Efficiency measurements between the third and fourth normal forms of database schemes
by Hirohisa Shimura

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
The purpose of this thesis is to explore the relationship between the efficiency of the third and the fourth normal forms in a database scheme and to prove the fourth normal form is more efficient and easier to use than the third normal form if the unnormalized form scheme contains a multivalued dependency.

Three methodologies are used to measure the efficiency of the third and the fourth normal form schemes. The first methodology measures the efficiency from the mathematical standpoint. The second methodology measures the efficiency from the machine standpoint, which is the machine time and the memory space required to execute a query. The third methodology measures the efficiency from the end user standpoint which is measured by counting the number of relational operations and execution time needed to execute a query.

First, if multivalued dependencies are present but not detected within the data model, a substantial amount of data redundancy will be expected to occur. If the third normal form scheme is utilized, the mathematical proof has shown that the memory space increases in size in a multiplicative fashion as the complexity of the data relationship increases. If the fourth normal form scheme is utilized, the memory space will increase in an additive fashion as the complexity of the data entities increases.

Second, during the machine efficiency measurements, the fourth normal form scheme performed more efficiently than the third normal form scheme in terms of the memory space utilization and the execution time. These observations were made during the machine simulation that encompassed four query exercises.

Third, observations regarding the relative ease of use associated with a series of end user queries has shown that the fourth normal form data scheme tended to be an easier database scheme to use. Ease of use was measured by the number of relational operations employed and execution time associated with a completed query routine.

Furthermore, the user's cognitive style has been shown to be an important factor in measuring the relative ease of use of a query.

The user's cognitive style may become a variable of interest during the database design phase. This should allow the database system to perform more efficiently when designed to incorporate the user's cognitive traits.

The thesis proves that the fourth normal form scheme is relatively more efficient and easier to use than the third normal form in terms of 1) the memory space allocation and utilization of both the theoretical and the machine stand points, 2) the machine execution time, and 3) the end user efficiency measurement measured by the number of relational operations employed and execution time associated with a completed query routine.

EFFICIENCY MEASUREMENTS BETWEEN THE THIRD AND FOURTH NORMAL FORMS OF DATABASE SCHEMES

by

Hirohisa Shimura

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 1987

APPROVAL

of a thesis submitted by

Hirohisa Shimura

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

November 13th 1987
Date

J Dunbigh Stanley
Chairperson, Graduate Committee

Approved for the Major Department

November 13th 1987
Date

J Dunbigh Stanley
Head, Major Department

Approved for the College of Graduate Studies

December 18, 1987
Date

Harry L Parsons
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Dean of Libraries when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Heather Stimson

Date 11-13-87

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The purpose of this thesis is to explore the relationship between the efficiency of the third and the fourth normal forms in a database scheme and to prove the fourth normal form is more efficient and easier to use than the third normal form if the unnormalized form scheme contains a multivalued dependency.

Three methodologies are used to measure the efficiency of the third and the fourth normal form schemes. The first methodology measures the efficiency from the mathematical standpoint. The second methodology measures the efficiency from the machine standpoint, which is the machine time and the memory space required to execute a query. The third methodology measures the efficiency from the end user standpoint which is measured by counting the number of relational operations and execution time needed to execute a query.

First, if multivalued dependencies are present but not detected within the data model, a substantial amount of data redundancy will be expected to occur. If the third normal form scheme is utilized, the mathematical proof has shown that the memory space increases in size in a multiplicative fashion as the complexity of the data relationship increases. If the fourth normal form scheme is utilized, the memory space will increase in an additive fashion as the complexity of the data entities increases.

Second, during the machine efficiency measurements, the fourth normal form scheme performed more efficiently than the third normal form scheme in terms of the memory space utilization and the execution time. These observations were made during the machine simulation that encompassed four query exercises.

Third, observations regarding the relative ease of use associated with a series of end user queries has shown that the fourth normal form data scheme tended to be an easier database scheme to use. Ease of use was measured by the number of relational operations employed and execution time associated with a completed query routine.

Furthermore, the user's cognitive style has been shown to be an important factor in measuring the relative ease of use of a query. The user's cognitive style may become a variable of interest during the database design phase. This should allow the database system to perform more efficiently when designed to incorporate the user's cognitive traits.

The thesis proves that the fourth normal form scheme is relatively more efficient and easier to use than the third normal form in terms of 1) the memory space allocation and utilization of both the theoretical and the machine stand points, 2) the machine execution time, and 3) the end user efficiency measurement measured by the number of relational operations employed and execution time associated with a completed query routine.

CHAPTER ONE

INTRODUCTION

The purpose of this research was to explore the relationship between the efficiency of the third and the fourth normal forms in a database scheme. The major hypothesis was that the fourth normal form is more efficient and easier to use than the third normal form if the unnormalized form scheme contains a multivalued dependency.

The third normal form is concerned with single-valued facts. A single-valued fact can be a one-to-one relationship or a one-to-many relationship, whereas the fourth normal form is associated with multivalued facts. A multivalued fact may correspond to a many-to-many relationship among data elements (Kent 1983, 121-123).

A database based on the third normal form is free from the four anomalies derived from functional dependency, where the four anomalies refer to maintenance problems within a relational database scheme. Some data redundancy is observed at this lower level of normalization. A fourth normal form database can sustain a minimal data redundancy by utilizing multivalued dependencies. The main difficulty corresponding to the fourth normal form is to capture an accurate set of multivalued dependencies.

James Martin (1983) stated that using the concept of the fourth normal form has caused confusion in database practice due to the confusion over defining the fourth normal form and the abstruse style of papers on the subject. Furthermore, Martin stated that "In common practice examples of non-fourth-normal-form data are rarely encountered, and easy to avoid" (1983, 233). While James Martin is opposed to idea of the fourth normal form, Daniel Martin reflects his opinion on this subject (1986,41):

> Historically it took many years to study the implications of relations of functional dependence between attributes, which is the opposite of independence. Both within a given relation and between different relations, the semantics of functional dependence interfered with the representation of the relation's data. We shall see that semantic problems occur when manipulating relations with relational algebra and show an example: the connection trap. Today a consensus has been reached: the use of the Fourth Normal Form.

The connection trap refers to a general problem of violation of data integrity and causes the loss of information or creates false information (Daniel Martin 1986, 41). The issue of data independence and integrity will be discussed in Chapter 4.

The motivation of this research is to support the argument for the usage of the fourth normal form with respect to the efficient use of the database. The research verified the hypothesis. The methods used to verify the hypothesis were: (1) theoretical proofs that the fourth normal form allocates less space than the third normal form in most situations; (2) time and space measurements of the third and the fourth normal form schemes done by a computer simulation; and (3) measurements of the number of operations and query formulation time required by users.

The conclusion based on the above methods is as follows: the third normal form with a multivalued dependency will create more data redundancies within a database as the complexity of the database increases. Hereafter, the third normal form refers to the third normal form with a multivalued dependency. The conclusion is restated as follows: The storage taken by a third normal form based database increases multiplicatively along with the complexity of a database. If the third normal form is losslessly decomposed into a fourth normal form based database, then the storage used by the fourth normal form increases additively. Therefore, the fourth normal form data scheme improves data redundancy control, and the efficiency of using a database is also increased.

Chapter 2 deals with the definition of a database system and the relative advantage of using a relational database model. Chapter 3 discusses normalization theory and normalized forms. Chapter 4 discusses the issue of data integrity from the view point of the third and fourth normal data schemes and their difference. In Chapter 4, the difference between the third and fourth normal form is also illustrated through the decomposition scheme with an emphasis on a multivalued dependency. If the third normal form contains a set of multivalued dependencies, the difference is manifested during the normalization.

Chapters 5 and 6 explore the hypothesis from the perspective of computer space utilized by each database scheme. Chapter 5 analyzes the mathematical difference between the two normalized schemes with respect to their storage requirements. Two mathematical proofs are

given. One proof illustrates a multiplicative increase of secondary storage space of the third normal form scheme. The other proof illustrates an additive increase of secondary storage space of the fourth normal form scheme. Chapter 6 discusses the methodology of the machine efficiency measurements and the results. It concludes with the fact that the fourth normal scheme is more efficient than the third normal form scheme with respect to the storage requirement and the execution time.

Chapter 7 analyzes the efficiency measurements between the third and the fourth normal data schemes with respect to the user's perspectives. Based on analyses presented in Chapter 7, three conclusions are reached. One conclusion is that the fourth normal form scheme is found to be more efficient in terms of machine efficiency measurements and the ease of use when compared with the third normal form database scheme. The second conclusion is that computer professionals require less time and a smaller number of commands to complete a query than business users. The third conclusion is that the user's cognitive style must be considered during the database design phase since the user's cognitive style can be a very important factor in determining ease of use. Chapter 8 summarizes and concludes the thesis.

CHAPTER TWO

DATABASE MANAGEMENT SYSTEMS

Introduction

In this chapter, the definition of a database management system (DBMS) and three data models related to database management systems are presented. The advantages of a relational data model are also discussed. There are various approaches to defining a database management system. Definitions of a database seem to differ from author to author, but one objective is consistent: to help a user retrieve data effectively and efficiently. Sample definitions from different authors illustrate the alternative views of a database management system.

Daniel Martin defines a database as "a collection of information on a well-defined subject that is exhaustive, nonredundant, and structured (1986, 5).

The concept of exhaustivity addresses the issue of the completeness of a database. A database must contain all reasonable information about the data in a database. Nonredundancy makes the minimal storage requirements of a database and the enforcement of consistency possible. If a database stores the same information more than once, an inconsistent situation occurs which leads to the poor utilization of memory capacity. With redundant data elements there is

no assurance that updates will correspond to all data entries. Consequently, a query may yield a contradicting result due to the lack of integrity generated via incorrect data updating. Martin, however, noted that "nonredundancy is not always easy to achieve and must sometimes be compromised" (1986, 5-6).

McFadden and Hoffer define a database as follows (1985, 3):

> ... a shared collection of interrelated data designed to meet the varied information needs of an organization. A data base has two important properties: it is integrated and it is shared. By integrated we mean that previously distinct data files have been logically organized to eliminate (or reduce) redundancy and to facilitate data access. By shared we mean that all qualified users in the organization have access to the same data, for use in a variety of activities.

These authors discuss the advantages of a database as compared to traditional file approaches. McFadden and Hoffer identify 10 benefits of the database approach and there are 1) minimal data redundancy, 2)consistency of data, 3) integration of data, 4)sharing of data, 5) enforcement of standards, 6) ease of application development, 7) uniform security, privacy, and integrity controls, 8) data accessibility and responsiveness, 9) data independence, and 10) reduced program maintenance (1985, 15-19).

Martin's database definition is focused on an idea of how information should be stored and structured within a database. McFadden and Hoffer's definition is predicated on how a database should be organized in terms of an organization's and users' views. Both definitions emphasize the requirement that a database management

system (DBMS) should allow the use of data to cross operational, functional, or organizational boundaries. A DBMS, therefore, enables the database users to satisfy multiple "views" of the data stored.

Fundamental Database Models

When a database is implemented, the database system usually is structured for using one of three models which are hierarchical, network, or relational in nature. Each model has its own definition and characteristics. A brief presentation of each model follows:

Hierarchical Database Model

The hierarchical data model has the following characteristics:

A hierarchical schema consists of two sets:

(i) A set of entity types E. Particular entity types will be denoted as E1, E2, . . . , En.

(ii) A set of logical relationships L among entity types. The relationship between entity types Ei and Ej will be denoted as Lij.

The logical relationships among entity types satisfy the following requirements:

(iii) There exists at most one relationship Lij between different entity types Ei and Ej. As a consequence, there is no need to name the relationships.

(iv) There exists no relationship between an entity Ei and itself; that is, Lii is not a member of the set L (Alagic 1986, 49).

Furthermore Alagic (1986, 53) states that "An entity type may be the owner in one or more sets, but as a member it can participate in only one set." McFadden and Hoffer state that

> The hierarchical data model represents data as a set of nested one-to-many (1:M) and one-to-one (1:1) relationships. ... Because of the similarity of hierarchies and tree data structures, terminology like root, level, and leaf are also used when discussing hierarchies (McFadden and Hoffer 1985, 169).

Atre (1980) discusses advantages and disadvantages of a hierarchical data model. The advantages of a hierarchical scheme are as follows:

> The major advantage of the hierarchical data model is the existence of proven data base management systems that use the hierarchical data model as the basic structure.
> The relative simplicity and ease of use of the hierarchical data model and the familiarity of data processing users with a hierarchy are major advantages.
> There is a reduction of data dependency. ...
> Performance prediction is simplified through predefined relationships (Atre 1980, 106).

Note that the hierarchical model has the following disadvantages:

> The many-to-many relationship can be implemented only in a clumsy way. This may result in redundancy in stored data. ...
> As a result of the strict hierarchical ordering, the operations of insertion and deletion become unduly complex.
> Deletion of parent results in the deletion of children. ...
> Hierarchical commands tend to be procedural because of the strictness of the structure.
> "Root" is the dominant node type. Any child node is accessible only through its parent node (Atre 1980, 106-109).

Network Database Model

The network model was developed to compensate for the hierarchical data model's limitations associated with data access and retrieval. The network model has the following characteristics:

> (1) Sets
> If E1 and E2 are entity types such that there exists a functional relationship E2 -> E1, then the ordered named pair E(E1, E2) is called a set. E1 is called the owner of the set E and E2 the member of the set E. ...
> (3) Set Occurrences
> An occurrence of a set consists of an occurrence of the owner entity type E and zero or more occurrences of the member entity type E. In terms of functional relationships this means that a set occurrence consists of a set of member entities and the owner entity to which they are mapped according to the functional relationships. ...

(4) Uniqueness of Owner
The fact that a set is in fact a representation of a functional relationship may be expressed by the condition that an entity may participate as a member in only one occurrence of a given set. Because of that, a member entity has a unique owner entity in every set in which it participates as a member (Alagic 1986, 51-55).

Daniel Martin (1986, 58) states that

> The network model features N-to-P relationships. Each N-to-P relationship usually consists of two 1-to-N relationships in opposite directions linking the same pair of records.

The advantages of the network model relative to the hierarchical form are as follows:

> The major advantage of the network data model is that, as for the hierarchical data model, there are successful data base management systems that use the network data model as the basic structure.
>
> The many-to-many relationship, which occurs quite frequently in real life, can be implemented easily.
>
> The network data model is backed by the CODASYL (Conference On Data Systems Languages) Data Base Task Group (DBTG) (Atre 1980, 121).

Atre states the disadvantages of using the network model (1980, 121):

> The main disadvantage of the network model is its complexity. The application programmer must be familiar with the logical structure of the data base, because she/he has to "navigate" through different set occurrences with the help of connector record type occurrences. ...
> ... it is possible, unless great care is taken, to lose data independence.

Relational Database Model

The third database model is the relational model, which has the following characteristics:

> A relation scheme R is a finite set of attribute names {A1,A2,...,An}. Corresponding to each attribute name Ai is a set of Di, 1 ≤ i ≤ n, called the domain of Ai. We also

> denote the domain of Ai by dom(Ai). Attribute names are sometimes called attribute symbols or simply attributes, particularly in the abstract. The domains are arbitrary, non-empty sets, finite or countably infinite. Let D = D1 U D2 U ... U Dn. A relation r on relation scheme R is a finite set of mappings {t1,t2,...,tp} from R to D with the restriction that for each mapping t ∈ r, t(Ai) must be in Di, 1 ≤ i ≤ n. The mappings are called tuples. ...
> ... a tuple is a set of values, one for each attribute name in the relational scheme. ...
> A key of a relation r on relation scheme R is a subset K = {B1,B2,...,Bn} of R with the following property. For any two distinct tuples t1 and t2 in r, there is a B K such that t1(B) = t2(B) (Maier 1983, 2-4).

There are two formal languages for the relational database in order to represent queries about properties of entities represented by the relational database: relational algebra and relational calculus.

> Relational algebra manipulates one or two relations as operands and produces a new relation as the result. ... Relational calculus manipulates relations implicitly by specifying conditions that can involve attributes from several relations (McFadden and Hoffer 1985, 191-193).

The four criteria of the normalization process, mentioned above, are discussed in detail in Chapter 3.

The relational data model offers the following advantages over nonrelational data forms: (1) simplicity, (2) nonprocedural requests, (3) data independence, and (4) theoretical foundation (Atre 1980, 94-95). Each advantage will be discussed in the context of a comparison between a relational model and a nonrelational model. According to Daniel Martin (1986, 54) the primary difference between a relational database and nonrelational (navigational) database is that a relational database manipulates one relation at a time, whereas a

nonrelational database manipulates one record at a time. A nonrelational database may require a program to achieve the same result which a relational database can do with one operation.

A nonrelational database is also called a navigational database since "... navigational means 'what describes which path to use to find the data in the database' " (Martin 1986, 54). A nonrelational, navigational, database must follow a path navigated by a program, whereas a relational database needs to know what to obtain from the data entity. The database management system requires

> ... the addition of logic to navigate the DBMS structures to derive the desired application structures.
>
> Such applications suffer from significant problems:
> - They tend to be larger and more difficult to write
> - The potential for multiple errors is great
> - Development time tends to be expanded
> - Applications therefore require larger resource investments
> - Changes in the database structure may require reinvestment in applications to enable them to derive the same desired data structures
> - Applications can only depend on the DBMS for management of the files over which it has control (Wood 1985, 15).

The final difference between a relational and nonrelational database is the type of tools available for each database in order to formulate complex queries. A relational database utilizes relational algebra and relational calculus, which are derived from set theory; however, a nonrelational database utilizes a data manipulation language. A typical data manipulation language has more than 200 reserved words; however, commonly used relational databases use only approximately 24 reserved words (Martin 1986, 55).

McFadden and Hoffer discuss five fundamental differences between relational and nonrelational database in addition to Martin's observation. McFadden and Hoffer listed these differences to illustrate the advantages of using a relational database (1985, 183-184):

1. Implementation independence: the relational model logically represents all relationships implicitly, hence, one does not need to know what association are or are not physically represented by an efficient access path (without looking at the internal data model).
2. Terminology: the relational model has been developed with its own set of terminology, most of which has equivalent terms in other data models.
3. Logical key pointers: the relational data model uses primary (and secondary) keys in records to represent the association between two records; ...
4. Normalization theory: properties of a data base that make it free of certain maintenance problems have been developed within the context of the relational model....
5. High level programming languages: programming languages have been developed specifically to access data bases defined via the relational data model; these languages permit data to be manipulated as groups or files and not procedurally one record at a time.

All three authors believe that a relational model is more powerful than the other database models because of the above observations.

The advantages of using a relational model are summarized, based on all three authors, as follows:

1) how the database executes a query;
2) tools available to manipulate data items; and
3) a well defined set of mathematical terminology (Martin 1986; McFadden and Hoffer 1985).

Farin and Nazario (1986, 46) state the disadvantages of using a relational model.

> The main weakness of a relational DBMS stems from its strengths. There is a potential risk of costly and inefficient access. Users may not select the optimal access path, and may not state their data requests in the manner most efficient for the DBMS. ...
> It is also difficult for a data base designer to optimize the physical representation of data prior to its use. These problems become more readily apparent when large databases are being maintained.

The issue of user efficiency with regard to relational data base design is one major focus of this thesis and will be discussed in Chapter 7.

CHAPTER THREE

NORMALIZATION THEORY AND NORMAL FORMS

Introduction

In this chapter, three concepts related to the relational database model are discussed. The first concept is the normalization theory, which guides the database designer in the construction of a database. The second concept is the dependency theory, which enables the database designer to justify the degree of normalization. The third concept is that of a normal form, which is the end product of the normalization process.

Normalization Theory

McFadden and Hoffer define normalization and the purpose of normalization as follows (1985, 221):

> Normalization is the analysis of functional dependencies between attributes (or data items). The purpose of normalization is to reduce complex user views to a set of small, stable data structures. Experience clearly shows that normalized data structures are more flexible, stable, and easier to maintain than unnormalized structures.

A possible disadvantage to the normalization process seems to exist. For instance, assume finding a certain piece of information within a database is necessary. When a normalized form is used, often a costly join operation must be utilized. As a result, a new relational table is created. Then the selection of a data field from the newly created

relation must be made. However, in order to extract the same information from an unnormalized form, a simple selection and projection operation is required. One must carefully define the benefits of data normalization in order to fully appreciate the power of a normalized relational database model.

Perkinson (1984, 42) discusses two benefits from database normalization: the minimization of "data maintenance" and minimization of "program maintenance." "Maintenance involves both a minor task such as expanding the length of a field and complex tasks such as adding new fields due to increased functional complexity" (Perkinson 1984, 41). Minimization of data maintenance is concerned with the problem of maintaining integrity during data update. Whenever a data item is modified, that item must be changed every time it appears in the database, either directly by the user, or indirectly by the system.

Minimization of program maintenance deals with a problem of internal change inside of the database. Perkinson (1984) gave an example of expanding the length of a certain data field. This internal change requires the database to locate and update every program that is used in the database, even for programs that do not include a specified data field. Only programs which "contain the data in question should be updated" (Perkinson 1984, 41).

Without the concept of normalization, a possible approach to avoiding the problems of maintenance is to break down the original large data file into a set of smaller data files. Each data item is related to every other item within a data file. However, this is an

intuitive notion of the normalization process. Decomposition of a large file into a set of small manageable files is the fundamental idea of normalization.

Four Criteria of Normalization

The four criteria of normalization as discussed by Ullman are summarized into four components (1982, 212):

1) Redundancy
2) Potential inconsistency (update anomalies)
3) Insertion anomalies
4) Deletion anomalies

These four criteria are also known as maintenance problems within the relational database scheme. These four anomalies and their elimination are used to introduce the existence of different levels of normal forms. A lower normal form suffers from these four anomaly situations, whereas a higher normal form solves them. These four anomalies are called the criteria of normalization since they are a means of progressing from a lower normal form to a higher normal form.

Data redundancy is a situation where the logical database scheme allows a particular data element to be stored in more than one location within the database. The logical database scheme may suffer from the following problems if data redundancy is allowed to occur:

1) Storage space in the computer may be wasted due to data redundancy;
2) In order to update all occurrences of a particular data item, several input prcedures must be made directly or indirectly by a user; and
3) Inconsistencies within the database may be exhibited if all occurrences of the same data items are not inserted (deleted, or updated) (McFadden and Hoffer 1985, 11).

An insertion anomaly is observed when a user inserts a new data item without inserting a value for every primary key. If an undefined value for a primary key is allowed within a database, it may lead to an update anomaly or the destruction of the functional dependencies among the data items. Thus, every primary key must have a defined value within the database. A primary key refers to "a key that is used to uniquely identify a record instance" (Martin 1983, 746).

A deletion anomaly occurs when a user fails to delete a data item from the database when a primary key has been deleted; conversely, when a user tries to delete a particular data item, some important but unrelated information may be inadvertently deleted.

An update anomaly is observed when a user fails to update the database properly. When multiple occurrences of data items in a data file are permitted, updating only a single occurrence of the data items may lead to an inconsistency in the database. Duplicate data items may not be updated simultaneously.

Dependencies and Redundancies

The relationship between dependencies and redundancies is the key to the normalization process. Avoidance of redundancies forces an original data file to be decomposed into a set of small data files. The rules utilized to decompose an original data file into a set of smaller data files are called dependencies.

> ... given the fact that a relation R is legal, i.e., satisfies certain dependencies, and given certain information about the value of R, we should be able to deduce other things about R (Ullman 1982, 213).

The dependency not only causes the redundancies, but also permits a lossless decomposition. An example can be used to illustrate this concept. Assume we have a relation SUPPLIER(SNAME, SADDRESS, ITEM, PRICE). The four maintenance problems mentioned earlier are observed in this relation. Since data redundancy illustrates the need for detection of a dependency, data redundancy is focused on. In this example, redundancy is easily detected since "the address of the supplier is repeated once for each item supplied" (Ullman 1982, 212). If the content of SNAME is known, it is easy to deduce the content of SADDRESS. (SADDRESS is functionally dependent on SNAME.) All but one attribute of SADDRESS in this relation is redundant. In order to avoid any redundancy, one can break down SUPPLIER into two distinct relations, namely SA(SNAME, SADDRESS) and SIP(SNAME, ITEM, PRICE) (Ullman 1982, 212). Furthermore, one can recover the original relation by joining SA and SIP by using a natural join. Therefore we have achieved the lossless decomposition (see Figure 1).

Several dependencies are central to the theory of normalization. Three dependencies are emphasized for normalization processes: (1) functional dependencies, (2) multivalued dependencies, and (3) join dependencies. These dependencies will be discussed in the sections which follow.

SUPPLIER(SNAME, SADDRESS, ITEM, PRICE)

|
|Normalization is undertaken.
V

SA(SNAME, SADDRESS)

and

SIP(SNAME, ITEM, PRICE).

SA(SNAME, SADDRESS) SIP(SNAME, ITEM, PRICE)

|
|Natural join is undertaken.
V

SUPPLIER(SNAME, SADDRESS, ITEM, PRICE)

Figure 1. Lossless Decomposition.

Functional Dependency. The definition of functional dependency (FD) is as follows:

> Let R(A1,A2,...,An) be a relation scheme, and let X and Y be subsets of{A1,A2,...,An}. We say X -> Y, read 'X functionally determines Y' or 'Y functionally depends on X' if whatever relation r is the current value for R, it is not possible that r has two tuples that agree in the components for all attributes in the set X yet disagree in one or more components for attributes in the set Y (Ullman 1982, 214).

Therefore, two tuples that agreed on the attributes of X would have to be the same entity. Ullman further states that:

> ... if R represents a many-one mapping from entity set E1 to entity set E2, and among the Ai's are attributes that form a key X for E1 and a key Y for E2, then X -> Y would hold, and in fact, X functionally determines any set of attributes of R. However, Y -> X would not hold unless the mapping were one-to-one (1982, 214).

Functional dependency is limited to representing one-to-one and one-to-many relationships. An illustration of functional dependencies may help to visualize this concept more clearly. A functional

dependency is found in the relation SA: SNAME -> SADDRESS as was discussed earlier. If one knows the name of the supplier, one can find the address since SNAME functionally determines SADDRESS. By asserting that a functional dependency exists, the storage of certain information can be limited. If a functional dependency: SNAME -> SADDRESS holds, no supplier can have two addresses in the database system (Ullman 1982, 215). If in actuality more than one address exists, the problem can be resolved by using multivalued dependencies.

Multivalued Dependency. The concept of multivalued dependency (MVD) has been introduced by Ronald Fagin (1977, 262-282). Fagin called a generalization of functional dependencies a multivalued dependency. Maier defines multivalued dependency as follows:

> Let R be a relation scheme, let X and Y be disjoint subsets of R, and let Z = R - (X Y). A relation r(R) satisfies the multivalued dependency (MVD) X ->-> Y if, for any two tuples t1 and t2 in r with t1(X) = t2(X), there exists a tuple t3 in r with t3(X) = t1(X), t3(Y) = t1(Y), and t3(Z) = t2(Z).
>
> The symmetry of t1 and t2 in this definition implies there is also a tuple t4 in r with t4(X) = t1(X), t4(Y) = t2(Y) and t4(Z) = t1(Z) (1983, 124).

Maier also states the relationship between multivalued dependencies and functional dependencies: "Let r be a relation on scheme R and let X and Y be subsets of R. If r satisfies the FD X -> Y, then r satisfies the MVD X ->-> Y" (1983, 127). Therefore, multivalued dependency is merely a generalization of functional dependency. To illustrate the need for multivalued dependencies, assume the following example: A relation R(CLASS, SECTION, STUDENT, EXAM, TEXT) (Fagin 1977, 268-270). There are no functional dependencies among this

relation; therefore, this relation is free from any of the four anomalies from the perspective of functional dependencies. However, this relation has the following property: if a particular class and section used more than one textbook and has more than one examination, an examination is dependent only on the class and section and not on the textbook used. In other words, for each class section, there is a well defined set of textbooks and a set of examinations. The set of textbooks is independent from the set of examinations but is dependent on class section alone.

Despite the fact that this relation does not have any functional dependencies, there exists a great deal of data redundancy. This data redundancy leads to the creation of the other three anomalies. For example, if a particular class section decides to drop a textbook, several deletions must be made (as many as the number of examinations in a particular class section). If a new textbook is added to a certain class section, several tuples must be created. Due to this multivalued dependency in the relation (CLASS SECTION ->-> TEXT, CLASS and SECTION STUDENT ->-> EXAM), one can further decompose the relation. Two projections R1 and R2 are the results of decomposition. R1 contains CLASS, SECTION, STUDENT, EXAM, and R2 contains CLASS, SECTION, TEXT. This projection now sets the database free from any of the four anomalies.

Join Dependency. The last dependency used in normalization theory is called join dependency. The concept of join dependency was first

introduced by Rissanen (1977, 317-325). Maier states the concept behind join dependencies (1983, 139):

> MVDs are an attempt to detect lossless decompositions that will work for all relations on a given relation scheme. However, MVDs are not completely adequate in this regard. A relation can have a nontrivial lossless decomposition onto three schemes, but have no such decomposition onto any pair of schemes....

Date (1986) clarified the join dependency theory by using the idea of n decomposability. As far as the fourth normal form is concerned, replacing a relation by two of its projections is the sole operation necessary or available in the nonlossless decomposition process (Date 1986, 385). A projection refers to the component of the database after the decomposition process is undertaken. There exist relations in which nonlossless decomposition can be achieved only by breaking down the relation into three or more projections, (i.e., two projections are not enough to achieve lossless decomposition.) A definition of join dependencies is as follows:

> Let R = {R1,R2,...,Rp} be a set of relation schemes over U. A relation r(U) satisfies the join dependency (JD) *[R1,R2,...,Rp] if r decomposes losslessly onto R1,R2,...,Rp. That is, r = πR1(r) $\bowtie$ πR2(r) $\bowtie$ ... $\bowtie$ πRp(r). We also write *[R1,R2,...,Rp] as *[R] (Maier 1983, 139-140).

Date (1986, 388) states that "JDs are a generalization of MVDs" and that they are "the most general form of dependency possible." To illustrate a join dependency, assume there exists a relation R(AGENT, COMPANY, PRODUCT). Furthermore there are no multivalued dependencies in R; however, a join dependency exists: *{(AGENT, COMPANY), (AGENT, PRODUCT), (COMPANY, PRODUCT)}. Due to the join dependency, one can decompose R into three projections: R1 = (AGENT, COMPANY), R2 =

(AGENT, PRODUCT), and R3 = (COMPANY, PRODUCT). If one arbitrarily chooses to join two projections from these three relations, a superfluous relation is created. In order to recover the original relation R, a natural join of all the three projections must be undertaken (Kent 1983, 123-125).

To summarize the three proceeding sections, normalization is the key process in the database design. In order to achieve lossless decomposition, a database designer must always confront the four criteria associated with normalization and dependencies. The focus of discussion will now shift to normal forms, an end product of normalization. Normal forms are used to represent a conceptual database model as the database design process continues to refine the conceptual view.

Normal Forms

Date (1986, 362) states that "Normalization theory is built around the concept of normal forms. A relation is said to be in a particular normal form if it satisfies a certain specified set of constraints." The concept of normal forms is essential in the relational database structure since a normal form is based upon the relationship existing within a database. There are six normal forms: the first normal form to the fourth normal form, the fifth normal form (also called Project-Join), and the Boyce-Codd normal form. Each normal form will be introduced and its role in a relational database will be discussed below.

The definition of a relational database scheme is stated mathematically according to Maier (1983, 94).

> Let U be a set of attributes, each with an associated domain. A relational database scheme R over U is a collection of relation schemes {R1,R2,...,Rp}, where Ri = (Si,Ki), $1 \leq i \leq p$,
>
> $$\bigcup_{i=1}^{p} Si = U,$$
>
> and $Si \neq Sj$ if $i \neq j$.
> A relational database d on database scheme R is a collection of relations {r1,r2,...,rp} such that for each relation scheme R = (S,K) in R there is a relation r in d such that r is a relation on S that satisfies every key in K. We abuse set notation and always assume that ri is the relation on Si.

S represents a set of attributes and K represents a set of designated keys.

An unnormalized data form is defined as the following:

> An unnormalized relation is a relation that contains one or more repeating groups. As a result, there are multiple values at the intersection of certain rows and columns (McFadden and Hoffer 1985, 222).

The disadvantage of an unnormalized form is that data redundancy is found. The goal of data normalization is to divide a database into a flexible and stable set of simple data schemes while free from the four anomalies previously discussed.

A definition of first normal form is the following:

> A relation scheme R is in first normal form (1NF) if the values in dom(A) are atomic for every attribute A in R. That is, the values in the domain are not lists or sets of values or composite values. A database scheme R is in first normal form if every relation scheme in R is in first normal form (Maier 1983, 96).

The advantage of 1NF is that "it may be not possible to express FDs" without 1NF (Maier 1986, 97). If a list or set of values or

composite values is allowed, there could exist some situation where semantically the same information is not allowed to represent the same functional dependencies. The example used by Maier is FD: "BIRTHDATE -> SIGN, which would allow two people born on the same day in different years to have different signs" (1983, 97).

An example by Date (1986, 367-374) illustrates second and third normal forms with respect to the four criteria of normalization; it will be presented at length below. This relation appears as follows: FIRST(S#, STATUS, CITY, P#, QTY), where S# represents the supplier number, STATUS represents the status of the city, CITY represents the name of the CITY, P# represents the parts number, and QTY represents quantity of the particular part number.

One must apply four criteria of normalization in order to improve the relation FIRST. Data redundancy is detected easily since every tuple with a particular S# must have the same STATUS, and CITY for every different part and the quantity a supplier supplies. An insertion anomaly is observed when a particular supplier in a particular city without a specific part is inserted. According to Date, no component of a primary key value can be unassigned (1986, 367). In relation FIRST, the combination of S# and P# is the primary key. A deletion anomaly is detected when a particular supplier is deleted from the relation FIRST. Not only may a supplier number be deleted but also city information may be deleted as well. An update anomaly is observed when a particular supplier moves to a new city. Two possibilities exist. First, the user searches every tuple with a particular supplier and the previous city name is changed to a new

city name. Second, if all tuples are not changed, an inconsistent result is produced.

In order to fulfill the four criteria of normalization the data structure must be decomposed into the two distinct relations, SECOND(S#, STATUS, CITY) and SP(S#, P#, QTY). Maier defined the second normal form as follows (1983, 99):

> Given a relation scheme R, an attribute A in R, and a set of FDs F over R, attribute A is prime in R with respect to F if A is contained in some key of R. Otherwise A is nonprime in R. ... A relation scheme R is in second normal form (2NF) with respect to a set of FDs F if it is in 1NF and every nonprime attribute is fully dependent on every key of R. A database scheme R is in second normal form with respect to F if every relation scheme R in R is in 2NF with respect to F.

Relation SECOND and SP are both in 2NF since nonkey attributes, STATUS and QTY are fully functionally dependent on the primary key S#, and S# and P# respectively. The reduction process from 1NF to 2NF is achieved by applying suitable projections. The original relation, FIRST, can always be recovered by utilizing the natural join of these projections, SECOND and SP; therefore, no information is lost in this reduction process. Date calls this process a "nonloss decomposition." A nonloss decomposition refers to a decomposition scheme which produces a set of projections and by joining a set of projections no information is lost (Date 1986, 371).

Relation SECOND still does not totally meet the four criteria of normalization. Data redundancy is observed since every city has only one status value. If more than one supplier resides in the same city, information about city status is redundant. An insertion anomaly is encountered when an insertion of a new city is made, particularly when

a supplier exists in a new city and no appropriate primary key value exists for that supplier. A deletion anomaly is observed when a deletion of a particular city is made: not only is the supplier's information deleted but also the status of the city is deleted if there is only one supplier in the city. Observation of an update anomaly is made when changing the status for a particular city. There are two possible solutions available in regard to update anomaly. One possibility is simply to search every occurrence of a tuple with a particular city and change the status. The other possibility is to let an inconsistent result propagate within a database because only one data record is updated (Date 1986, 372).

The solution to these problems is to break down the relation SECOND into two projections, SC(S#,CITY) and CS(CITY,STATUS). These two projections are said to be in the third normal form. Maier defines third normal form as follows (1983, 100):

> Given a relation scheme R, a subset X of R, an attribute A in R, and a set of FDs F, A is transitively dependent upon X in R if there is a subset Y of R with X -> Y, Y $\not\rightarrow$ X, and Y -> A under F and A $\notin$ XY. ... A relation scheme R is in third normal form (3NF) with respect to a set of FDs F if it is in 1NF and no nonprime attribute in R is transitively dependent upon a key of R. A database scheme R is in third normal form with respect to F if every relation scheme R in R is in third normal form with respect to F.

Relations CS, SC, and SP are third normal form since no transitive dependencies exist on nonprime attributes on keys. Furthermore, these three relations are free from the four anomalies.

3NF has certain inadequancies in

the case of a relation that
1. had multiple candidate keys, where
2. those candidate keys were composite, and
3. the candidate keys overlapped (i.e., had at least one attribute in common) (Date 1986, 374).

A candidate key refers to "a key that uniquely identifies normalized record instances of a given type" (Martin 1983, 738).

In order to prevent these design inadequancies, a different normal form is needed to address the four criteria of normalization. To illustrate the situation, a new relation SSP(S#, SNAME, P#, QTY) is introduced and "the candidate keys are (S#, P#) and (SNAME, P#)" (Date 1986, 376). One must note that SSP is in 3NF since the definition of 3NF does "not require an attribute to be fully dependent on the primary key if it was itself a component of some attribute key of the relation" (Date 1986, 376). Therefore, SNAME is not fully dependent on (S#, P#) and was ignored. However, SSP has failed to meet the four criteria of normalization. Data redundancy is easily observed in that every tuple has the S# and SNAME, whereas "S# and SNAME are determinants because each of them determines the other" (Date 1983, 376). An insertion anomaly is detected when a particular supplier is inserted without inserting a corresponding part number into the database. The same logic used to illustrate the second normal form is applied here since no primary key can contain a null value. A deletion anomaly is detected when a particular supplier number is deleted, as well as the other information concerning the supplier. An update anomaly is found when changing the name of a supplier. There are two possibilities: one is to search and change every occurrence

of a particular supplier, and the other is to produce a possibly inconsistent result.

The solution is to decompose the relation SSP into two projections, SS(S#, SNAME) and SP(S#, P#, QTY). Maier defined the new relation as follows (1983, 118):

> A relation scheme R is in Boyce-Codd normal form (BCNF) with respect to a set of FDs F if it is in 1NF and no attribute in R is transitively dependent upon any key of R. A database scheme R is in Boyce-Codd normal form with respect to a set of FDs F if every relation scheme R in R is in Boyce-Codd normal form with respect to F.

The theoretical difference between 3NF and BCNF is that in 3NF only nonprime attributes in R cannot be transitively dependent upon a key of R, whereas in BCNF, any attribute in R cannot be transitively dependent upon a key of R. BCNF mainly calls attention to functional dependencies, which are useful with respect to 1-to-1, many-to-1, and 1-to-many relationships. Functional dependencies alone cannot provide a complete model. There are some situations where no further decomposition can be made with respect to functional dependencies. Thus, a relation fails to meet the four criteria of normalization. If this situation occurs, introduction of a new normal form cannot be avoided. This normal form must satisfy the four criteria of normalization. The new normal form was called a fourth normal form and was first introduced by Fagin (1977, 262, 278).

To depict the difference between the third normal form and the fourth normal form, a relation R(CLASS, SECTION, STUDENT, EXAM, TEXT) will be utilized from Fagin (1977, 267). It is readily shown that relation R fails to meet the four criteria of normalization. Data

redundancy exists within relation R since every occurrence of a data element of EXAM, must accompany all occurrences of data elements of TEXT. It is also the case that every occurrence of a data element of TEXT must accompany all occurrences of data elements of EXAM. There will be M entries of the same information regarding textbooks (TEXT) for every occurrence of examination (EXAM) where M represents the number of distinct textbooks (TEXT). Data redundancy exists since multivalued dependencies exist in this relation: {CLASS, SECTION} ->-> TEXT and {CLASS, SECTION, STUDENT} ->-> EXAM. However, the relation R does not have any determinants. Therefore, the relation R is in the third normal form.

The relation R fails to satisfy the four criteria of normalization. An insertion anomaly is detected when an insertion of a new textbook is made without an entry of examination type. This insertion is not allowed since CLASS, SECTION, STUDENT, EXAM, and TEXT is the only key (composite key) in relation R, and no key attribute can have a null value associated with it. A deletion anomaly is observed when the only exam score in the relation is deleted. At the same time the other information about CLASS, SECTION, STUDENT, and TEXT is deleted. An update anomaly is detected when a certain updating mechanism is performed. Every tuple must be searched and updated accordingly. Otherwise, an inconsistent result may be produced.

To resolve these four anomalies, the lossless decomposition of relation R into two its projections, R1={CLASS, SECTION, TEXT} and R2={CLASS, SECTION, STUDENT, EXAM} must be done. These two

projections are called fourth normal forms. Maier defined the fourth normal form as follows (1983, 136):

> Let F be a set of FDs and MVDs over U. A relation scheme R $\subseteq$ U is in fourth normal form (4NF) with respect to F if for every MVD X->-> Y implied by F that applies to R either the MVD is trivial or X is a superkey for R. A database scheme R is in fourth normal form with respect to F if every relation scheme R in R is in fourth normal form with respect to F.

A superkey is a broad definition of a key, similar to the definition provided in Chapter 2 (Maier 1983, 4-5).

Finally, the fifth normal form is introduced. Date states that "the 5NF is the ultimate form" but that fifth normal forms "are pathological cases and likely to be rare in practice" (1986, 390). Maier defined the fifth (project join) normal form as follows (1983, 141):

> Let R be a relation scheme and let F be a set of FDs and JDs. R is in projection-join normal form (PJNF) with respect to F if for every JD *[R1,R2,...,Rp] implied by F that applies to R, *[R1,R2,...,Rp] is implied by the key FD of R.

To summarize, there is a well defined theory of normalization to support relational databases. The three most important forms are as follows: the third, the Boyce-Codd, and the fourth normal forms, where the third and the Boyce-Codd normal forms deal with single These normal forms are practiced widely depending upon the application. The following chapter deals with third and fourth normal forms, since the efficiency measurement of both normalized forms is the topic of this research.

CHAPTER FOUR

DATA INTEGRITY OF THIRD AND FOURTH NORMAL FORMS

Introduction

In this chapter, the issue of data integrity in third and fourth normal form schemes is discussed. First, a definition of data integrity is presented and how both normalized forms played their roles in the issue of data integrity. Second, three maintenance policies, which maintain data integrity within the third normal form scheme, are discussed. These maintenance policies become a crucial issue when the third normal form contains the multivalued dependency which is considered a violation of the fourth normal form. Third, data independence is discussed as a problem of detecting a multivalued dependency. Fourth, a difference between the third and the fourth normal forms is illustrated from the logical view.

Data Integrity

One of the goals of a database management system is data integrity, that is, maintaining the validity of data. Often,

maintaining data integrity can be expensive and difficult to achieve. Normalization is the technique that allows a database management system to sustain data integrity. The third and fourth normal forms were chosen since these forms are widely used during the database system design step.

The third normal form is concerned with single-valued facts. A single-valued fact can be a one-to-one relationship or a one-to-many relationship, whereas the fourth normal form is associated with multivalued facts. A multivalued fact may correspond to a many-to-many relationship among data elements (Kent 1983, 121-123).

When facing a many-to-many relationship among data entities, there are two ways to normalize a data record. One way is to use the third normal form, which utilizes the single data record approach. The other method is to use the fourth normal form, which involves breaking down a data entity into two data records. A record appears in the third normal form "if every field is either part of the key or provides a (single-valued) fact about exactly the whole key and nothing else" (Kent 1983, 121). "Under fourth normal form, a record type should not contain two or more independent multivalued facts about an entity" (Kent 1983, 122). Independence is a very important issue and is discussed later in this chapter.

Maintenance Policies

A maintenance policy becomes crucial for data integrity if a relation in the third normal form contains a multivalued dependency

which is considered a violation of the fourth normal form. This section discusses the possible approaches used in defining various maintenance policies and the selection of the most appropriate maintenance policy. The representation method for a third normal form projection within the database is also introduced.

Consider the following example, extracted from Kent (1983, 122), to illustrate the maintenance policies necessary when violating the fourth normal form. Assume that we have employees within a firm that have different skills and speak various languages. Two many-to-many relationships can be discovered, the first between employees and skills, and the second between employees and languages (Kent 1983, 122). "Under fourth normal form, these two relationships should not be represented in a single data record .... Instead, they should be represented in the two records .... (see Figure 2) (Kent 1983, 122).

EMPLOYEE SKILL LANGUAGE

KEY

A single record representation

EMPLOYEE SKILL

KEY

EMPLOYEE LANAGUAGE

KEY

A two record representation

Figure 2. Data Record Representations.

Source: Kent, W. "A Simple Guide To Five Normal Forms in Relational Database Theory." Communication of ACM 26, no. 2. (1986): 122.

Kent (1983, 122) further discusses that there are three maintenance policies if the fourth normal form is violated, since the violation of the fourth normal form leads to uncertainties as to how to store data records. The three maintenance policies can be used to maintain "two independent multivalued facts in one record" (Kent 1983, 122).

Disjoint Format Policy. The first maintenance policy is called a disjoint format. This format allows a record to be stored as either a skill or a language, but not as both (see Table 1).

Table 1. The Disjoint Format Maintenance Policy.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	
Smith	type	
Smith		French
Smith		German
Smith		Greek

Source: Kent, W. "A Simple Guide To Five Normal Forms in Relational Database Theory." Communication of ACM 26, no. 2. (1986): 122.

The problems associated with the first policy are:

> This is not much different from maintaining two separate record types. We note in passing that such a format also leads to ambiguities regarding the meanings of blank fields. A blank SKILL could mean the person has no skill, that the field is not applicable to this employee, that the data is unknown, or, as in this case, that the data may be found in another record (Kent 1986, 122).

Random Mix Policy. The second maintenance policy is called a random mix. There are three variations of a random mix. The first variation