ABC \cap ABC) = DG \cup ABC$

$CE \cup [BFG \cap (BF \cup ADE)] = CE \cup BF$; $CE \cup [BFG \cap (DG \cup ABC)] = CE \cup BG$

$CE \cup (BFG \cap BCD) = CE \cup B$; $CE \cup (BFG \cap ABC) = CE \cup B$

Therefore T2 holds:

T2: $BF \cup A$; $BF \cup D$; $BF \cup E$; $CE \cup B$; $DG \cup AB$; $DG \cup BC$.

Now, applying 5_d-5_f of the algorithm gives:

$BF \cup [ADE \cap (CE \cup B)] = BF \cup E$ (already in T2)

$BF \cup [ADE \cap (DG \cup AB)] = BF \cup AD$ (superset, do not put in T2)

$BF \cup [ADE \cap (DG \cup BC)] = BF \cup D$ (already in T2)

$DG \cup [ABC \cap (BF \cup A)] = DG \cup AB$ (already in T2)

$DG \cup [ABC \cap (BF \cup D)] = DG \cup B$ (put in T2 and delete any superset)

$DG \cup [ABC \cap (BF \cup E)] = DG \cup B$ (already in T2 from previous step)

$DG \cup [ABC \cap (CE \cup B)] = DG \cup BC$ (superset, do not put in T2)

$CE \cup [BFG \cap (BF \cup A)] = CE \cup BF$ (superset, do not put in T2)

$CE \cup [BFG \cap (BF \cup D)] = CE \cup BF$ (superset, do not put in T2)

$CE \cup [BFG \cap (BF \cup E)] = CE \cup BF$ (superset, do not put in T2)

$CE \cup [BFG \cap (DG \cup B)] = CE \cup BG$ (superset, do not put in T2)

Therefore, T2 holds;

T2: $BF \cup A$; $BF \cup D$; $BF \cup E$; $CE \cup B$; $DG \cup B$.

Again, applying 5_d-5_f of the algorithm produces no new entries in T2; so proceed to step 6 to get:

T2: $BF \cup A$; $BF \cup D$; $BF \cup E$; $CE \cup B$; $DG \cup B$; ABC; BCD.

Finally, applying step 7 gives all keys:

T2: $BF \cup A$; $BF \cup D$; $BF \cup E$; $CE \cup B$; $DG \cup B$; ABC; BCD.

2.6 Chapter Summary and Remarks

A new algorithm for finding all keys for relational data bases has been presented. It is clear, from the examples given in Section 2.5, that, at least by hand computation, Algorithm A1 seems easier to apply than the algorithm suggested by Delobel and Casey.

The keys can be studied within the framework of the implication matrix which is a familiar algebraic approach. It is important to note that Algorithm A1 is not written in a ready to program notation, and so care must be taken in choosing the appropriate notation for actual programming. Also, optimization is possible and the examples in Section 2.5 should provide some help in this direction.

Finally, Algorithm A1 has not been programmed, and so no experimental comparisons have been made between the approach taken here and that of Delobel and Casey.

CHAPTER 3

FUNCTIONAL PARTITION AND REDUCTION

3.1 Introduction

In this chapter, computational savings are considered which lead to new concepts of functional partition and reduction. Also, detailed mathematical analysis of prime and non-prime attributes is given, and conditions are given under which one can delete a functional relation without changing the keys.

Functional partition provides a way of finding the keys of a relation $\mathfrak{R}$, defined in Ω, from the projections of $\mathfrak{R}$ into subrelations defined on subsets of Ω. This, in turn, implies that transitive closures are taken in submatrices of the implication matrix, P, and so, storage requirements are minimized.

Functional reduction is a direct consequence of the mathematical properties of prime and non-prime attributes which are also discussed in this chapter. It is shown that the keys of a relation $\mathfrak{R}$ defined in Ω depend on the given functional relations in $\mathfrak{R}$ and on a subset, not necessarily proper, of Ω, while in Chapter 2, it was always the case that all the attributes in Ω were used, in Algorithm A1, for finding the keys.

This chapter also shows that functional relations of the form $L_i \rightarrow L_i$ can be deleted without altering the keys.

3.2 Functional Partition

Algorithm A1 in Chapter 2 showed that the keys of a relation $\Re$ are completely determined by the original set of functional relations as specified by the data base administrator or the user. If this set is large, then the number of computational steps for finding the keys is enormous, and so the question is: Can one find the keys of $\Re$ from subsets of the original set of functional relations and, if so, how does one define these subsets? This section presents such subsets provided by the binary operation ($\approx$), defined on the set of functional relations, that partitions this set into equivalence classes.

Let $\Re$ be a relation defined in Ω, and let the set of functional relations in $\Re$ be denoted by $\{F_i\} = \{F_i | L_i \to R_i,\ i = 1,2,\ldots,m\}$. This chapter assumes, unless otherwise specified, that $\Omega = \bigcup_{i=1}^{m} (L_i \cup R_i)$ and this assumption will be substantiated later.

Definition 3.2.1

Two functional relations F_i and F_j in $\{F_i\}$ are called adjacent, written as $F_i \sim F_j$, if $(L_i \cup R_i) \cap (L_j \cup R_j) \neq \phi$. Otherwise, F_i and F_j are called non-adjacent and denoted as $F_i \nsim F_j$.

Definition 3.2.2

F_i and F_j in $\{F_i\}$ are connected, written as $F_i \approx F_j$, if at least one of the following conditions holds:

i) $F_i \sim F_j$;

ii) There exists at least one subcollection $\{F_{\ell_i}\} \subseteq \{F_i\}$ such that $F_i \sim F_{\ell_i}, F_{\ell_i} \sim F_{\ell_{i+1}}, \ldots, F_{\ell_{i+k}} \sim F_j$.

Theorem 3.2.1

The connected relation ($\approx$) is an equivalence relation on $\{F_i\}$.

Proof:

1. $\approx$ is reflexive

It is always true that $F_i \sim F_i$ since $L_i \cup R_i \neq \phi$; hence, $(L_i \cup R_i) \cap (L_i \cup R_i) \neq \phi$. Therefore, by Definition 3.2.2, $F_i \approx F_i$.

2. $\approx$ is symmetric

If $F_i \approx F_j$, and if $F_i \sim F_j$, then $F_j \sim F_i$; hence, $F_j \approx F_i$.

If $F_i \approx F_j$, and if there exists a subcollection $\{F_{\ell_i}\} \subseteq \{F_i\}$ such that $F_i \sim F_{\ell_i}, F_{\ell_i} \sim F_{\ell_{i+1}}, \ldots, F_{\ell_{i+k}} \sim F_j$, then reversing the step gives

$F_j \sim F_{\ell_{i+k}}, \ldots, F_{\ell_{i+1}} \sim F_{\ell_i}, F_{\ell_i} \sim F_i$; hence, $F_j \approx F_i$.

3. $\approx$ is transitive

If $F_i \approx F_j$, and if $F_j \approx F_k$, then there are two subcollections $\{F_{\ell_i}\}$ and $\{F_{\ell_j}\}$, subsets in $\{F_i\}$, each possibly with one element in it, such that

$F_i \sim F_{\ell_i}, F_{\ell_i} \sim F_{\ell_{i+1}}, \ldots, F_{\ell_{i+k}} \sim F_j$ and

$F_j \sim F_{\ell_j}, F_{\ell_j} \sim F_{\ell_{j+1}}, \ldots, F_{\ell_{j+n}} \sim F_k$. But, this implies, by Definition 3.2.2, that $F_i \approx F_k$.

Therefore, the binary operation, $\approx$, partitions $\{F_i\}$ into equivalence classes such that F_i and F_j are in the same class if and only if (iff) $F_i \approx F_j$. The equivalence classes will be denoted by $\overline{F_i}$ where $\overline{F_i}$ is the set of all functional relations connected to F_i. An equivalence class can be denoted by any of its members.

Definition 3.2.3

If for each $j \neq i$, $F_i \not\approx F_j$, then $\approx$ induces the identity partition on $\{F_i\}$. i.e. Each functional relation is in a class by itself.

Definition 3.2.4

If for all i and all j, $F_i \approx F_j$, then $\approx$ induces the universal partition on $\{F_i\}$. i.e. All functional relations are in only one class.

Lemma 3.2.1

A partition on $\{F_i\}$ induces a partition on Ω.

Proof:

If F_i and F_j are in different classes, then it follows that either $F_i \not\sim F_j$ or that there exists no F_k in $\overline{F_j}$ such that $F_i \sim F_k$. Therefore, let C_i be the set of attributes in $\overline{F_i}$, then $C_i \cap C_j = \phi$ for $i \neq j$, and $\cup C_i = \Omega$. Hence, $\{C_i\}$ is a partition on Ω.

3.3 Procedures for Obtaining Functional Partition

Section 3.2 showed that the binary operation ($\approx$) is an equivalence relation on the original set of functional relations. This section presents both a graphical method, suitable for hand computation, and an algebraic method, suitable for hand computation and computer implementation, for producing this partition.

3.3.1 Graphical Method

Given $\{F_i\}$. Consider m points labeled $F_1, F_2, \ldots, F_m$, respectively. For each i and each j, if $F_i \sim F_j$, then draw a line or arc between F_i and F_j, otherwise do not join F_i and F_j. The number of connected subgraphs that results, is the number of equivalence classes in $\{F_i\}$; F_i and F_j are in the same class iff they are in the same connected subgraph.

Whenever each point F_i is an isolated point, then the result is the identity partition. However, if all the points F_i are on the same connected graph, then one gets the universal partition.

Example 3.3.1.1

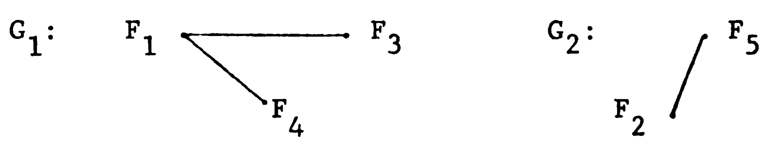
Let Ω = ABCDEFGHKLM. Consider the following functional relations:

F_1: AB $\rightarrow$ C; F_2: H $\rightarrow$ KL; F_3: BD $\rightarrow$ FG; F_4: AE $\rightarrow$ C; F_5: KM $\rightarrow$ L.

Note that $F_1 \approx F_3$ and $F_1 \approx F_4$ since

(AB $\cup$ C) $\cap$ (BD $\cup$ FG) $\neq \phi$ and (AB $\cup$ C) $\cap$ (AE $\cup$ C) $\neq \phi$

respectively. Similarly, $F_2 \approx F_5$. Applying the graphical method gives the following two connected subgraphs G_1 and G_2.



Therefore, there are two equivalence classes $\overline{F_1} = \{F_1, F_3, F_4\}$ and $\overline{F_2} = \{F_2, F_5\}$ that correspond to G_1 and G_2 respectively. Note that $F_3 \approx F_4$ although there is no line joining F_3 and F_4. However, the subgraph G_1 is connected and hence F_1,F_3 and F_4 are in the same equivalence class. Also, the partition on $\{F_i\}$ induces a partition on Ω, namely, if Ω_1 is taken to be the set of attributes mentioned in F_1,F_3 and F_4, then Ω_1 = ABCDEFG and Ω_2 = HKLM which are mentioned in F_2 and F_5; $\Omega_1\Omega_2$ is a partition on Ω.

3.3.2 Algebraic Method

Given $\{F_i\}$, form the implication matrix, P, such that each row, P_i, corresponds to a functional relation F_i. Form the direct incidence matrix, P', from P by letting each entry in P' be the logical vector inner product of rows of P; $P' = [p'_{ij}]$ where $p'_{ij} = p'_{ji} = (P_i, P_j)$ and $(P_i, P_j) = \sum_{k=1}^{\#(\Omega)} p_{ik} \cdot p_{jk}$ where $\cdot$ is logical product and Σ is logical sum. P' is a symmetric matrix of 0's and 1's with all 1's along the diagonal and whose rows and columns are labeled $F_1, F_2, \ldots, F_m$ respectively. The ones in each distinct row of the transitive closure $(P')^*$ of P' define a corresponding equivalence class under $\approx$.

<u>Example 3.3.2.1</u>

Applying the algebraic method to the functional relations in Example 3.3.1.1 gives:

$$P = \begin{array}{c|ccccccccccc} & A & B & C & D & E & F & G & H & K & L & M \\ \hline F_1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ F_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ F_3 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ F_4 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ F_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}$$

$$(P')^* = \begin{array}{c|ccccc} & F_1 & F_2 & F_3 & F_4 & F_5 \\ \hline F_1 & 1 & 0 & 1 & 1 & 0 \\ F_2 & 0 & 1 & 0 & 0 & 1 \\ F_3 & 1 & 0 & 1 & \underline{1} & 0 \\ F_4 & 1 & 0 & \underline{1} & 1 & 0 \\ F_5 & 0 & 1 & 0 & 0 & 1 \end{array}$$

Therefore, $\overline{F_1} = \{F_1, F_3, F_4\}$ and $\overline{F_2} = \{F_2, F_5\}$.

With suitable rearrangements of columns and rows in P, the implication matrix is a rectangular matrix having non-zero rectangular submatrices along its main diagonal with the remaining elements equal to zero.

The following section will show that the keys of $\mathfrak{R}$ can be determined from the non-zero submatrices in the implication matrix, and so storage requirements are reduced because the zero submatrices would not have to be stored.

3.4 Properties of Functional Partition

A partition on $\{F_i\}$ induces a partition on Ω. Let $\Omega_1\Omega_2 \cdots \Omega_n$ be such a partition. The relation $\mathfrak{R}$, defined in Ω, can be projected into subrelations S_i, defined in Ω_i, $i = 1,2,\ldots,n$, respectively. Let $L_{i,j} \to R_{i,j}$ be the set of functional relations $L_i \to R_i$ in $\mathfrak{R}$ that hold in S_j. The following lemma and theorem show that the keys of $\mathfrak{R}$ can be obtained from the keys of S_i, $i = 1,2,\ldots,n$.

Lemma 3.4.1

Let $L_{i,j} \to R_{i,j}$ be the set of functional relations in S_j. If $L_i \to R_i$ is a functional relation in $\mathfrak{R}$ where $L_i \subseteq \Omega$ and $R_{i,j} = R_i \subseteq \Omega_j$, then there exists $K_{i,j} \subseteq L_i$ such that $K_{i,j} \subseteq \Omega_j$ and $K_{i,j} \to R_{i,j}$.

Proof:

The proof follows from the fundamental properties, P1-P5, of the functional relations. The applications of reflexivity, transitivity, projectivity, and additivity on the functional relations in S_j result only in functional relations whose attributes belong to Ω_j. However, applying the augmentation property with attributes not belonging to Ω_j results in a functional relation $L_i \to R_i$ in $\mathfrak{R}$ where $L_i \subseteq \Omega$ and $R_{i,j} = R_i \subseteq \Omega_j$. Therefore, there must exist a subset $K_{i,j} \subseteq \Omega_j$ such that $K_{i,j} \to R_{i,j}$.

The following theorem is the main result of this section. It shows that the keys of $\mathfrak{R}$ can be determined from subsets of the original set of functional relations. Consequently, the major goal of functional partition, as stated at the beginning of Section 3.2, is achieved.

Theorem 3.4.1

If $\Omega = \Omega_1\Omega_2 \cdots \Omega_n$, and if $K \subseteq \Omega$, then the necessary and sufficient conditions for K to be a key of $\mathfrak{R}$ are $K = \bigcup K_i$ where $K_i \neq \phi$ and $K_i \subseteq \Omega_i$ such that K_i is a key of S_i.

Proof:

Necessary condition: Let K be a key of $\mathfrak{R}$. Since $\Omega_1\Omega_2 \cdots \Omega_n$ is a partition on Ω, $K = (K \cap \Omega_1) \cup (K \cap \Omega_2) \cup \ldots \cup (K \cap \Omega_n)$. Let $K_i = K \cap \Omega_i$. One must show that, for each i, $K_i \neq \phi$, and that K_i is the key of S_i. Suppose that, for some i, $K_i = \phi$, then $\overline{K}_i = K = K_1 \cup K_2 \cup \ldots \cup K_{i-1} \cup K_{i+1} \cup \ldots \cup K_n \to \Omega \to \Omega_i$. But $\overline{K}_i$ has no subset that belongs to Ω_i, and so, it contradicts Lemma 3.4.1. Hence, for each i, $K_i \neq \phi$. Therefore, $K = K_1 \cup K_2 \cup \ldots \cup K_n \to \Omega \to \Omega_i$ and so, by Lemma 3.4.1, there must exist a subset $M_i \subseteq K$ such that $M_i \subseteq \Omega_i$ and $M_i \to \Omega_i$. But $K_i \subseteq K$ is the only subset in K that belongs to Ω_i; hence, $M_i \subseteq K_i$ and $M_i \to \Omega_i$. Therefore,

$M = M_1 \cup M_2 \cup \ldots \cup M_n \rightarrow \Omega$ by additivity, and so $M \subseteq K$ which contradicts the hypothesis that K is a key of $\mathfrak{R}$ unless $M_i = K_i$. Therefore, K_i is a key of S_i.

<u>Sufficient condition</u>: If, for each i, $K_i \subseteq \Omega_i$ is a key of S_i, then $K = \cup K_i \rightarrow \Omega$ by additivity. It remains only to show that there is no proper subset of K that implies Ω. Suppose $M \subset K$ and $M \rightarrow \Omega$, then by the same procedure as above, $M = \cup M_i$ where $M_i \subseteq K_i$ and $M_i \neq \phi$. Again applying the same reasoning as above gives $M_i \rightarrow \Omega_i$. But, $M_i \rightarrow \Omega_i$ contradicts the fact that K_i is a key of S_i, unless $M_i = K_i$. Therefore $K = \cup K_i$ is a key of $\mathfrak{R}$.

<u>Corollary 3.4.1.1</u>

If the connected functional relations induce the identity partition on $\{F_i\}$, then $K = \cup L_i$ is the only key of $\mathfrak{R}$.

<u>Proof</u>:

The proof follows from Theorem 3.4.1 where each F_i: $L_i \rightarrow R_i$, $i = 1,2,\ldots,m$, is in an equivalence class by itself. Let $\Omega_i = (L_i \cup R_i)$, and project $\mathfrak{R}$ into $S_i[\Omega_i]$. L_i is the only key of S_i, since there is only one functional relation in each S_i; hence, $\cup L_i$ is the only key of $\mathfrak{R}$ defined in $\Omega = \Omega_1\Omega_2 \cdots \Omega_m$.

<u>Corollary 3.4.1.2</u>

If no functional relations are given in $\mathfrak{R}$ defined in Ω, then Ω is the only key of $\mathfrak{R}$.

<u>Proof</u>:

Let $\Omega = A_1A_2 \cdots A_n$. Since no functional relations are given in $\mathfrak{R}$, then it follows from the reflexive property that

$F_i\colon A_i \to A_i$, $i = 1,2,\ldots,n$, are the only given functional relations in $\mathfrak{R}$. Therefore there exists an identity partition on $\{F_i\}$ and, by Corollary 3.4.1.1, the only key of $\mathfrak{R}$ is $K = \bigcup A_i = \Omega$.

If A and B are sets, then the set difference is defined as $A - B = \{x \mid x \in A \text{ and } x \notin B\}$. The following corollary shows that every attribute in $\mathfrak{R}$, that is not mentioned in any functional relation in $\mathfrak{R}$, belongs to every key of $\mathfrak{R}$.

Corollary 3.4.1.3

Let $\mathfrak{R}$ be a relation in Ω, and suppose that the functional relations in $\mathfrak{R}$ are defined in $\Omega_1 \subset \Omega$. Let $S_1[\Omega_1]$ be the projection of $\mathfrak{R}$ in Ω_1. If K_1 is a key of S_1, then $K = K_1 \cup (\Omega - \Omega_1)$ is a key of $\mathfrak{R}$.

Proof:

Let $S_2[\Omega - \Omega_1]$ be the projection of $\mathfrak{R}$ in $(\Omega - \Omega_1)$. It is clear that no functional relations are given in S_2, except those by reflexivity. Therefore, by Corollary 3.4.1.2, $(\Omega - \Omega_1)$ is the only key of S_2. Also, since Ω_1 and $(\Omega - \Omega_1)$ are disjoint such that $\Omega = \Omega_1 \cup (\Omega - \Omega_1)$, then, by Theorem 3.4.1, $K = K_1 \cup (\Omega - \Omega_1)$ is a key of $\mathfrak{R}$.

The preceding demonstrates that to find the keys of a relation $\mathfrak{R}$, defined in Ω, one needs only be concerned with the connected functional relations in Ω. So, the first step for finding the keys is the possible projection of $\mathfrak{R}$ into subrelations S_i, $i = 1,2,\ldots,n$, where, for each i, the functional relations in S_i are connected. This projection is not always possible, namely, whenever the connected functional relations induce the universal partition on $\{F_i\}$.

Therefore, without loss of generality, one can consider a relation $\mathfrak{R}$, defined in Ω, such that the functional relations in $\mathfrak{R}$ are all connected, and that $\Omega = \bigcup(L_i \cup R_i)$ which substantiates the statement made at the beginning of Section 3.2.

Example 3.4.1

Consider the functional relations given in Example 3.3.1.1. Applying Algorithm A1 gives only one key, ABDEHM, for the relation $\mathfrak{R}$ defined in Ω = ABCDEFGHKLM. But, the same example showed that $\{F_i\}$ can be partitioned into two equivalence classes that induced the partition $\Omega_1\Omega_2$ on Ω, where Ω_1 = ABCDEFG and Ω_2 = HKLM. Project $\mathfrak{R}$ into $S_1[\Omega_1]$ and $S_2[\Omega_2]$; the functional relations in S_1 are:

$$AB \rightarrow C;\ BD \rightarrow FG;\ AE \rightarrow C;$$

and the functional relations in S_2 are:

$$H \rightarrow KL;\ KM \rightarrow L.$$

Again, K_1 = ABDE is the only key of S_1, and K_2 = HM is the only key of S_2. Note that $K = K_1 \cup K_2$ = ABDEHM is the only key of $\mathfrak{R}$ defined in $\Omega = \Omega_1 \cup \Omega_2$.

3.5 Properties of Prime and Non-Prime Attributes

Recall the definitions of prime and non-prime attributes given in Chapter 1. It is essential to characterize the properties of these attributes for they play an important role in determining whether a relation is in second or third normal form to be studied in Chapter 4. Also, they help characterize the keys which form the basis of the normal forms and they are essential in the functional reduction which is to be explained later. Again, let $\mathfrak{R}$ be the relation in Ω, and let $\{F_i\}$ be the connected functional relations in $\mathfrak{R}$ such that $\Omega = \bigcup(L_i \cup R_i)$.

Lemma 3.5.1

If an attribute A_k is not a member of any L_i, then there exists no functional relation $\ell \to r$ in $\mathfrak{R}$ such that $A_k \in \ell$ and $A_k \notin r$ unless there exists an $\ell' \subseteq \ell$ such that $A_k \notin \ell'$ and $\ell' \to r$.

Proof:

If, for every i, $A_k \notin L_i$, then $A_k \in R_j$, for some j. It follows from properties P1 - P5 of functional relations that $A_k \in \ell$ only by applying the augmentation property or by applying additivity with $A_k \to A_k$. In either case, there must exist $\ell' \subseteq \ell$ such that $A_k \notin \ell'$ and $\ell' \to r$.

Theorem 3.5.1

If an attribute A_k is not a member of any L_i, then A_k is non-prime.

Proof:

Algorithm A1 showed that every key of $\mathfrak{R}$ is of the form $L_i \cup \alpha_i$, for some i, where $\alpha_i \subseteq T_i'$ and $|\alpha_i| \geq 0$.

If, for every i, $|\alpha_i| = 0$, then L_i is a key and $A_k \notin L_i$ by hypothesis; hence, A_k is non-prime.

If, for some i, $|\alpha_i| \neq 0$, then $L_i \cup \alpha_i \to \Omega$ and $L_i \not\to \Omega$. Therefore, there must exist a subset $Z_i \subseteq L_i \cup T_i \cup \alpha_i$ such that $Z_i = \alpha_i \cup [Z_i \cap (L_i \cup T_i)]$ and $Z_i \to T_i' - \alpha_i$. Also $\alpha_i = Z_i \cap T_i'$, and α_i is the minimal subset of T_i' such that $Z_i \to T_i' - \alpha_i$. But if $A_k \in \alpha_i$, then $A_k \in Z_i$ and $A_k \notin (T_i' - \alpha_i)$ since Z_i and $T_i' - \alpha_i$ are disjoint, and so $Z_i \to T_i' - \alpha_i$ contradicts Lemma 3.5.1 whenever $A_k \in Z_i$ and

$A_k \notin T_i' - \alpha_i$. Therefore, $A_k \notin \alpha_i$, and so $A_k \notin L_i \cup \alpha_i$, which implies that A_k is non-prime.

Example 3.5.1

Example 3.4.1 showed that ABDE and HM are the only keys of S_1 and S_2 respectively. The attribute L is not a member of any L_i in S_2, and L is non-prime. Also, the attributes C, F and G are not members of any L_i in S_1, and they are also non-prime.

However, the attribute K is a member of L_2 in S_2 and yet K is non-prime. Therefore, a non-prime attribute may appear as a member of some L_i.

Corollary 3.5.1.1

If an attribute A_k is prime, then A_k is a member of some L_i.

Proof:

This statement is true because it is the contrapositive of the statement in Theorem 3.5.1 which is true.

Corollary 3.5.1.2

Every key K of $\Re$ is a subset of $\cup L_i$.

Proof:

By definition, every attribute A_k in a key K is prime, therefore, by Corollary 3.5.1.1, A_k is a member of some L_i and hence $K \subseteq \cup L_i$.

Lemma 3.5.2

If an attribute A_k is not a member of any R_i, then there exists no functional relation $\ell \rightarrow r$ in $\Re$ such that $A_k \notin \ell$ and $A_k \in r$.

Proof:

If, for every i, $A_k \notin R_i$, then $A_k \in L_j$, for some j. Only, by using the additivity property with $A_k \to A_k$, that one gets $A_k \in r$. But then $A_k \in \ell$. Therefore there exists no functional relation $\ell \to r$ in $\mathfrak{R}$ such that $A_k \notin \ell$ and $A_k \in r$.

Theorem 3.5.2

If an attribute A_k is not a member of any R_i, then A_k is a member of every key of $\mathfrak{R}$.

Proof:

Let $K = L_i \cup \alpha_i$ be a key of $\mathfrak{R}$ and suppose that $A_k \notin K$, then $K \to \Omega \to A_k$, by definition of a key and by projectivity. However, this contradicts Lemma 3.5.2 and therefore $A_k \in K$ for every key K of $\mathfrak{R}$.

Example 3.5.2

Let Ω = ABCDEFGHK, and let the functional relations in $\mathfrak{R}$ be as follows:

$$AB \to CD;\ ABE \to FG;\ CE \to HK;\ AH \to BFK.$$

The keys are: ABE, AEH and ACE. The attributes, A and E, belong to every key, and they are the only attributes in Ω that do not belong to any R_i.

Corollary 3.5.2.1

If an attribute A_k is not a member of any R_i, then A_k is prime.

Proof:

By Theorem 3.5.2, A_k is a member of every key of $\mathfrak{R}$, hence, by definition, A_k is prime.

Corollary 3.5.2.2

If an attribute A_k is non-prime, then A_k is a member of some R_i.

Proof:

This statement is true because it is the contrapositive of the statement in Corollary 3.5.2.1 which is true.

Corollary 3.5.2.3

If every attribute in $\cup R_i$ is prime, then every attribute in Ω is prime.

Proof:

Let $\Omega_1 = \cup R_i$, and let $\Omega_2 = \Omega - \Omega_1$. Every attribute A_k in Ω_2 is not a member in Ω_1 and so, by Corollary 3.5.2.1, every A_k is prime. But, $\Omega = \Omega_1 \cup \Omega_2$ and, by hypothesis, every attribute in Ω_1 is prime; hence, every member in Ω is prime.

Corollary 3.5.2.4

If every attribute in $\cup R_i$ is prime, then $\Omega = \cup L_i$.

Proof:

By Corollary 3.5.2.3, every attribute A_k in Ω is prime and so, by Corollary 3.5.1.1, every A_k is a member of some L_i, hence $\Omega \subseteq \cup L_i$. But, by definition of Ω, $\cup L_i \subseteq \Omega$. Therefore $\Omega = \cup L_i$.

Corollary 3.5.2.5

Let $\Omega_1 = \cup L_i$, and let $\Omega_2 = \cup R_i$. If $\Omega_1\Omega_2$ is a partition on Ω, then Ω_1 is the only key of $\mathfrak{R}$.

Proof:

Since $\Omega_1\Omega_2$ is a partition on Ω, then every attribute in

Ω_1 is not a member of any R_i and every attribute in Ω_2 is not a member of any L_i. But, by Theorem 3.5.1, every attribute in Ω_2 is non-prime and, by Theorem 3.5.2, every attribute in Ω_1 is a member of every key of $\mathfrak{R}$. Therefore, if K is a key of $\mathfrak{R}$, then $\Omega_1 \subseteq K$ but, by Corollary 3.5.1.2, $K \subseteq \Omega_1$ and so $K = \Omega_1$ is the only key of $\mathfrak{R}$.

Example 3.5.3

Refer back to relation S_1 and its functional relations in Example 3.4.1. Let $\Omega_1 = \cup L_i = ABDE$, and let $\Omega_2 = \cup R_i = CFG$, then $\Omega_1\Omega_2$ is a partition on $\Omega = \Omega_1 \cup \Omega_2$ and $\Omega_1 = ABDE$ is the only key of S_1.

Theorems 3.5.1 - 3.5.2 and their corollaries show that the keys are subsets of the attributes, $\cup L_i$, that appear only on the left side of the original functional relations $L_i \rightarrow R_i$. This is a fundamental result and it is used in the following section to show that the columns -- in the implication matrix -- corresponding to the attributes that belong to $(\cup R_i) - (\cup L_i)$ can be deleted leaving exactly the same set of keys.

3.6 Functional Reduction

Let $\mathfrak{R}$ be a relation in Ω, and let $\{F_i : L_i \rightarrow R_i\}$ be the connected functional relations in $\mathfrak{R}$ such that $\Omega = \cup(L_i \cup R_i)$.

Let $\Omega_1 = \cup L_i$, and let Ω_2 be the set of attributes in Ω that are not members of any L_i, so that $\Omega_1\Omega_2$ is a partition on Ω. Project $\mathfrak{R}$ into the subrelation $S_1[\Omega_1]$. The functional relations in S_1, called reduced functional relations, are derived from the given functional relations in $\mathfrak{R}$ by deleting the attributes in Ω_2. Note that, for every i, the functional relations in S_1

are of the form $L_i \to R_i'$ such that $R_i' \subseteq R_i$ whenever $\Omega_2 \neq \phi$ and $R_i' = R_i$ only when $\Omega_2 = \phi$, and so, by projectivity, the reduced functional relations in S_1 still hold true.

The following theorem is a fundamental result which shows that the keys of $\mathfrak{R}$ depend on the functional relations in $\mathfrak{R}$ and on Ω_1, while, in Algorithm A1 in Chapter 2, all the attributes in Ω were used to find the keys.

Theorem 3.6.1

Let $K \subseteq \Omega$, then K is a key of $\mathfrak{R}$ if and only if K is a key of S_1, the projection of $\mathfrak{R}$ on $\cup L_i$.

Proof:

Necessary condition: If K is a key of $\mathfrak{R}$, then, by definition, $K \to \Omega$ and, by projectivity, $K \to \Omega_1$. Moreover, since K is a key of $\mathfrak{R}$, then K is a minimal subset in Ω such that $K \to \Omega$. But, by Corollary 3.5.1.2, K is a subset of Ω_1, and so K is a minimal subset in Ω_1 such that $K \to \Omega$. Therefore, it remains to show that K is a minimal subset in Ω_1 such that $K \to \Omega_1$. Let $K_1 \subset K$, and suppose $K_1 \to \Omega_1$. But, by additivity, $\Omega_1 = \cup L_i \to \cup R_i$ and so, by transitivity, $K_1 \to \cup R_i$ and, by additivity, $K_1 \to (\cup L_i) \cup (\cup R_i) = \Omega$. However, $K_1 \to \Omega$ contradicts the hypothesis that K is a key. Therefore K is also a key of S_1.

Sufficient condition: If K is a key of S_1, then $K \to \Omega_1$, and $K \to \Omega_1 \to \cup R_i$ by definition and transitivity respectively. Therefore $K \to (\cup L_i) \cup (\cup R_i) = \Omega$ by additivity. If there exists a proper subset $K_1 \subset K$ such that $K_1 \to \Omega$, then $K_1 \to \Omega_1$ by projectivity, which contradicts the hypothesis that

K is a key of S_1. Therefore, K is a key of $\mathfrak{R}$.

Example 3.6.1

Refer back to Example 3.5.2. The keys of $\mathfrak{R}$ are:

ABE, AEH and ACE.

Apply the method as proposed in Theorem 3.6.1. Project the relation $\mathfrak{R}$ in Ω into $S_1[\Omega_1]$ where $\Omega_1 = \cup L_i$ = ABCEH and $\Omega_2 = \Omega - \Omega_1$ = DFGK. The reduced functional relations in S_1 are:

AB → C; ABE → ABE (by reflexivity); CE → H; AH → B.

The transitive closure, P^*, of the implication matrix, P, in S_1 is:

$$P^* = \begin{array}{c|ccccc} & A & B & C & E & H \\ \hline AB & 1 & 1 & 1 & 0 & 0 \\ ABE & 1 & 1 & \underline{1} & 1 & \underline{1} \\ CE & 0 & 0 & 1 & 1 & 1 \\ AH & 1 & 1 & \underline{1} & 0 & 1 \end{array}$$

Again, the keys of S_1 are: ABE, AEH, and ACE which are exactly the same keys of $\mathfrak{R}$.

3.7 Functional Deletion

Note that in Example 3.6.1, the functional relation ABE → ABE holds by reflexivity. The following definition, property, and lemmas will show that one can delete a functional relation of the form $L_i \to L_i$ from $\{F_i\}$ without changing the set of keys.

Definition 3.7.1

A functional relation F_i of the form $L_i \to L_i$ is called a reflexive form. Otherwise, it is called a non-reflexive form.

Property 3.7.1

If $\ell \rightarrow r$ is a derived functional relation from the given $\{F_i\}$, then $\ell \supseteq L_i$ for some i, since the application of properties P1 - P5 will either unalter or augment L_i. Therefore $\ell \supseteq L_i$.

Lemma 3.7.1

If F_i is a reflexive form, and if there exists at least one $j \neq i$ such that $L_j \subset L_i$ in the original set of L_i, then any key that can be derived from L_i, can also be derived from L_j.

Proof:

Let $L_{ij} = L_i - L_j$. If $L_i \cup \beta_i \rightarrow \Omega$, then $(L_j \cup L_{ij}) \cup \beta_i = L_j \cup (L_{ij} \cup \beta_i) = L_j \cup \beta_j \rightarrow \Omega$. So, there is always at least one $\beta_j \subseteq L_i \cup \beta_i$, and hence $L_j \cup \beta_j \subseteq L_i \cup \beta_i$, such that $L_j \cup \beta_j \rightarrow \Omega$. Therefore, $L_i \cup \beta_i$ can never be a key whenever $L_j \cup \beta_j$ is unless $L_i \cup \beta_i = L_j \cup \beta_j$.

Lemma 3.7.2

If F_i is a reflexive form, and if for every $j \neq i$, $L_j \not\subset L_i$ in the original set of L_i, then any key that can be derived from L_i can also be derived from L_k, for some $k \neq i$.

Proof:

If, for every $j \neq i$, $L_j \not\subset L_i$ in the original set of L_i, then, in P^*, $L_i \rightarrow L_i$ only. So, $T_i = \emptyset$, and $T_i' = \Omega - L_i$ since, by definition, $L_i \cap T_i = \emptyset$. Hence, if $L_i \cup \beta_i \rightarrow \Omega$, for $\beta_i \subseteq T_i'$, then there must exist $Z_i \subseteq L_i \cup \beta_i$ such that $Z_i \rightarrow T_i'$, and so, by Property 3.7.1, $Z_i = L_k \cup \gamma_k$, for some

$k \neq i$, and $L_k \cup \gamma_k \to T_i'$. But, since $L_k \subseteq L_i \cup \beta_i$ and $L_i \cup \beta_i \to \Omega$, then there always exists $\beta_k \subseteq L_i \cup \beta_i$ such that $L_k \cup \beta_k \subseteq L_i \cup \beta_i$ and $L_k \cup \beta_k \to \Omega$. Therefore, $L_i \cup \beta_i$ can never be a key whenever $L_k \cup \beta_k$ is, unless $L_i \cup \beta_i = L_k \cup \beta_k$.

Theorem 3.7.1

Any functional relation in a reflexive form can be deleted from $\{F_i\}$ without changing the keys of a relation defined in Ω.

Proof:

If F_i is a reflexive form, then, by Lemmas 3.7.1 and 3.7.2, every key that can be derived from L_i can also be derived from L_j, for some $j \neq i$. Therefore, one needs only show that if, for $j \neq i$, $L_j \cup \alpha_j$ is a key, then F_i need not be used to find α_j

If $L_j \cup \beta_j \to \Omega$, $\beta_j \supseteq \alpha_j$, then there must exist $Z_j \subseteq L_j \cup T_j \cup \beta_j$ such that $Z_j \to T_j'$. Hence, if $Z_j = L_i \cup \gamma_i$, then, for $\gamma_i' \subseteq L_j \cup T_j$, $L_i \cup \gamma_i \cup \gamma_i' = L_i \cup \beta_i \to \Omega \to T_j'$ such that $\beta_j = T_j' \cap (L_i \cup \gamma_i) = T_j' \cap (L_i \cup \beta_i)$, and so, by Lemmas 3.7.1 and 3.7.2, there always exists $L_k \cup \beta_k \subseteq L_i \cup \beta_i$, for some $k \neq i$, such that $L_k \cup \beta_k \to \Omega \to T_j'$. Hence, if $L_j \cup \alpha_j$ is a key, then, by Algorithm A1, $L_i \cup \beta_i$ need not be used since it is a superset. Therefore, F_i can be deleted from the original set of functional relations without changing the keys of a relation defined in Ω.

3.7.1 Examples

Example 3.7.1.1

Consider the reduced functional relations in Example 3.6.1. Deleting the functional relation $ABE \to ABE$ gives the new set of reduced functional relations:

$$AB \to C;\ CE \to H;\ AH \to B\ .$$

Again, the keys are: ABE, AEH, and ACE.

Example 3.7.1.2

Refer back to relation S_1 and its functional relations in Example 3.4.1. Projecting S_1 into $S_1'[ABDE]$ gives the following reduced functional relations in S_1':

$$AB \to AB;\ BD \to BD;\ AE \to AE\ .$$

All the reduced functional relations are reflexive forms; so, they can be deleted. Note that $\Omega_1' = ABDE$ in S_1' and there are no non-reflexive form functional relations in S_1', so, by Corollary 3.4.1.2, Ω_1' is the only key of S_1'. This justifies the answers in Examples 3.4.1 and 3.5.3.

Example 3.7.1.3

Refer back to relation S_2 and its functional relations in Example 3.4.1. It was shown that $K_2 = HM$ is the only key of S_2. Projecting S_2 into $S_2'[HKM]$ gives the following reduced functional relations in S_2':

$$F_1\colon\ H \to K\ ;\quad F_2\colon\ KM \to KM.$$

Note that F_2 is a reflexive form and so it can be deleted. Therefore, F_1 is the only functional relation left in S_2' and so HM is the only key of S_2', and hence of S_2.

Observe that when F_2: KM → KM in S_2' is deleted, the keys must still be found in Ω_2' = HKM, although not all attributes in Ω_2' appear in the reduced functional relations after the reflexive forms have been deleted.

3.8 Chapter Summary and Remarks

This chapter dealt mainly with computational savings. It was shown that if $\{F_i: L_i \to R_i\}$ is the set of given functional relations in $\mathfrak{R}$, then the first step for finding the keys is the partition of $\{F_i\}$ into classes of connected functional relations.

Moreover, it was shown that the keys depend only on the functional relations and the attributes in $\cup L_i$. Extensive computational savings can be achieved whenever $\cup L_i \subset \Omega$, where Ω is the set of all attributes in $\mathfrak{R}$. Also, further savings are possible whenever a functional relation is in a reflexive form.

Finally, since the keys of $\mathfrak{R}$ are determined from subrelations of $\mathfrak{R}$, defined on a partition of Ω, then adding or deleting attributes to a subset of this partition will only affect the keys of the corresponding subrelation, while the keys of the other subrelations remain unchanged. Therefore, the concepts of functional partition, reduction, and deletion help provide an answer to the question of the effect on the keys of adding or deleting attributes in $\mathfrak{R}$.

CHAPTER 4

NORMAL FORMS

4.1 Introduction

Chapter 4 deals mainly with relations in second and third normal forms. The definition of the relational model does not specify procedures for determining whether a normalized relation is in second or third normal form. This chapter provides such procedures, as a result of analyzing in detail the mathematical properties of a relation in second and third normal forms.

4.2 Properties of the Second Normal Form (SNF)

This section characterizes the mathematical properties of a relation in SNF for the purpose of gaining insight into ways of determining whether a relation is in SNF.

4.2.1 Mathematical Properties

Let $\mathfrak{R}$ be a relation in Ω. The functional relations, $F_i: L_i \to R_i$, $i = 1,2,\ldots,m$, in $\mathfrak{R}$ are all connected and $\Omega = \cup(L_i \cup R_i)$. Let $\Omega_1 = \cup L_i$, and let $\Omega_2 = \cup R_i$. It is assumed that $\{F_i\}$ has more than one element. Let P_Ω and Q_Ω be the set of prime and non-prime attributes in $\mathfrak{R}$ respectively. Also, let p and q be arbitrary elements in P_Ω and Q_Ω respectively.

If every attribute in Ω_2 is prime, then $\mathfrak{R}$ is in SNF, because, by Corollary 3.5.2.3, $P_\Omega = \Omega$; hence, $Q_\Omega = \phi$. However, SNF's exist for which $Q_\Omega \neq \phi$.

Further, it is easily shown that, if K is a key, then no non-empty proper subset of K is dependent on any other distinct non-empty proper subset of K, for otherwise a proper subset of K would imply Ω and K would not be a key. Consequently, if Ω_1 is the only key of $\mathfrak{R}$, then $Q_\Omega = \Omega_2 \neq \phi$, and so $\mathfrak{R}$ is not in SNF, because $L_i \rightarrow R_i$, for $i = 1,2,\ldots,m$, are given where $L_i \subset \Omega_1$ and $R_i \subseteq Q_\Omega$.

The preceding statements suggest the following two important theorems.

Theorem 4.2.1.1

If $\Omega_1\Omega_2$ is a partition on Ω, then $\mathfrak{R}$ is not in SNF.

Proof:

By Corollary 3.5.2.5, Ω_1 is the only key of $\mathfrak{R}$, and so $\mathfrak{R}$ is not in SNF.

Theorem 4.2.1.2

If, for some i, $L_i \cup \alpha_i$, $|\alpha_i| \geq 1$, is the only key of $\mathfrak{R}$, and if $Q_\Omega \neq \phi$, then $\mathfrak{R}$ is not in SNF.

Proof:

$P_\Omega = L_i \cup \alpha_i$, $L_i \subset L_i \cup \alpha_i$, and $L_i \rightarrow R_i$ is given. If $R_i \subseteq L_i \cup \alpha_i$, then $L_i \rightarrow R_i$ would imply that a proper subset of a key is dependent on another distinct proper subset of the same key, which is impossible. Therefore $R_i \subseteq Q_\Omega$, and so $\mathfrak{R}$ is not in SNF.

As a consequence to the definition of a relation in SNF, the following property is true.

Property 4.2.1.1

If every key of $\mathfrak{R}$ consists of only one attribute, then

$\mathfrak{R}$ is in SNF, for none of the keys has a non-empty proper subset upon which a non-prime attribute could be dependent.

Algorithm A1 showed that every key is of the form $L_i \cup \alpha_i$, $|\alpha_i| \geq 0$. So, if $L_i \cup \alpha_i$ is a key, and if, for $j \neq i$, there exists an $L_j \subset L_i$ in the original set of L_i, then $L_j \cup \alpha_j = L_i \cup \alpha_i$ is also the same key. For the remainder of this chapter, the following notation is used. If $L_j \cup \alpha_j$ is a key, then, for every $i \neq j$, $L_i \not\subset L_j$ in the original set of L_i.

Lemma 4.2.1.1

If $L_i \rightarrow R_i$, and if there exists no $L_j \subset L_i$ in the original set of L_i, then, for any $\ell_1 \subset L_i$ and $\ell_2 \subseteq \Omega$, $\ell_1 \not\rightarrow \ell_2$.

Proof:

If $\ell_1 \rightarrow \ell_2$, then by Property 3.7.1, $\ell_1 \supseteq L_k$, for some k, and so $L_k \subseteq \ell_1 \subset L_i$, which contradicts the hypothesis that there exists no $L_k \subset L_i$ in the original set of L_i. Therefore $\ell_1 \not\rightarrow \ell_2$.

Consequently, the following theorems lead to special cases in the algorithm for determining whether a relation is in SNF.

Theorem 4.2.1.3

If every key of $\mathfrak{R}$ is one of the original L_i, then $\mathfrak{R}$ is in SNF.

Proof:

If L_i is a key, then, by the construction of L_i and by Lemma 4.2.1.1, for any $\ell_1 \subset L_i$ and $\ell_2 \subseteq \Omega$, $\ell_1 \not\rightarrow \ell_2$. So, if $q \in Q_\Omega$, then choose $\ell_2 = q$, and so $\ell_1 \not\rightarrow q$. Therefore, $\mathfrak{R}$ is in SNF.

Theorem 4.2.1.4

If every key of $\Re$ is of the form $L_i \cup \alpha_i$, $|\alpha_i| \leq 1$, and if, for every j such that $L_j \subset L_i \cup \alpha_i$, $L_j \not\to q$, then $\Re$ is in SNF.

Proof:

If $\ell \subset L_i \cup \alpha_i$, and if $\ell \to r$ is a derived functional relation from the given set of functional relations, then, by Property 3.7.1, $\ell \supseteq L_k$, for some k, and $L_k \subseteq \ell \subset L_i \cup \alpha_i$. But $L_i \cup \alpha_i$ is a key and so, by the construction of a key, $L_k \not\subset L_i$, hence $\ell \not\subset L_i$. Therefore, $\ell = L_i$ or $\ell = L_k$, for some $k \neq i$, and so, if $q \in Q_\Omega$ and $q \notin r$, then $\Re$ is in SNF.

The following theorem and corollary are fundamental in determining whether a relation is in SNF.

Theorem 4.2.1.5

To determine whether a relation, $\Re$, is in SNF, one needs only consider the functional relations, $L_i \to T_i$, after forming the transitive closure of the implication matrix P, such that $L_i \subseteq P_\Omega$.

Proof:

If R is not in SNF, then there exists at least one key, say $L_i \cup \alpha_i$, such that $\ell \subset L_i \cup \alpha_i$ and $\ell \to q$, where $q \in Q_\Omega$. So, by Property 3.7.1, $\ell = L_j \cup \beta_j$, for some j. But $\ell \subseteq P_\Omega$, since ℓ is a subset of a key, and so $L_j \subseteq \ell \subseteq P_\Omega$.

Let t_j be the set of all attributes in Ω such that $\ell = L_j \cup \beta_j \to t_j$. To find t_j, the transitive closure is formed on $L_j \cup \beta_j$ and the original set of functional relations in $\mathfrak{R}$. Denote the transitive closure by the successive steps, $L_j \cup \beta_j \to t_{1j}$, $L_j \cup \beta_j \to t_{2j}, \ldots, L_j \cup \beta_j \to t_{nj} = t_j$. If, for some k, $L_k \not\subseteq P_\Omega$, and if, for some i, $L_k \subseteq L_j \cup \beta_j \cup t_{ij}$, then $L_k \not\subseteq L_j \cup \beta_j$, since $L_j \cup \beta_j \subseteq P_\Omega$, and so $t_{ij} \cap L_k \neq \phi$; hence, there must exist $q \in Q_\Omega$ such that $q \in (t_{ij} \cap L_k)$. Therefore, if $L_k \not\subseteq P_\Omega$, then, for some i, $L_k \not\subseteq L_j \cup \beta_j \cup t_{ij}$, unless $\ell = L_j \cup \beta_j \to q$, but then $\mathfrak{R}$ would not be in SNF, since $\ell \to q$, and so L_k is not needed.

The following corollary provides a sufficient condition for a relation to be in SNF. However, Example 4.3.2.1 will show a relation which is in SNF but which does not satisfy this sufficient condition.

Corollary 4.2.1.5.1

Let K be an arbitrary key of $\mathfrak{R}$. If, for every i such that $L_i \subseteq P_\Omega$ and $L_i \neq K$, $L_i \not\to q$, then $\mathfrak{R}$ is in SNF.

Proof:

To determine whether a relation, $\mathfrak{R}$, is in SNF, then by Theorem 4.2.1.5, one considers only the functional relations, $L_i \to T_i$ in P^*, such that $L_i \subseteq P_\Omega$. By the hypothesis of the corollary, if $L_i \subseteq P_\Omega$, then $L_i \not\to q$ unless L_i is a key. But, if L_i is a key, then, by the construction of a key, for every $j \neq i$, $L_j \not\subset L_i$. Also, for $i \neq k$, if L_i and $L_k \cup \alpha_k$, $|\alpha_k| \geq 1$, are keys of $\mathfrak{R}$, and if $\ell = L_k \cup \beta_k$, for $\beta_k \subset \alpha_k$, then $L_i \not\subseteq L_k \cup T_k \cup \beta_k$, for otherwise $L_k \cup \beta_k$ would imply Ω, since L_i is a key, and $L_k \cup \alpha_k$ would not be a key. Therefore, if $L_k \cup \beta_k \to q$, then, by taking the transitive closure on $L_k \cup \beta_k$ and $L_i \to T_i$ such that

$L_i \subseteq P_\Omega$, there must exist $L_j \subseteq P_\Omega$, for $j \neq k$ and $L_j \neq K$, such that $L_j \to q$. However, $L_j \to q$, for $L_j \neq K$, contradicts the hypothesis, and so $L_k \cup \beta_k \nrightarrow q$. Therefore $\Re$ is in SNF.

4.2.2 Examples and Remarks

This section provides examples and states remarks that are properties of relations in SNF. It is always assumed that $\Omega = \bigcup(L_i \cup R_i)$ where $L_i \to R_i$, $i = 1,2,\ldots,m$, are the given functional relations in $\Re$.

Example 4.2.2.1

$AB \to C$; $CE \to D$; $AD \to B$.

The keys are: $AB \cup E$; $AD \cup E$; $CE \cup A$. Therefore, $P_\Omega = \Omega$ and $Q_\Omega = \emptyset$. So, by definition, $\Re$ is in SNF.

Example 4.2.2.2

$AB \to CDE$; $AE \to BFG$.

The keys are: AB; AE. Therefore, $P_\Omega = ABE$ and $Q_\Omega = CDFG$. So, by Theorem 4.2.1.3, $\Re$ is in SNF.

The two previous examples show that if $\Re$ is in SNF, then Ω may or may not be equal to $\bigcup L_i$. Also, if $\Re$ is in SNF, then it is not necessarily true that every attribute in $\bigcup R_i$ is prime.

Example 4.2.2.3

$AB \to D$; $D \to A$; $C \to AD$.

The key is $C \cup B$. Therefore, $P_\Omega = BC$ and $Q_\Omega = AD$. So, by Theorem 4.2.1.4, $\Re$ is not in SNF.

In this example, $\Omega = \bigcup L_i$ and $\Re$ is not in SNF, while in Example 4.2.2.1, $\Omega = \bigcup L_i$ and $\Re$ was shown to be in SNF. Therefore, if $\Omega = \bigcup L_i$, then $\Re$ may or may not be in SNF.

Example 4.2.2.4

AB $\rightarrow$ CD; AC $\rightarrow$ BEF; DF $\rightarrow$ AE; BE $\rightarrow$ DC.

The keys are: AB; AC; BE $\cup$ F; DF $\cup$ B; DF $\cup$ C. Therefore, $P_\Omega = \Omega$ and $Q_\Omega = \phi$. So, by definition, $\Re$ is in SNF.

Example 4.2.2.5

AB $\rightarrow$ CD; AE $\rightarrow$ FG.

The key is: AB $\cup$ E or AE $\cup$ B. Therefore, P_Ω = ABE and Q_Ω = CDFG. So, by Theorem 4.2.1.4, $\Re$ is not in SNF.

By Examples 4.2.2.4 and 4.2.2.5, if every attribute in $\cup L_i$ is prime, then $\Re$ may or may not be in SNF.

Example 4.2.2.6

A $\rightarrow$ B; B $\rightarrow$ C.

The key is: A. Therefore, by Theorem 4.2.1.3, $\Re$ is in SNF.

By Examples 4.2.2.4 and 4.2.2.6, one sees that if $\Re$ is in SNF, then an attribute in $\cup L_i$ may or may not be prime.

4.3 Procedure for SNF

Section 4.3.1 presents an algorithm (Algorithm A2) which determines whether a relation is in SNF and Section 4.3.2 gives examples. Algorithm A2 uses the notation which is introduced next.

Let $\mathcal{L}_p = \{L_i | L_i \subseteq P_\Omega$ and $L_i \rightarrow R_i$ is a given functional relation$\}$, and let $\mathcal{L}_c = \{L_i | L_i \cup \alpha_i$ is a key and $|\alpha_i| \geq 1\}$ which implies that $\mathcal{L}_c \subseteq \mathcal{L}_p$. Also, let $|\mathcal{L}_p| = n_p$, and let $|\mathcal{L}_c| = n_c$. Consider $\{F_i: L_i \rightarrow R_i\}$ such that $L_i \in \mathcal{L}_p$, and renumber them, if necessary, $F_1, F_2, \ldots, F_{n_p}$, $n_p \leq m$ where m is the total number of elements in the original set, $\{F_i\}$, of functional relations in $\Re$. That is, if $L_k \notin \mathcal{L}_p$, then $n_p+1 \leq k \leq m$. Also, if $L_i \in \mathcal{L}_c$, and if $L_j \in (\mathcal{L}_p - \mathcal{L}_c)$, then $i < j$.

In Algorithm A2, the transitive closures are taken over $\{F_i\}$, $i = 1,2,\ldots,n_p$, except when forming P^*, they are taken over $\{F_i\}$, $i = 1,2,\ldots,m$, in $\Re$. Moreover, since more than one key, $L_i \cup \alpha_i$, might be derived from the same L_i, every statement in Algorithm A2 that includes α_i must be executed once for each distinct key, $L_i \cup \alpha_i$, derived from the same L_i.

4.3.1 Algorithm A2

SNF ← TRUE

Apply Algorithm A1 to get the keys then find Q_Ω, P_Ω, $\mathcal{L}_p$, and $\mathcal{L}_c$.

If $Q_\Omega = \emptyset \vee \mathcal{L}_c = \emptyset$ then terminate the algorithm.

While $i \leq n_c$ do

  If $L_i \rightarrow q \wedge q \in Q_\Omega$ then SNF ← FALSE, terminate the algorithm.

end

If for every i such that $L_i \cup \alpha_i$ is a key, $|\alpha_i| \leq 1$ then terminate the algorithm.

While $i \leq n_c$ do

  While $j \leq n_p$ do

    If $(j \neq i) \wedge (L_j \subset L_i \cup T_i \cup \alpha_i) \wedge (\alpha_{ij} = L_j \cap \alpha_i \neq \emptyset) \wedge (\alpha_{ij} \neq \alpha_i)$ then form the transitive closure on $L_i \cup \alpha_{ij}$.

      If $L_i \cup \alpha_{ij} \rightarrow q$ then SNF ← FALSE, terminate the algorithm.

  end

end

4.3.2 Examples

Example 4.3.2.1

$AB \to CD$; $CE \to AF$; $DF \to BE$; $BCF \to X$.

The keys are: $AB \cup E$; $AB \cup F$; $CE \cup B$; $CE \cup D$; $DF \cup A$; $DF \cup C$. Therefore, $P_\Omega = ABCDEF$, $Q_\Omega = X$, $\ell_p = \{AB,CE,DF,BCF\}$, and $\ell_c = \{AB,CE,DF\}$.

By Theorem 4.2.1.4, the relation is in SNF and this result can also be easily checked by Algorithm A2.

Example 4.3.2.2

$ABC \to DEG$; $AB \to CF$; $CD \to E$; $EG \to AC$; $BD \to F$.

The keys are: AB; $EG \cup B$; $CD \cup BG$. Therefore, $P_\Omega = ABCDEG$, $Q_\Omega = F$, $\ell_p = \{ABC,AB,CD,EG,BD\}$ and $\ell_c = \{CD,EG,BD\}$.

Note that $BD \in \ell_c$ and $BD \to F$, $F \in Q_\Omega$; hence, the relation is not in SNF.

Example 4.3.2.3

$AB \to CD$; $CEY \to AF$; $DFY \to BE$; $BCF \to X$.

The keys are: $AB \cup EY$; $AB \cup FY$; $CEY \cup B$; $CEY \cup D$; $DFY \cup A$; $DFY \cup C$. Therefore $P_\Omega = ABCDEFY$, $Q_\Omega = X$, $\ell_p = \{AB,CEY,DFY,BCF\}$, and $\ell_c = \{AB,CEY,DFY\}$.

Applying the last <u>do</u> statement in Algorithm A2 gives: $BCF \in \ell_p$, $BCF \subset AB \cup CD \cup FY$, $\alpha_{ij} = BCF \cap FY = F \neq \phi$, and $F \neq FY$.

Forming the transitive closure on $AB \cup F$ gives: $AB \cup F \to X$, $X \in Q_\Omega$. But $ABFY$ is a key and $ABF \subset ABFY$. Therefore, the relation is not in SNF.

4.4 Properties of the Third Normal Form (TNF)

Section 4.4.1 characterizes the mathematical properties of a relation in TNF, and Section 4.4.2 gives examples to support the theory; this leads to Algorithm A3 in Section 4.5 for determining whether a relation in SNF is also in TNF.

4.4.1 Mathematical Properties

This section begins by identifying a fundamental property of keys of a relation, whose consequence is a necessary and sufficient condition, in Theorem 4.4.1.2, for a relation which is in second normal form to also be in third normal form.

Theorem 4.4.1.1

Any two keys, K_1 and K_2, of $\mathfrak{R}$, are fully dependent on each other. i.e. $K_1 \Leftrightarrow K_2$.

Proof:

$K_1 \to K_2$ since K_1 is a key. If $K_1 \not\Rightarrow K_2$, then there must exist a proper subset $S_1 \subset K_1$ such that $S_1 \to K_2$, but then $S_1 \to K_2 \to \Omega$, by transitivity and because K_2 is a key. However, $S_1 \to \Omega$ contradicts the hypothesis that K_1 is a key. Therefore, $K_1 \Leftrightarrow K_2$.

Example 4.4.1.1

Refer back to Example 4.2.2.1. All the keys are fully dependent on each other. Also, by definition, a relation in TNF is also in SNF; so, every non-prime attribute is fully dependent on each key. $\mathfrak{R}$ is in TNF since $Q_\Omega = \phi$.

In this example, ABE and ADE are keys and $AD \to B$; the prime attribute B is not fully dependent on the key ADE. So, a prime attribute of a relation in TNF might not be fully dependent on each key. This proved to be a weakness in the definitions of SNF and TNF, which led Kent [48] to give alternative definitions for SNF and TNF. This point will be discussed in more detail in Chapter 5.

Example 4.4.1.2

$AB \to CD$; $BE \to AF$; $DE \to BC$; $BD \to A$.

The keys are: BE; DE. Therefore $P_\Omega = BDE$ and $Q_\Omega = ACF$. Note that $BE \Leftrightarrow DE$. By Theorem 4.2.1.3, $\mathfrak{R}$ is in SNF. Later, it can be shown that $\mathfrak{R}$ is also in TNF.

The thing to observe here is that BD is not disjoint from the keys BE and DE, yet $BD \to A$ and A is non-prime. This will prove to be a weakness in the definitions of TNF, as proposed by Codd and Kent; Chapter 5 analyzes this point more carefully.

The preceding theorem and examples, in this section, suggest the following important lemma.

Lemma 4.4.1.1

If $\mathfrak{R}$ is in TNF, then every non-prime attribute is dependent on itself, on the keys and on subsets that always have a non-empty intersection with the set of all keys of $\mathfrak{R}$.

Proof:

Let q_1 and q_2 be arbitrary non-prime attributes in Q_Ω. If K is a key, then, by definition, $K \to q_1$ and $K \to q_2$. But, $\mathfrak{R}$ is also in SNF; hence, $K \Rightarrow q_1$ and $K \Rightarrow q_2$. Also, $q_1 \to q_1$, by reflexivity. If $q_1 \to q_2$, then q_1, q_2 and K are disjoint and $K \to q_1 \to q_2$, by transitivity. However, this contradicts the hypothesis that $\mathfrak{R}$ is in TNF. Therefore $q_1 \not\to q_2$. Finally, Example 4.4.1.2 showed that there may exist a subset, S, that has a non-empty intersection with the set of all keys of $\mathfrak{R}$, such that $S \rightarrow q_1$ or $S \rightarrow q_2$. Any other dependence contradicts the definition of TNF.

Recall, in Chapter 1, the definition of a relation in TNF. The definition of transitive dependence allows a relation, $\mathfrak{R}$, in TNF to have multiple keys, because if K_1 and K_2 are any two disjoint keys of $\mathfrak{R}$, then, for any $q \in Q_\Omega$, $K_1 \rightarrow K_2 \rightarrow q$. But, $K_2 \rightarrow K_1$. So, q is non-transitively dependent on K_1. Consequently, the following important theorem gives necessary and sufficient conditions for a relation in SNF to also be in TNF.

Theorem 4.4.1.2

A relation, $\mathfrak{R}$, in SNF is in TNF if and only if, for any two disjoint non-empty proper subsets, S_1 and S_2 in Q_Ω, $S_1 \not\rightarrow S_2$ and $S_2 \not\rightarrow S_1$.

Proof:

Necessary condition: If $\mathfrak{R}$ is in TNF, then, by definition, every non-prime attribute in $\mathfrak{R}$ is non-transitively dependent on each key K of $\mathfrak{R}$. If $S_1 \rightarrow S_2$, then $K \rightarrow S_1 \rightarrow S_2$, by transitivity and the fact that K is a key. But K, S_1 and S_2 are mutually disjoint; so, $K \rightarrow S_1 \rightarrow S_2$ contradicts the hypothesis that $\mathfrak{R}$ is in TNF. Therefore, $S_1 \not\rightarrow S_2$ and similarly $S_2 \not\rightarrow S_1$.

Sufficient condition: If $S_1 \not\rightarrow S_2$, and if $S_2 \not\rightarrow S_1$, then, as in Lemma 4.4.1.1, S_1 and S_2 are only dependent on themselves, on keys, and on subsets that always have a non-empty intersection with the set of all keys of $\mathfrak{R}$. However, in all these cases, it is easily shown, by the definition of transitive dependence, that S_1 and S_2 are non-transitively dependent on each key of $\mathfrak{R}$. Therefore, $\mathfrak{R}$ is in TNF.

Corollary 4.4.1.2.1

If every attribute in $\cup R_i$ is prime, then $\mathfrak{R}$ is in TNF.

Proof:

If $\cup R_i \subseteq P_\Omega$, then, by Corollary 3.5.2.3, $P_\Omega = \Omega$ and $Q_\Omega = \emptyset$. Therefore, since there are no non-prime attributes in $\mathfrak{R}$, then, by definition, $\mathfrak{R}$ is in TNF.

Corollary 4.4.1.2.2

If $|Q_\Omega| = 1$, then a relation, $\mathfrak{R}$, in SNF is also in TNF.

Proof:

If $|Q_\Omega| = 1$, then there are no disjoint proper subsets in Q_Ω, and so, by Theorem 4.4.1.2, $\mathfrak{R}$ is in TNF.

The following theorem is very important; it is basic to the proof of the fundamental result in this section. The result is another set of necessary and sufficient conditions for a relation in SNF to also be in TNF.

Theorem 4.4.1.3

If $S \subseteq Q_\Omega$, then, for every $p \in P_\Omega$, $S \not\to p$.

Proof:

Suppose $S \to p$. But, p is prime; so, it is a member of at least one key, say, $L_i \cup \alpha_i$, $|\alpha_i| \geq 0$. Let $\ell = (L_i \cup \alpha_i) - p$, then $\ell \subset L_i \cup \alpha_i$ and ℓ is not a key, since $L_i \cup \alpha_i$ is. Hence, $S \cup \ell \to p \cup \ell = L_i \cup \alpha_i \to \Omega$, by additivity and transitivity respectively. But $S \cup \ell \to \Omega$, and $S \cup \ell$ is not a superset of any key, since $S \cap P_\Omega = \emptyset$. Therefore, $S \cup \ell$ is a key and so $S \subseteq P_\Omega$. But, this contradicts the hypothesis that $S \subseteq Q_\Omega$, and so $S \not\to p$.

The following result is fundamental and, as Section 4.5 will show, it simplifies greatly the algorithm for determining whether a relation in SNF is also in TNF.

Theorem 4.4.1.4

A relation, $\mathfrak{R}$, in SNF, is also in TNF if and only if, for every i, $L_i \cap P_\Omega \neq \emptyset$.

Proof:

Necessary condition: Suppose the relation, $\mathfrak{R}$, is in TNF. If, for some i, $L_i \cap P_\Omega = \emptyset$, then $L_i \subseteq Q_\Omega$; so, by Theorem 4.4.1.3, $L_i \not\rightarrow p$ for every $p \in P_\Omega$. Hence, for some i, $L_i \cup R_i \subseteq Q_\Omega$ such that $L_i \rightarrow R_i$ is one of the original functional relations in $\mathfrak{R}$, for otherwise $R_i \cap P_\Omega \neq \emptyset$; so, for some $p \in R_i \cap P_\Omega$, $L_i \rightarrow R_i \rightarrow p$ by projectivity and transitivity respectively. However, $L_i \rightarrow p$ contradicts Theorem 4.4.1.3, since $L_i \subseteq Q_\Omega$. Therefore, $R_i \subseteq Q_\Omega$; so, $L_i \cup R_i \subseteq Q_\Omega$. But, by Theorem 4.4.1.2, $\mathfrak{R}$ would not be in TNF, since $L_i \rightarrow R_i$, $L_i \cap R_i = \emptyset$, and $L_i \cup R_i \subseteq Q_\Omega$, which contradicts the hypothesis that $\mathfrak{R}$ is in TNF. Therefore, for every i, $L_i \cap P_\Omega \neq \emptyset$.

Sufficient condition: Suppose, for every i, $L_i \cap P_\Omega \neq \emptyset$. If $|Q_\Omega| \geq 2$, then let S_1 and S_2 be any two disjoint non-empty proper subsets in Q_Ω. If $S_1 \rightarrow S_2$, then, by Property 3.7.1, and for some i, $S_1 \supseteq L_i$. Hence, $L_i \subseteq S_1 \subseteq Q_\Omega$; so $L_i \cap P_\Omega = \emptyset$, since $P_\Omega \cap Q_\Omega = \emptyset$. But, $L_i \cap P_\Omega = \emptyset$ is a contradiction since, by hypothesis, $L_i \cap P_\Omega \neq \emptyset$ for every i. Therefore, $S_1 \not\rightarrow S_2$; so, by Theorem 4.4.1.2, $\mathfrak{R}$ is also in TNF.

The previous theorem fully characterizes a relation in TNF. However, it will be easier, in Algorithm A3 in Section 4.5, to use the contrapositive form of Theorem 4.4.1.4 to determine whether a relation in SNF is also in TNF. The following important corollary gives this contrapositive form.

Corollary 4.4.1.4.1

A relation, $\mathfrak{R}$, in SNF, is not in TNF if and only if, for some i, $L_i \subseteq Q_\Omega$.

Proof:

This statement is true because it is the contrapositive of the statement in Theorem 4.4.1.4 which is true.

4.4.2 Examples

In the following examples, each relation is in SNF. The problem is to determine those which are also in TNF.

Example 4.4.2.1

Refer back to Example 4.3.2.1. Note that $Q_\Omega = X$, and so $|Q_\Omega| = 1$. Therefore, by Corollary 4.4.1.2.2, $\mathfrak{R}$ is also in TNF.

Example 4.4.2.2

Refer back to Example 4.2.2.2. Note that $P_\Omega = ABE$ and $Q_\Omega = CDFG$. For every i, $L_i \cap P_\Omega \neq \phi$; so, by Theorem 4.4.1.4, $\mathfrak{R}$ is also in TNF.

Example 4.4.2.3

Refer back to Example 4.2.2.4. Note that $Q_\Omega = \phi$; so, by definition, $\mathfrak{R}$ is also in TNF.

Example 4.4.2.4

$AB \rightarrow CDE$; $AE \rightarrow BFG$; $CD \rightarrow FG$.

The keys are: AB, AE. Therefore, $P_\Omega = ABE$ and $Q_\Omega = CDFG$.

By Theorem 4.2.1.3, $\Re$ is in SNF. But, CD is one of the original L_i, and $CD \subseteq Q_\Omega$; therefore, by Corollary 4.4.1.4.1, $\Re$ is not in TNF.

The preceding theory and examples lead to the following simple algorithm which determines whether a relation, in SNF, is also in TNF.

4.5 Procedure for TNF

The following algorithm is a direct consequence of Theorem 4.4.1.4 and Corollary 4.4.1.4.1.

Algorithm A3

1. Apply Algorithm A2 to determine whether the relation is in SNF. Find P_Ω and Q_Ω.
2. If $\Re$ is not in SNF, then go to 5. Otherwise, go to 3.
3. If, for some i, $L_i \subseteq Q_\Omega$, then go to 5. Otherwise, go to 4.
4. The relation is in TNF. Stop.
5. The relation is not in TNF. Stop.

4.6 Chapter Summary and Remarks

Chapter 4 gives algorithms for determining whether a relation is in second or third normal form. The correctness of the algorithms, A2 and A3, derives from an extensive mathematical analysis preceding each algorithm. These algorithms are suitable for hand computation as well as computer implementation, as was Algorithm A1 in Chapter 2. However, none of these algorithms have been programmed.

It is clear that Algorithm A1 is the building block for the algorithms, A2 and A3, given in this chapter. This is expected since the keys, as found by Algorithm A1, form the basis of a relation in second or third normal form.

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 Conclusions

This thesis provides a detailed mathematical analysis of the basic concepts of the relational model, as originally proposed by Codd. Three algorithms are presented. Their correctness derives from the basic properties of the functional relations, as given by Armstrong [2], and the extensive theoretical background provided in this thesis. The concept of the implication matrix, as applied to the functional relations in Algorithm A1 to find the keys of a relation, has also proved to be fruitful in providing a useful mathematical theory and algorithms (A2 and A3 in Chapter 4) for determining whether a relation is in second or third normal form. Algorithms comparable to A2 and A3 do not exist anywhere in the literature. The mathematical properties of a relation in second or third normal form and the three suggested algorithms are the main contributions of this thesis. The data base administrator may use these results to determine whether a relation is in either of these normal forms.

Although this thesis provides three algorithms, no actual programming was undertaken because performance criteria are not central to the purpose of this thesis which is to develop and show the utility of the mathematical foundation.

Many problems remain to be solved in the relational model. This thesis concentrated on the normal forms, as proposed by Codd, only because no substitutes could be suggested before their full mathematical properties were well studied. Now that those properties have been examined in this thesis, an examination of the underlying definitions of normal forms will prove fruitful in determining whether the motivations for the second and third normal forms have been realized. This points to a new and rich area for further research.

5.2 Alternative Definitions for Normal Forms

As was stated in Chapter 1, the objective of the normal forms is to minimize the update, deletion, and insertion anomalies. For this reason, Codd introduced the concept of functional dependence and proposed the definitions of first, second, and third normal forms. The examples in Chapter 1 showed that a relation, $\mathfrak{R}$, stored in a first normal form, is highly redundant and inefficient to maintain. Moreover, projecting a relation, in first normal form, into a collection of relations in second or third normal form reduces the undesirable anomalies mentioned by Codd.

A relation is a collection of facts, and, roughly speaking, a dependence, $A \rightarrow B$, corresponds to a fact. The basic motivation behind second normal form is that a fact stored in a relation should be dependent on the whole key for the relation. That is, if $A \to B$, then A and B should not be stored in the same relation, whenever A is a proper subset of a key, otherwise this fact is stored more than once and if B should change, then this update should be done in more than one place. However, the second normal form, as defined by Codd, does not totally meet this objective, as can easily be seen from the following example.

Example 5.2.1

$AB \to CD$; $CD \to AB$; $A \to C$.

The keys are: AB, CD, and $A \cup D$. Therefore, $P_\Omega = ABCD$ and $Q_\Omega = \emptyset$. Since $Q_\Omega = \emptyset$, the relation is in SNF. The relation, $\mathfrak{R}$, might look like this:

$\mathfrak{R}$:

A	B	C	D
1	1	1	1
1	2	1	2
1	3	1	3
1	4	1	4

Suppose now, one wishes to update the fact, $A \to C$ by changing the values of A or C. It is clear that, as the value of C changes, then this change occurs in more than one tuple.

The definition of a relation, $\mathfrak{R}$, in SNF, as proposed by Codd, is restricted to non-prime attributes to be fully dependent on each key of $\mathfrak{R}$. But, as was shown in Example 5.2.1, C is prime and C is not fully dependent on the key AB since $A \to C$. This shortcoming led Kent [48] to suggest the following alternative definition of a relation in SNF; it will be referred to as KSNF.

Definition 5.2.1

A relation in first normal form is in KSNF if every attribute in the complement of a key is fully dependent on that key.

It is important to note that, by definition, a relation in KSNF is also in SNF. Based on this alternative definition, it is easy to see that the relation, $\mathfrak{R}$, in Example 5.2.1, is not in KSNF since $A \to C$. Project $\mathfrak{R}$ into $\mathfrak{R}_1[AC]$ and $\mathfrak{R}_2[ABD]$. It is clear that $\mathfrak{R}_1$ and $\mathfrak{R}_2$ are in KSNF, and so in SNF. Note that the fact, $A \to C$, is stored only once in the relation $\mathfrak{R}_1$. Therefore, the alternative definition, as suggested by Kent, is an improvement, in terms of update minimization, over Codd's definition of a relation in SNF.

The second normal form, as proposed by Codd, is only the first step in eliminating certain redundancies in information storage. Other dependencies may exist in a relation, and Codd proposed the third normal form to eliminate information about an entity which can be derived from other attributes of the entity. Example 1.3.3 already elaborated on this concept. Moreover, the third normal form is restricted to the second normal form and to non-prime attributes to be non-transitively dependent on each key of the relation. However, Example 5.2.1 gave the reasons for Kent's alternative definition for the second normal form; so, for similar reasons, Kent suggested the following alternative definition for the third normal form. It will be referred to as KTNF.

Definition 5.2.2

A relation in KSNF is in KTNF, if every attribute in the complement of a key is non-transitively dependent on that key.

Again, it is important to note that a relation in KTNF is in TNF, but the converse may not be true.

It is clear, from the definition of KTNF, that every attribute in the complement of a key is fully dependent on itself and on that key. But, Lemma 4.4.1 showed that an attribute may also be dependent on subsets that always have a non-empty intersection with the set of all keys of the relation. Therefore, although the conditions for a relation to be in KTNF are more stringent than those of a relation in TNF, it is still possible to have facts stored more than once in a relation, as the following example will show.

Example 5.2.2

$AB \rightarrow CD$; $AC \rightarrow D$

The key is: AB. Therefore, $P_\Omega = AB$ and $Q_\Omega = CD$. By Theorem 4.2.1.3, the relation is in SNF, and, by Theorem 4.4.1.4, it is also in TNF. Moreover, by definitions, the relation is also in KSNF and KTNF. The relation, $\mathfrak{R}$, might look like this:

$\mathfrak{R}$:

A	B	C	D
1	1	1	1
1	2	1	1
2	1	1	2
1	3	1	1

Suppose now, one wishes to update the fact $AC \rightarrow D$. It is clear that as the value of D changes, then this change occurs in more than one tuple.

It is important to note that the fact, $AB \rightarrow D$, is a consequence, by transitivity, of the facts $AB \rightarrow C$ and $AC \rightarrow D$. But the objective of the third normal forms, TNF and KTNF, is not to store information like $AB \to D$ in the above example. Therefore, Example 5.2.2 points to a weakness in the statements of the third normal forms, as proposed by Codd and Kent.

The ultimate goal, in the relational model, is to have a relation in TNF or KTNF; so, this thesis proposes only one alternative definition that is more restrictive than the definitions of Codd and Kent. The basic motivation for the following alternative definition has been explained in Example 5.2.2.

Definition 5.2.3

A relation, $\mathfrak{R}$, in first normal form is in canonical normal form (CNF) if, for any two distinct, but not necessarily disjoint, non-empty subsets, S_1 and S_2, in the set of all attributes in $\mathfrak{R}$, $S_1 \to S_2$, then S_1 is a key of $\mathfrak{R}$.

It follows, from the definitions, that a relation in CNF is also in KTNF, and so in TNF. Also, note that the relation, $\mathfrak{R}$, in Example 5.2.2, is not in CNF since $AC \to D$ and AC is not a key of $\mathfrak{R}$. Project $\mathfrak{R}$ into $\mathfrak{R}_1[ABC]$ and $\mathfrak{R}_2[ACD]$. Note that $\mathfrak{R}_1$ and $\mathfrak{R}_2$ are in CNF. Moreover, $\mathfrak{R}$ is the natural join of $\mathfrak{R}_1$ and $\mathfrak{R}_2$, that is $\mathfrak{R}[ABCD] = \mathfrak{R}_1[ABC] * \mathfrak{R}_2[ACD]$, because $AC \to D$ is the sufficient condition for $\mathfrak{R} = \mathfrak{R}_1 * \mathfrak{R}_2$ to be true; so, no essential information is lost, because the original functional relations in $\mathfrak{R}$ are preserved. Since $AB \to C$ holds in $\mathfrak{R}_1$, and $AC \to D$ holds in $\mathfrak{R}_2$, by transitivity, $AB \to D$ holds in $\mathfrak{R}$. Therefore, the definition of CNF, as suggested previously, is an improvement, in terms of storage requirements, over the definitions of Codd and Kent.

Further, the following example should help explain the objectives of the suggested canonical normal form.

Example 5.2.3

$AB \to D$; $BC \to A$; $CD \to B$; $BD \to A$.

The keys are: BC, CD. Therefore $P_\Omega = BCD$ and $Q_\Omega = A$. It is easy to show, by definition, that the relation is in KTNF, and so in TNF. The relation, $\mathfrak{R}$, might look like this:

$\mathfrak{R}$:	A	B	C	D
	1	1	1	1
	1	1	2	1
	1	1	3	1
	1	2	4	2

The fact, $BC \to A$, is a consequence, by transitivity, of the facts $BC \to D$, since BC is a key of $\mathfrak{R}$, and $BD \to A$. But since $\mathfrak{R}$ is in KTNF, the objective of Kent's alternative definition has not been completely met. However, $\mathfrak{R}$ is not in CNF, because $AB \to D$ and $BD \to A$ but neither AB nor BD is a key of $\mathfrak{R}$.

Project $\mathfrak{R}$ into $\mathfrak{R}_1[ABD]$ and $\mathfrak{R}_2[BCD]$. The functional relations in $\mathfrak{R}_1$ are: $AB \to D$ and $BD \to A$, while in $\mathfrak{R}_2$, they are: $BC \to D$ and $CD \to B$. Therefore, $\mathfrak{R}_1$ and $\mathfrak{R}_2$ are in CNF. Moreover $\mathfrak{R}[ABCD] = \mathfrak{R}_1[ABD] * \mathfrak{R}_2[BCD]$ and the original functional relations in $\mathfrak{R}$ are preserved; so no essential information has been lost.

Examples 5.2.2 and 5.2.3 showed that the canonical normal form , as suggested in this thesis, meets better the original objectives of TNF and KTNF.

Codd's SNF required each non-prime attribute to be fully dependent on each key, but allowed a prime attribute not to have such full dependence; hence, update anomalies still could exist. Kent's KSNF attempted to remedy that problem by requiring every attribute in the complement of a key to be fully dependent on that key. However, Chapter 4 showed that a subset, S, of attributes might have a non-empty intersection with all keys of a relation in KSNF, and it still might be that $S \to A$, where A is some prime or non-prime attribute, in which case update anomalies would still exist. The canonical normal form defined here removes the update anomalies in all these cases, and leaves open the question of an appropriate definition of "optimality" when applied to normal forms, a question which might instigate substantial further research.

5.3 Suggestions for Future Research

This thesis did not deal with the concept of optimization in the normal forms. Codd [22] and Kent [48] motivated the subject, but their definitions of an optimal normal form lack general and different criteria for possible comparison. It appears that the extensive mathematical theory and the algorithms presented in this thesis can be extended, with some variations, to deal with the optimal normal forms.

Although this thesis provides three algorithms, no actual programming was undertaken so performance criteria were not set. The algorithms should be tested on real data bases and their performance and complexity evaluated.

Codd [25] and Strnad [68] define a collection of operations on relations, and this collection is called the relational algebra. These operations are not necessarily binary. They include the traditional set operations (cartesian product, union, intersection, etc.) and new operations on relations, suitable only in the relational model, such as projection, join, division, and restriction. The user's queries can be expressed in the relational algebra by forming a new normalized relation from the existing collection of relations. The investigation of the mathematical properties of these operations is strongly suggested.

Finally, this thesis dealt only with the logical structure of a relational data base, and no mention was made of the storage structure. Rissanen, et. al [57] analyze this problem in the form of examples so a much more rigorous and general mathematical approach is needed in this area.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. ALTMAN, E.B., et. al (1973)
"Specifications in a Data Independent Accessing Model", IBM Research Report, RJ 1141, IBM Research Laboratory, San Jose, California.

2. ARMSTRONG, W.W. (1974)
"Dependency Structures of Data Base Relationships", Information Processing 74, North Holland Publishing Company, pp. 580-583.

3. ASTRAHAN, M.M., et. al (1972)
"Concepts of a Data Independent Accessing Model", IBM Research Report, RJ 1105, IBM Research Laboratory, San Jose, California.

4. BACHMAN, C.W. (1972)
"The Evolution of Storage Structures", CACM, Vol. 15, No. 7, pp. 628-634.

5. BACHMAN, C.W. (1973)
"The Programmer as Navigator", CACM, Vol. 16, No. 11, pp. 653-658.

6. BELL SYSTEM TECHNICAL JOURNAL (1973)
"Information Management System Issue", Vol. 52, No. 10, pp. 1681-1764.

7. BIRKHOFF, G. (1967)
"Lattice Theory", 3rd Edition, American Mathematical Society Colloquium, Publ. XXV, Providence, R.I.

8. BJORNER, D., et. al (1973)
"The Gamma Zero N-ary Relational Data Base Interface: Specifications of Objects and Operations", IBM Research Report, RJ 1200, IBM Research Laboratory, San Jose, California.

9. BOSAK, R. (1962)
"An Information Algebra", CACM Vol. 5, No. 4, pp. 190-204.

10. BOYCE, R.F., et. al (1973)
"Specifying Queries as Relational Expressions: SQUARE", IBM Research Report, RJ 1291, IBM Research Laboratory, San Jose, California.

11. BOYCE, R.F., et. al (1973)
"Using a Structured English Query Language as a Data Definition Facility", IBM Research Report, RJ 1318, IBM Research Laboratory, San Jose, California.

12. BRACCHI, G., et. al (1972)
"A Language for a Relational Data Base", Sixth Annual Princeton Conference on Infromation Sciences and Systems, March 23-24, Princeton, New Jersey.

13. CANNING, R.G. (1974)
"Problem Areas in Data Management", EDP Analyzer, Vol. 12, No. 3.

14. CASEY, R.G., et. al (1974)
"Generalized Page Replacement Algorithms in a Relational Data Base", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, May 1-3, Ann Arbor, Michigan.

15. CHAMBERLIN, D.D., et. al (1974)
"SEQUEL: A Structured English Query Language", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, May 1-3, Ann Arbor, Michigan.

16. CHILDS, D.C. (1968)
"Feasibility of a Set Theoretic Data Structure", Proc. IFIP Congress, Vol. 1, North Holland Pub. Co., Amsterdam, pp. 420-430.

17. CODASYL (1969)
"Data Base Task Group Report", Available from ACM HQ., New York.

18. CODASYL (1971)
"Data Base Task Group Report", Available from ACM HQ., New York.

19. CODASYL (1971)
"Introduction to Feature Analysis of Generalized Data Base Management Systems", CACM, Vol. 14, No. 5, pp. 308-318.

20. CODASYL (1971)
"Feature Analysis of Generalized Data Base Management Systems", Available from ACM HQ., New York.

21. CODD, E.F. (1970)
"A Relational Model of Data for Large Shared Data Banks", CACM, Vol. 13, No. 6, pp. 377-387.

22. CODD, E.F. (1971)
"Further Normalization of the Data Base Relational Model", IBM Research Report, RJ 909, IBM Research Laboratory, San Jose, California.

23. CODD, E.F. (1971)
"Normalized Data Base Structure: A Brief Tutorial", IBM Research Report, RJ 935, IBM Research Laboratory, San Jose, California.

24. CODD, E.F. (1971)
"A Data Base Sublanguage Founded on the Relational Calculus", IBM Research Report, RJ 893, IBM Research Laboratory, San Jose, Calfironia.

25. CODD, E.F. (1971)
"Relational Completeness of Data Base Sublanguage", IBM Research Report, RJ 987, IBM Research Laboratory, San Jose, California.

26. CODD, E.F. (1974)
"Recent Investigations in Relational Data Base Systems", IBM Research Report, RJ 1385, IBM Research Laboratory, San Jose, California.

27. CODD, E.F. (1974)
"Seven Steps to Rendezvous with the Casual User", IBM Research Report, RJ 1333, IBM Research Laboratory, San Jose, California.

28. CODD, E.F., et. al (1974)
"Interactive Support for Non-programmers: The Relational and Network Approaches", IBM Research Report, RJ 1400, IBM Research Laboratory, San Jose, California.

29. DATE, C.J., et. al (1971)
"File Definition and Logical Data Independence", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, Nov. 11-12, San Diego, California, pp. 117-138.

30. DATE, C.J., et. al (1971)
"Storage Structure and Physical Data Independence", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, Nov. 11-12, San Diego, California, pp. 139-168.

31. DATE, C.J. (1972)
"Relational Data Base Systems: A Tutorial", Fourth International Symposium on Computer and Information Sciences, Miami Beach, Florida, Plenum Press.

32. DATE, C.J. (1972)
"An Introduction to the April 1971 Report of the CODASYL Data Base Task Group", IBM Technical Report, TR. 12.104, IBM United Kingdom Laboratories LTD, Hursley Park, Winchester Hampshire.

33. DATE, C.J., et. al (1974)
"The Relational and Network Approaches: Comparison of the Application Programming Interfaces", IBM Research Report, RJ 1401, IBM Research Laboratory, San Jose, California.

34. DATE, C.J. (1975)
"An Introduction to Data Base Systems", Addison Wesley, New York.

35. DAVIES, C.T. (1967)
"A Logical Concept for the Control and Management of Data", IBM Research Report, AR-0803-00, Poughkeepsie, New York.

36. DELOBEL, C., et. al (1973)
"Decomposition of a Data Base and the Theory of Boolean Switching Functions", IBM Journal of Research and Development, Vol. 17, No. 5, pp. 374-386.

37. D'IMPERIO, M.E. (1969)
"Data Structures and Their Representations in Storage", Annual Review in Automatic Programming, Vol. 5, Pergamon Press, New York, pp. 1-75.

38. DODD, G.G. (1969)
"Elements of Data Management Systems", Computing Survey, Vol. 1, No. 2, pp. 117-133.

39. EARNEST, C.P. (1974)
"A Comparison of the Network and Relational Data Structure Models", Computer Sciences Corporation Report, El Segundo, California.

40. ENGLES, R.W. (1970)
"A Tutorial on Data Base Organization", IBM Technical Report, TR 00.2004, Poughkeepsie, New York.

41. ENGLES, R.W. (1971)
"An Analysis of the April 1971 Data Base Task Group Report", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, Nov. 11-12, San Diego, California, pp. 69-91.

42. FEHDER, P.L. (1972)
"The Representation Independent Language Part 1: Introduction and the Subsetting Operation", IBM Research Report, RJ 1121, IBM Research Laboratory, San Jose, California.

43. FEHDER, P.L. (1973)
"The Representation Independent Language Part 2: Derivation and Insert, Update, and Delete Operations", IBM Research Report, RJ 1251, IBM Research Laboratory, San Jose, California.

44. FLORENTIN, J.J. (1974)
"Consistency Auditing of Data Bases", The Computer Journal, Vol. 17, No. 1, pp. 52-58.

45. HEATH, I.J. (1971)
"Unacceptable File Operations in a Relational Data Base", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, Nov. 11-12, San Diego, California, pp. 19-33.

46. HENRY, W.R. (1969)
"Hierarchical Structure of Data Management", IBM Systems Journal, Vol. 8, No. 1, pp. 2-16.

47. HSIAO, D., et. al (1970)
"A Formal System for Information Retrieval from Files", CACM, Vol. 13, No. 2, pp. 67-73.

48. KENT, W. (1973)
"A Primer of Normal Forms", IBM Technical Report, TR 02.600, IBM System Development Division, San Jose, California.

49. LEVIEN, R.E., et. al (1967)
"A Computer System for Inference Execution and Data Retrieval", CACM, Vol. 10, No. 11, pp. 715-721.

50. LORIE, R.A. (1974)
"XRM- An Extended (n-ary) Relational Memory", IBM Scientific Center Report, G320-2096, Cambridge, Mass.

51. MCGEE, W.C. (1969)
"Generalized File Processing", Annual Review in Automatic Programming, Vol. 5, Pergamon Press, New York, pp. 77-149.

52. MEALY, G.H. (1967)
"Another Look at Data", Proc. AFIPS, FJCC, Vol. 31, AFIPS Press, Montvale, New Jersey, pp. 525-534.

53. MELTZER, H.S. (1969)
"System Design Considerations for Data Base Support", IBM Technical Report, TR 02.468, IBM System Development Division, San Jose, California.

54. NAVATHE, S. (1974)
"Logical Normal Forms for Data Translation", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, May 1-3, Ann Arbor, Michigan.

55. PALERMO, F.P. (1972)
"A Data Base Search Problem", IBM Research Report, RJ 1072, IBM Research Laboratory, San Jose, California.

56. PALERMO, F.P. (1973)
"An APL Implementation of Relational Operators and a Search Algorithm", IBM Research Report, RJ 1273, IBM Research Laboratory, San Jose, California.

57. RISSANEN, J., et. al (1973)
"Decomposition of Files, a Basis for Data Storage and Retrieval", IBM Research Report, RJ 1220, IBM Research Laboratory, San Jose, California.

58. ROBERTS, D.C. (1972)
"File Organization Techniques", Advances in Computers, Vol. 12, pp. 115-174.

59. ROSENBERG, A.L. (1971)
"Data Graphs and Addressing Schemes", Journal of Computer and System Sciences, Vol. 5, No. 3, pp. 193-238.

60. ROSENBERG, A.L. (1973)
"Suffixes of Addressable Data Graphs", Information and Control, Vol. 23, pp. 107-127.

61. ROTHNIE, J.B. (1972)
"The Design of Generalized Data Management Systems", PhD Dissertation, Department of Civil Engineering, Massachusetts Institute of Technology.

62. ROTHNIE, J.B. (1974)
"An Approach to Implementing a Relational Data Management System", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, May 1-3, Ann Arbor, Michigan.

63. SENKO, M.E., et. al (1972)
"A Data Independent Architecture Model I: Four Levels of Description from Logical Structures to Physical Search Structures", IBM Research Report, RJ 982, IBM Research Laboratory, San Jose, California.

64. SEVERANCE, D.G. (1972)
"Some Generalized Modeling Structures for Use in the Design of File Organizations", PhD Dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, Michigan.

65. SIBLEY, E.H., et. al (1973)
"A Data Definition and Mapping Language", CACM, Vol. 16, No. 12, pp. 750-759.

66. SIBLEY, E.H. (1974)
"On the Equivalence of Data Based Systems", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, May 1-3, Ann Arbor, Michigan.

67. SMITH, D.P. (1971)
"An Approach to Data Description and Conversion", PhD Dissertation, The Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania.

68. STRNAD, A.L. (1971)
"The Relational Appraoch to the Management of Data Bases", Proc. IFIP Congress, Ljubljana, Yougoslavia.

69. TAYLOR, R.W. (1971)
"Generalized Data Base Management System Data Structures and Their Mapping to Physical Storage", PhD Dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, Michigan.

70. WANG, C.P., et. al (1974)
"An Approach for Segment Synthesis in Logical Data Base Design", IBM Research Report, RJ 1397, IBM Research Laboratory, San Jose, California.

71. WHITNEY, V.K.M. (1972)
"RDMS: A Relational Data Management System", Proc. Fourth International Symposium on Computer and Information Sciences, Miami Beach, Florida, Dec. 14-16, Plenum Press.

72. WHITNEY, V.K.M. (1974)
"Relational Data Management Implementation Techniques", Proc. ACM-SIGFIDET Workshop on Data Description, Access and Control, May 1-3, Ann Arbor, Michigan.

73. WILLIAMS, R. (1971)
"A Survey of Data Structures for Computer Graphics Systems", Computing Survey, Vol. 3, No. 1, pp. 1-22.

74. WONG, E., et. al (1971)
"Canonical Structure in Attribute Based File Organization", CACM, Vol. 14, No. 9, pp. 593-597.