PLATFORM INDEPENDENT DATABASE MANAGEMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY

AHMET KABARCIK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2006

Title of the Thesis : **Platform Independent Database Management**

Submitted by **Ahmet Kabarcık**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University

[illegible]

Prof. Dr. Yurdahan Güler
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Mehmet Reşit Tolun
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Assist. Prof. Dr. Reza Hassanpour
Supervisor

Examination Date : 15.09.2006

Examining Committee Members

Assoc. Prof. Dr. Nihan Çiçekli	(METU)	N. Çiçekli
Assist. Prof. Dr. Reza Hassanpour	(Çankaya Univ.)	[illegible]
Dr. Abdül Kadir Görür	(Çankaya Univ.)	[illegible]

STATEMENT OF NON-PLAGIARISM

Name, Last Name	:	Ahmet Kabarcık
Signature	:	A. Kabarcık
Date	:	15.09.2006

ABSTRACT

PLATFORM INDEPENDENT DATABASE MANAGEMENT

Kabarcık, Ahmet

M.S.c., Department of Computer Engineering

Supervisor: Assist. Prof. Dr. Reza Hassanpour

September 2006, 50 pages

Functional dependencies of attributes for normalization and the most suitable primary key of the table must be defined to get the best design decisions for the databases, which have randomly inserted values.

This study investigates some probabilistic models over average case complexity for random databases formed by independent random tuples with a common discrete distribution. Mathematical values are tried to find for functional dependency. This study aims to help the designer to find the most suitable minimal key and to give an idea whether normalization is needed by detecting functional dependencies of the attributes of tables automatically.

Keywords: random database, functional dependency, minimal key, average case complexity, entropy, big O notation.

ÖZ

PLATFORMDAN BAĞIMSIZ VERİTABANI YÖNETİMİ

Kabarcık, Ahmet

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. Reza Hassanpour

Eylül 2006, 50 sayfa

Rastgele değerler atanarak oluşturulan veritabanlarında en doğru tasarımı elde etmek için tablolardaki özelliklerin normalizasyon için işlevsel bağımlılığının ve tablonun ana anahtarının belirlenmesi gerekir.

Bu çalışma; her satırı, atanmadan önce, bir muamma olan ve her satırı birbirinden bağımsız olarak atanmış rastgele veritabanlarında ortalama haldeki karışıklığa göre bazı olasılık modellerini inceler. İşlevsel bağımlılık için matematiksel değerler bulunmaya çalışılmıştır. Bu çalışma; en uygun ve en küçük anahtarı bulmayı ve tablolardaki alanların işlevsel bağımlılıklarını otomatik olarak bulup normalizasyonun gerekip gerekmediği konusunda tasarımcıya yardımcı olmayı hedefler.

Anahtar Kelimeler: rastgele veritabanları, en küçük anahtar, ortalama haldeki karışıklık, entropi, big O notasyonu.

ACKNOWLEDGMENTS

Special thanks to my mother, E. Fatma Kabarcık, who has been self-sacrificing throughout my life and I have always been grateful to my father, Mustafa Kabarcık and my brothers, Zübeyir Kabarcık, Haşim Kabarcık, Fahri Kabarcık for their valuable support.

I would like to thank my supervisor Assist. Prof. Dr. Reza Hassanpour for his guidance to me throughout the study of this thesis work.

Thanks to Dr. Nael Salman, Assist. Prof. Dr. Emre Sermutlu and Assist. Prof. Dr. Fahd Jarad for spending their valuable times to help my thesis.

I will always remember the support of my friends at this thesis; Tansel Avkar, Hüseyin Şahin Akbal, Celal Küçükoğuz and Sinan Eş.

At last but not least, I thank to my dear, Arzu Durgun. She has always showed me the positive aspects of all events, also the worst ones and she has always carried my weight with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Database is created to store the data to tables for fetching whenever it is required. There are two basics for the database; optimum storage of the data and operations on that data. Storage component affects the performance and the result of the operations on it.

The primary key of a relational table uniquely identifies each record in the table. Key is an ordinary attribute that guarantees to have unique values (such as student number, where each student is recorded once). Key may be a single attribute or a combination of multiple attributes. If a key is composed from the smallest number of attributes, it is called minimal key. Identification of the records by using keys is fundamental base for the majority of algorithms. Usage of key, beside the search application, is insertion, update, delete and maintaining the tuples.

Functional dependency specifies the relationship between two attribute sets. In a relation, value of first attribute set defines the value of the second one. Functional dependency size is related to modification capability of the attributes.

In relational database design, organizing the combination of attributes to avoid the duplication of data is called normalization. Normalization divides a table into two or more small tables according to functional dependency. Each small table describes a part of whole. Deletion, insertion and update modifications of one part affect only one place. Data is accessed and manipulated quickly and efficiently without having risk for the integrity of the whole. To merge the data of the tables, primary keys are used at algorithms.

The facts mentioned till here are the aspects of the optimum table design. Consequently, choosing unsuitable primary key and the needs of normalization at a table are the problems for the designer. The manager who is looking for the problems of a table has to control the attributes and the contents of all the table. However an automated system for that process can save the time and performance, there is not an enough study for that approach in database design environment. The studies of Demetrovics, Katona, Miklos, Seleznjev and Thalheim [1] construct the basis of this study. Although this work does not have one to one correlation at each point with his studies, they have guided us to gain our results.

1.2 Motivation

A solution for the automatic examine of the tables can be generated by using the entropy values of the attributes. Entropy value needs to get the statistics of the values of attributes. Although worst case complexity can occur at some tables whose tuple size is limited, average case complexity gets to the front at the usage of this approach. Average case complexity avoids from non-redundant tests. The applied processes on these conditions produce a value for functional dependency.

Multiple key can be tested by the values of functional dependencies of attributes. There may be more than one key at a table. To find the number of minimal keys and size of minimal keys, big O and little o notations are studied respectively.

1.3 Objectives of the Study

The current applications of database do not contain a system for automatic test of the tables. These tests aim to expose advices to the designer for normalization and the multiple key as an alternative for the primary key. The goal of this study is to establish that automatic system. Consequently, databases can be updated to the ones that has heuristic support within the database design and more flexible ones and be prevented wasting space of memory and time.

1.4 Organization of the Thesis

This thesis comprises five chapters. Chapter 1 is an introduction to this study which contains background of the study, motivation points of this thesis and the objectives of the thesis.

Chapter 2 begins with the definitions of the fundamental methods which have roles on the creation of our cardinal methods. After the section of the Bonferroni inequality, detailed exposition and the proofs of the cardinal methods, which we use at our thesis, are presented. This chapter is the background of the approach which will be used in the chapter 3.

Chapter 3 contains one section which aims to define how we adapt the methods explained in chapter 2 to our thesis. Two subsections are formed. One for the usage of the methods for normalization and the other one for the usage of the methods for primary key.

Chapter 4 is the application part of the adapted methods of our thesis. The usage of these methods for normalization and primary key reveals . The comments about how to read the meanings of these result tables are made in two subsections.

Chapter 5, the last chapter is the conclusion part which is the summary of all the study and which gives the information of how we reached to our goal.

CHAPTER 2

BACKGROUND

2.1 Relational Database

A database can be understood as a collection related files. The relational database model was a huge step forward, as it allowed files to be related by means of a common field. In order to relate any two files, they simply need to have a common field, which makes the model extremely flexible [2].

The required data for database applications have no fundamental changes for decades. It is noticeable that the data is more stable than the applications on itself. The data held in the storage may have no meaning by itself. It becomes meaningful with the operations on it. For example, the data *"Arzu Durgun"* and *"Ceng 389"* open to make comments. A certain meaning is created when it is the result of a query to get the answers of the following questions; *"Who gives the course of java programming lesson?"* and *"What is the course code for java programming lesson?"*.

However data is meaningful with the operations on it for the last users; operations' simplicity, efficiency, maintainability and flexibility are affected by the organization of the data storage. For example, a table has the following fields; *(studentId, name,* *surname, fatherName, address, phoneNumber, calculusGrade).* It is obvious that this table is not a good organized one. *calculusGrade* has no functional dependency with the others. To get the *calculusGrade*, searching the table which contains *fatherName, address* and *phoneNumber* is not efficient. After the normalization application, the *calculusGrade* field may be kept in a table called *grades* and others are kept as a table called *registry* to give simplicity, efficiency, maintainability and flexibility to the applications of this storage.

Two tables are established down. Tables are related by the fields *cCode* of the first table and *courseCode* of the second one. By this relation, *it can be seen that Operating Systems course is given by Assist.Prof. Oğuz Kabarcık.* By these relations, it becomes enough describing a course *once* in the second table.

Table 2.1: Relational Table 1

cCode	title	name	surname
ceng412	Dr.	Arzu	Durgun
ceng412	Dr.	Nael	Salman
ceng501	Assist.Prof	Oğuz	Kabarcık

Table 2.2: Relational Table 2

courseCode	courseName
ceng389	Java
ceng412	Database
ceng501	Operating Systems

2.2 Functional Dependency

Consider a relation $\Re$ that has two attributes A and B. The attribute B of the relation is functionally dependent on the attribute A. The value of attribute A uniquely determines the value of B and if there were several tuples that have the same value of A then all these tuples will have identical value of attribute B.

Functional dependency does not imply a one-to-one relationship between A and B, although a one-to-one relationship may exist between A and B. For example, let A be *student_number* and be *date_of_birth* [3].

2.3 Sperner Theorem

Sperner system is a set system (F, E) in which no element is contained in another. Formally, If X, Y are in F and $X \neq Y$, then X is not contained in Y and Y is not contained in X.

$$|S| \leqslant \binom{n}{[n/2]} \tag{2.1}$$

$|S|$ is the cardinality of a finite set S [4].

2.4 Random Database

Most of the human activities on computer applications have the major tasks for organizing the collected data for optimum storage and operations on that data. Bank accounting, ticket reservation, student registry at a school, stock registry are some basic examples of that human activities.

Data is stored to 2-dimensional tables under databases. Table has m rows and n columns, where m rows are tuples and n columns are attributes and U={1,...,n} represents all attributes of the table.

$$R = \begin{pmatrix} t_1(1) & \dots & t_1(n) \\ \dots & \dots & \dots \\ t_m(1) & \dots & t_m(n) \end{pmatrix} \tag{2.2}$$

Tuples $t_j(U)$=(t_j(1),...,t_j(n)), j=1,...,m, are rows with domains in D_1x...xD_n, where D_i are i=1,...,n. $\Re$=$\Re$(m,n) is the relation corresponding to R. i.e. $t_j(U) \neq t_i(U)$, $j \neq i$ in $\Re$ [5].

R is said a random database, if tuples $t_j(U)$, j=1,...,m are independent and identically distributed random records with a given discrete distribution $P\{t_j(U) = k(U)\} = P\{k(U)\}$ for $k(U) \,\epsilon\, \prod_{i \epsilon U} D_i$ (cartesian product).

Most of the tables have stochastic type of tuples which means that the existing tuple can not be determined from the previous one. Therefore inserted tuples are always random. This randomization may cause design problems at the hands of inexperienced designers.

The occurrence of such a scenario may be an instance for the problems of random tuples. Two fields of a simply designed table may be *courseName* and *lecturer.* Although this table has no constraints for inserting the data, designer may accept that each course is given by one lecturer. Therefore, table contains distinct courses at each record and *courseName* is accepted as primary key by the inexperienced designer. The relation between *courseName* and *lecturer* is many-to-one. Everything works normally at this condition, until one course is divided into two sections and the course is started to be given by two lecturers. The relation between the fields of *courseName* and *lecturer* becomes many-to-many by the randomly inserted tuple(s). Primary key fails and the structure of the table changes because of the newly inserted random tuple(s).

2.5 Information Theory

Probability is ratio of the occurrence number of an event to the sum of the all events' occurrence number. The probability value for an impossible event is 0 and it is 1 for an inevitable event.

Statistic is a quantity that is calculated from a sample of data. It is used to obtain information about unknown samples of a population.

The word entropy is the amount of disorder of a system. It is the measure of the uncertainty about the realization of a random variable. It is the expected amount of information.

Information theory aliasing as Shannon entropy is the branch of mathematics dealing with the efficient and accurate storage, transmission, and representation of information. The basic concept of entropy in information theory is measuring how much randomness exists in a signal or random event. The philosophy of information theory is "the less you know, the more valuable the information".

For instance, usage ratios of the letters at English language are calculated. The most used letter is 'e', whereas 'z' is the one of the least used ones. In spite of this situation, the characters of the strings are random. The next character can not be predicted, it does have some randomness and entropy is the measurement of this randomness [6].

Information theory defines the entropy in terms of a discrete random event x, with possible states $1..n$ as:

$$H(x) = -\sum_{i=1}^{n} p(i) log_2 p(i) \tag{2.3}$$

For example, a coin is tossed up and \$1 is betted on the flip of coin. If one already has a good guess about the result, then the actual result is less informative. The unit of randomness measurement of the information is taught as bit and the result is;

$$I(\frac{1}{2}, \frac{1}{2}) = (-\frac{1}{2} log_2 \frac{1}{2}) + (-\frac{1}{2} log_2 \frac{1}{2}) = 1 bit \tag{2.4}$$

Meanwhile, the coin may be rigged and it is regulated to come up heads with probability 0.99. Player will bet heads and have expected value \$0.98 for the bet. That means player would only be willing to pay less than \$0.02 for advance information about the outcome of the flip. If the coin were fair as in the upper paragraph, expected value would be zero and player would be willing to pay up to \$1 for advance information. Information for the rigged condition will get the following result;

$$I(\frac{1}{100}, \frac{99}{100}) = (-\frac{1}{100} log_2 \frac{1}{100}) + (-\frac{99}{100} log_2 \frac{99}{100}) = 0.08 bit \tag{2.5}$$

At fair condition, the value of information is 1, although it is 0.08 at rigged condition. This difference is because knowledge increased at rigged condition, then the value of information decreased [7].

2.6 Big O Notation

A theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size n, which is usually the number of items. More precisely, it is used to describe an asymptotic upper bound for the magnitude of a function in terms of another, usually simpler, function. Informally, saying some equation $f(n) = O(g(n))$ means it is less than some constant multiple of $g(n)$.

For instance $4n^2 - 2n + 4$ is $O(n^2)$ because as n grows larger, the term n^2 will be dominant and other coefficients can be neglected.

At the equation $n^2 + 3n + 4 < 2n^2$ for all $n > 10$; $3n + 4$ is $O(n^2)$, too, but big-O notation is often misused to mean equal to rather than less than. The notion of equal to is expressed by (n).

The importance of this measure can be seen in trying to decide whether an algorithm is adequate, but may just need a better implementation, or the algorithm will always be too slow on a big enough input. For instance, quicksort, which is $O(n \log n)$ on average, running on a small desktop computer can beat bubble sort, which is $O(n^2)$, running on a supercomputer if there are a lot of numbers to sort. To sort 1,000,000 numbers, the quicksort takes 6,000,000 steps on average, while the bubble sort takes 1,000,000,000,000 steps! [8].

The big-O complexity of an algorithm is important, but it does not tell the entire story. Any accurate and complete analysis of an algorithm should answer the question of what the real cost of executing the algorithm is for real or typical problems.

Assume that two jobs are offered by two companies to a person. First one is offering to duplicate the salary at every five years. Second one is offering to add \$1000 to the salary per year. Salary is increasing at $O(2^n)$ at first company and it is $O(n)$ at second company.

This information is not enough to choose the best one. The starting salary must be known whether both salaries are same. If the first company starts with the salary \$0, salary never increases. If the starting salary at the second company is much higher than the first company, then even the wonderful raises the first company offers might not actually make any difference before you retire. In this case, n isn't going to get too large, so the behavior of the salary as the person becomes infinitely old is not an issue [9].

2.7 Little o Notation

A theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size n, which is usually the number of items. Informally, saying some equation $f(n) = o(g(n))$ means $f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity. More formally it means for all $c > 0$, there exists some $k > 0$ such that $0 \leqslant f(n) < cg(n)$ for all $n \geqslant k$. The value of k must not depend on n, but may depend on c.

For instance, $3n + 4$ is $o(n^2)$ since for any c we can choose $k > (3 + \sqrt{9 + 16c})/2c$. $3n + 4$ is not $o(n).o(f(n))$ is an upper bound which is not asymptotically tight [10].

2.8 Best, Worst and Average Case Complexities

Best, worst and average cases of a given algorithm determine what the level of resource usage is at least, at most and on average, respectively in computer science. The mentioned resource may be running time or memory, for instance.

However the best case usage is mostly imaginary, average performance and worst case performance are used widely in algorithm analysis. Probabilistic analysis techniques, especially expected value (entropy), are used to determine expected average running times.

Worst case performance analysis is often easier than the average case performance. Determining what the average input and the limitation of the input are generally very hard and sensitive. If the input of average case performance is defined insufficiently, the analysis does not give the wanted result.

Linear search on an array has a worst case performance $\mathrm{O}(n)$, when the algorithm has to check every element, but the average running time is $\mathrm{O}(n/2)$, when the item is around the middle of an array.

Applying insertion sort on n elements. On average, half the elements in an array $A_1 ... A_{j-1}$ are less than an element A_j, and half are greater. Therefore we check half the subarray so $t_j = j/2$. Working out the resulting average case running time yields a quadratic function of input size, just like the worst case running time.

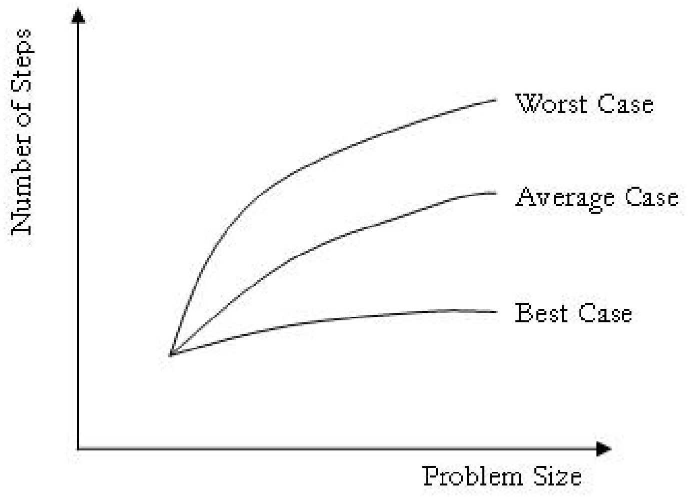


Figure 2.1: Complexity Comparison Diagram

2.9 Poisson Approximation (Stein-Chen Method)

Sometimes calculating binomial probability is hard and long. It is possible to use other distributions to approximate binomial distributions. Poisson approximation is one of them.

Suppose we have an event that occurs over and over like buses stopping at a bus stop. Occurrence of event is X, a time interval is T. The parameter of the poisson distribution is the product of the length T and average length of time between occurrences of event. The distribution of X is discrete, and calculating the probability that X is equal to some particular value X is very simple.

Consider a sequence of independent Bernoulli trials with success probability p. Let $N(n; k_1, k_2)$ denote the number of times that k_1 failures are followed by k_2 successes among the first n Bernoulli trials. We employ the Stein-Chen method to obtain a total variation upper bound for the rate of convergence of $N(n; k_1, k_2)$ to a suitable Poisson random variable. As a special case, the corresponding limit theorem is established. Similar results are obtained for $N_{k_3}(n; k_1, k_2)$, the number of times that k_1 failures followed by k_2 successes occur k_3 times successively in n Bernoulli trials. The bounds obtained are generally sharper than, and improve upon, some of the already known results. Finally, the technique is adapted to obtain Poisson approximation results for the occurrences of the above-mentioned events [11].

2.10 Monte Carlo Method

Any method which solves a problem by generating suitable random numbers and observing that fraction of the numbers obeying some property or properties. The method is useful for obtaining numerical solutions to problems which are too complicated to solve analytically [12].

Interestingly, the Monte Carlo method does not require truly random numbers to be useful. Much of the most useful techniques use deterministic, pseudo-random sequences, making it easy to test and re-run simulations. The only quality usually necessary to make good simulations is for the pseudo-random sequence to appear "random enough" in a certain sense. That is that they must either be uniformly distributed or follow another desired distribution when a large enough number of elements of the sequence are considered [13].

Because of the repetition of algorithms and the large number of calculations involved, Monte Carlo is a method suited to calculation using a computer, utilizing many techniques of computer simulation.

2.11 The Bonferroni Inequality

The Bonferroni inequality is a fairly obscure rule of probability that can be quite useful. By far its most useful application is in joint confidence intervals. The inequality gives you a confidence interval without assuming independence of the various parameters. It usually turns out at around 95% confidence region is not much smaller than with the assumption of independence.

Bonferroni inequality;

$$P(a_1a_2...a_n) \geqslant P(a_1) + P(a_2) + ... + P(a_n) - n + 1 \;\; [14] \tag{2.6}$$

For instance, suppose that we have ten events a_i, with $P(a_i) = 0.99$. We want to estimate the joint probability $P(a_1...a_{10}) = P(a_1)...P(a_{10}) = 0.99^{10} = 0.904382075009$. However we have no grounds to assume independence. If we use the Bonferroni inequality get:

$$P(a_1...a_n) \geqslant P(a_1) + ... + P(a_n) - n + 1 = 10(0.99) - 9 = 0.9 \tag{2.7}$$

2.12 Functional Dependencies in Random Databases

Assume that A and B are the sets of attributes where $A \subseteq U \setminus B$ and $B \subseteq U \setminus A$. Random number of tuples in database R is defined as $N = N(A, B)$ when the functional dependency does not exist, i.e., $t(A) = t'(A)$ and $t(B) \neq t'(B)$. Data distribution of $N(A, B)$ defines the degree of functional dependency between A, B. As a result of this situation, it can be said that P{A→B} = P{N(A,B) = 0}.

For any tuple $t(U)$, denote by

$$p(k(A), k(B)) = P\{t(A) = k(A), t(B) = k(B)\}, \tag{2.8}$$

$$p(k(A)/k(B) = P\{t(B) = k(B)/t(A) = k(A)\} \ [1]; \tag{2.9}$$

i.e., The conditional probability, $p(k(A)/k(B))$ is $t(B) = k(B)$ when $t(A) = k(A)$ and for such models, mean of $N(A, B)$ is calculated as the followings;

$$\begin{aligned}\lambda = \lambda(A, B) = E[N(A, B)] &= M \sum_{k(A),k(B)} p(k(A))(1 - p(k(B)/k(A)))p(k(B), k(A)) \\ &= ME[p(t(A))(1 - p(t(B)/t(A)))] \ [1], \end{aligned} \tag{2.10}$$

$E[\cdot]$ is accepted as the mathematical expectation for the joint distribution $\{p(k(A), k(B))\}$. As an example, for a uniform database, $\lambda = M2^{-a}(1 - 2^{-b})$ where $a = h(A)$, $b = h(B)$, and $M = m(m - 1)/2$. if R is a standard Bernoulli database, then $a = |A|$ and $b = |B|$.

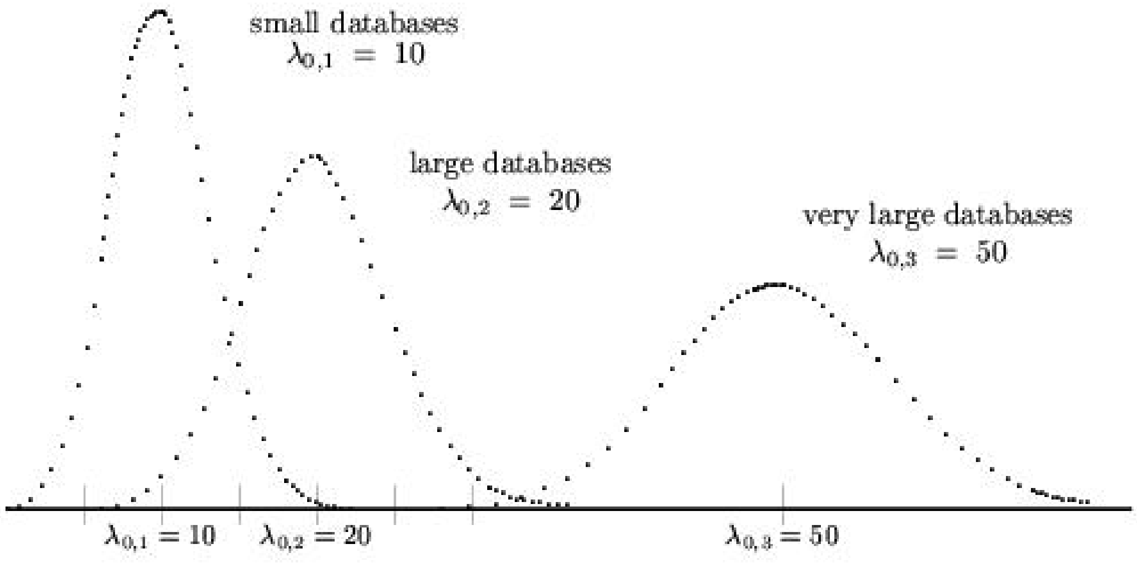


Figure 2.2: Poisson Distribution of Independencies in Different Databases With Different Parameters [15]

Theorem 1. [1] *Let R be a uniform type database and $p(k(B)/k(A)) \leqslant \delta < 1$. Sup-*

pose that $0<\lambda\leqslant ca$; then

$$P\{A \to B\} = e^{-\lambda}(1 + O(2^{-\gamma a})) \quad as\, m \to \infty, \tag{2.11}$$

where $0<c<\frac{1}{2}\ln 2$ and $\gamma = \frac{1}{2}-c/\ln 2>0$.

Corollary 1. [1] Let R be a uniform-type random database. If $a - 2\log_2 m \to \alpha$ and $b \to \beta$, where $\mid \alpha \mid\leqslant +\infty$ and $1 \leqslant \beta \leqslant +\infty$, then

$$P\{A \to B\} \to exp\{-2^{-(\alpha+1)}(1 - 2^{-\beta})\} \quad as\, m \to \infty. \tag{2.12}$$

This asymptotic result can be explained as in the following lines. The size of a database designates the length of the left side or in other words, possible candidates of dependents. That's why, the right side B of a database, which has m tuples, has subset of $F_0(B)$ of all the possible functional dependencies $F(B)$. The λ value is used to determine the set of $F_0(B)$, since the probability P{A→B} $\sim$ $e^{-\lambda}$ as m $\to \infty$. This set is much smaller than $F(B)$. According to the average case, a high probability of functional dependency that is occurred by $F(B)$, which is valid in the random database, is contained by $F_0(B)$. Henceforth, heuristic algorithms like validity checker of functional dependencies from $F(B)$, should begin with the constraints of $F_0(B)$. This condition forms much faster success.

Theorem 1 exposed the importance of the stochastic relations of the attributes for the functional dependency. Stochastic properties of attribute set A and stochastic relationship between the attribute values in A and B define the functional dependency of a random database with sufficiently large m. Also, the data distribution of attribute A defines the factor, and the conditional data distribution of attribute B with a given $t(A)$ characterizes the second factor for calculation of the functional dependency.

2.13 Keys in Random Databases and Relations

2.13.1 Random Databases

The asymptotic results that are mentioned at the upper section for the functional dependency property is very similar with the key probability $P\{R \models A\}$ when $b = h(B) \to \infty$ as $m \to \infty$. A formula for $P\{R \models A\}$ also exists for a uniform random database. Again the entropy value, $a = h(A)$ and average calculation of tuples, $M = m(m-1)/2$ are same as in the functional dependency. The random number of key condition violations are also determined here, $N = N(A) =\mid \{t_i(A) = t_j(A), i, j = 1, ..., m, i < j\} \mid$. The distinguish tuple capability in R of A is characterized by the distribution of $N(A)$. It is obviously seen that $P\{R \models A\} = P\{N(A) = 0\}$ for a set of attributes $A \subseteq U$ and mean number of key condition violations is denoted by the formula, $\lambda = \lambda(A) = E[N(A)] = M \sum_{k(A)} p(k(A))^2 = ME[p(t(A))]$, where $E[\cdot]$ is the mathematical expectation for the distribution $\{p(k(A))\}$.

Theorem 2. [1] *Let R be a uniform-type database.*

(i) *If $0 < \lambda \leqslant ca$, then*

$$P\{R \models A\} = e^{-\lambda}(1 + O(2^{-\gamma a})) \quad \textit{as } m \to \infty, \tag{2.13}$$

where $0 < c < \frac{1}{2}\ln 2$ and $\gamma = \frac{1}{2} - c/\ln 2 > 0$;

(ii) *If R is uniform then $P\{R \models A\} = \prod_{j=1}^{m-1}(1 - j2^{-a}) \leqslant e^{-\lambda}$. If additionally, $0 < \lambda \leqslant 2^{a(2-\gamma)/3}$, where $0 < \gamma < 2$, then*

$$P\{R \models A\} = e^{-\lambda}(1 + O(2^{-\gamma a})) \quad \textit{as } m \to \infty. \tag{2.14}$$

In the uniform case, $\lambda = M2^{-a}$, we also formulate the following:

Corollary 2. [1] *Let R be a uniform-type random database. If $a - 2log_2 m \to \theta\alpha$, where $|\alpha| \leqslant +\infty$, then*

$$P\{R \models A\} \to exp\{-2^{-(\alpha+1)}\} \;\; as\ m \to \infty. \tag{2.15}$$

Remark. [1] Assume that the length of the shortest key in A is determined as $v = v(A)$ i.e. an integer random variable, which equals the length of a minimal key subset B in A. That's why, $\{v(A) \leqslant r\} = \{R \models A\}$ if $|A| = r$. The entropy value is $|A| = a$ for Bernoulli database and $\alpha = a - 2\log_2 m$. It is an obvious result of theorem 2, that if $\alpha \to x/\ln 2 - 1$, then

$$P\{v(A) \leqslant a\} = P\{v(A) - 2log_2 m \leqslant \alpha\} \to exp\{-e^{-x}\} \;\; as\ m \to \infty.\ [1] \tag{2.16}$$

The results, till now, occurred that the size of a shortest key after the normalization of A has asymptotically the Gumbel (double exponential) distribution and the values of the random variable $v(A)$ valued near $2\log_2 m$.

It is observed that keys are more likely in a very small interval. Therefore, to get the fastest algorithms, which search for keys in relations, must check these intervals first.

2.13.2 Random Relations

The only difference at definition between R, which is a random database, and $\Re$, which is a corresponding relation, is that $\Re$ may have identical tuples. Here, $u = h(U)$.

Thus, $N(U) \leqslant N(A)$ and, therefore,

$$P\{R \models A\} = P\{N(A) = 0/N(U) = 0\} = P\{N(A) = 0\}/P\{N(U) = 0\}. \text{ [1]} \quad (2.17)$$

After then, $\lambda(U)/\lambda(A) \asymp 2^{a-u}$ can be established for uniform-type databases. Formulas that are occurred after the main results of asymptotic behavior of key probability can be applied for random relations. Application of Theorem 2 and Eq. (2.17), without trying to have a result for generality results with the following:

Theorem 3. [1] *Let R be a uniform-type and $\Re$ be the corresponding random relation.*

(i) *If $0 < \lambda(A) \leqslant ca$ and $0 < \lambda(U) \leqslant cu$, then*

$$P\{\Re \models A\} = exp\{-\lambda(A)(1 - \lambda(U)/\lambda(A))\}(1 + O(2^{-\gamma a})) \quad as\ m \to \infty. \quad (2.18)$$

where $0 < c < \frac{1}{2}\ln 2$ and $\gamma = \frac{1}{2} - c/\ln 2 > 0$;

(ii) *If $\lambda(A) \to \lambda_0$ and $u - a \to \infty$, then $P\{\Re \models A\} \to e^{-\lambda_0}$ as $m \to \infty$, where $0 \leqslant \lambda_0 \leqslant +\infty$;*

(iii) *If R is uniform, then $P\{\Re \models A\} = \prod_{j=1}^{m-1}(1 - j2^{-a})/(1 - j2^{-u})$.*

Application of the same arguments in Theorem 3, clearly, (2.16) is also valid for random relations if $u - a \to \infty$ *as* $m \to \infty$.

2.14 Minimal Keys

After all, it is time for deriving the main asymptotic result from the previously told results for the minimal key probability $P\{R \models_{min} A\}$ at the set of attributes A.

This situation is an extreme condition and firstly it is studied on standard Bernoulli database. Arguments are similar at the condition of $D_i = \{0, 1, ..., d\}, i = 1, ..., n$, and $d \geqslant 2$. Notations are same with the previous sections and result is as the following:

Theorem 4. [1] *Let R be a standard Bernoulli database and $0 < \lambda_0 < \lambda < ca$, where $0 < c < \frac{1}{2}\ln 2$. Then, there exists $\gamma > 0$ such that for any sufficiently large λ_0,*

$$P\{R \models_{min} A\} = e^{-\lambda}(1 - e^{-\lambda})^a(1 + O(2^{-\gamma a})) \quad \text{as } m \to \infty, \tag{2.19}$$

By Theorem 4, the evaluation of the asymptotic maximum value of the minimal key probability is possible.

Proposition 1. [1] *If $0 < \lambda_0 \leqslant \lambda$, then*

$$P\{R \models_{min} A\} = P(a) \leqslant P_{max}(a) \sim e^{-1}/(a+1) \quad \text{as } m \to \infty, \tag{2.20}$$

and also $P(a) = P_{max}(a)$ iff

$$2\log_2 m - \log_2 \ln(2\log_2 m) - 1 + o(1) \leqslant a \leqslant 2\log_2 m - 1, \tag{2.21}$$

i.e., $\lambda = \ln(a+1)(1 + o(1))$ *as $m \to \infty$.*

The resemblance between the result that is obtained from the Theorem 4 and Proposition 1, and the worst case complexity of minimal key systems can be analyzed now. Thus, in average, a small number of minimal key is occurred without mattering whether the size of m is small or large. Because of this reason, worst-case complexity results are highly different for these cases.

Proposition 1 showed that the probability to be a minimal key for a set of attributes A tends to be 0 as $a \to \infty$. This result is not affected by the length of m or by the size of attribute set A. The importance of this relationship, when the conditional minimal key probability $P(a/K) = P\{R \models_{min} A/R \models A\}$ is considered, is shown at the following corollary.

Corollary 3. [1] *Let* $m^2 = 2^{a+1}(\ln a + d)$, *i.e.,*$\lambda \sim \ln a + d$, *and* $d \to x$, *where* $|x| \leqslant +\infty$,*and also* $d \geqslant \lambda_0 - \ln a$, $\lambda_0 > 0$. *Then*

$$P(a/K) \to exp\{-e^{-x}\} \quad as\ m \to \infty. \tag{2.22}$$

Now it follows directly that if $\lambda = \ln a + d$ and $d \to +\infty$, then

$$P(R \models_{min} A) \sim P\{R \models A\} \sim e^{-\lambda} \quad as\ m \to \infty. \tag{2.23}$$

In addition, same formulation of the main results about asymptotic behavior of minimal key probability can also be applied for the random relations too. It is obvious, $\{R \models_{min} A\} \subseteq \{N(U) = 0\}$ and, therefore,

$$P\{\Re \models_{min} A\} = P\{R \models_{min} A/N(U) = 0\} = P\{R \models_{min} A\}/P\{N(U) = 0\}. \tag{2.24}$$

Applying Theorems 2 and 4 and Eq. (2.24) the following is obtained:

Theorem 5. [1] *Let* $\Re$ *be a random relation and the conditions of Theorem 4 be valid. Suppose* $u = n \geqslant \delta \log_2 m$, *where* $\frac{6}{5} < \delta < \infty$; *then, there exists* $\gamma > 0$ *such that for any sufficiently large* $\lambda_0 > 0$,

$$P\{\Re \models_{min} A\} = exp\{-\lambda(1 - 2^{a-n})\}(1 - e^{-\lambda})^a(1 + O(2^{-\gamma a})) \quad as\ m \to \infty. \tag{2.25}$$

Remark. [1] The results that are obtained at Proposition 1 are also valid for random relations if $n - a \to \quad as\ m \to \infty$.

As a conclusion of until here, assume to have a uniform random database R and $|D_i| \geqslant 2$, for $i \in A$. The asymptotic result of $P\{R \models_{min} A\}$ can also be generalized for this case. $d_i = |D_i| - 2 \geqslant 0$ for $i \in A$. Denote by $\pi_k = |A|^{-1}|\{i : d_i = k, i \in A\}|$ for $k = 0, ..., L$, where $L = max_{i \in A} d_i \geqslant 0$. Note that $\sum_{k=0}^{L} \pi_k = 1$ and $\{\pi_k\}_0^L$ is distribution of values d_i for $i \in A$. Denote by $g(z)$ the *generating function* for the distribution $\{\pi_k\}_0^L$ and whereby $g(z) := \sum_{k=0}^{L} z^k \pi_k$, $|z| \leqslant 1$. In particular, for a standard Bernoulli database, $L = 0$, $\pi_0 = 1$ and $g(z) = 1$ are obtained. Let $\lambda = \ln a + d$, where $a = h(A)$. The following condition is introduced:

$$a e^{-\lambda} g(e^{-\lambda}) = e^{-d} g(e^{-d}/a) \to 0 \;\; as\ m \to \infty. \tag{2.26}$$

Write $\pi_{max} = max_{1 \leqslant k \leqslant L} \pi_k$. Evidently, to ensure (2.26) it is sufficient, if $d \to +\infty$, since $|g(z)| \leqslant 1$ for $|z| \leqslant 1$; or if $e^{-d} max(\pi_0, e^{-d}\pi_{max}/a) \to 0 \;\; as\ m \to \infty$.

Proposition 2. [1] *Let R be a uniform and (2.26) hold. If $\log_2(L + 2) = o(a)$ as $m \to \infty$ and $\lambda \leqslant 2^{\varepsilon a}$, where $0 < \varepsilon < \frac{1}{3}$, then (2.23) is valid.*

In the following simple examples of nontrivial distributions, $L \geqslant 1$.

Example 1 *(Two-steps distribution)* [1]. Let $\pi_k = 0$, $k = 1, ..., L-1$ and $\pi_0 + \pi_L = 1$, $\pi_L > 0$. Then $g(z) = \pi_0 + z^L \pi_L$, and 2.26 holds iff $e^{-d} max(\pi_0 \pi_L e^{-d-L\ln a}) \to 0 \;\; as\ m \to \infty$.

Example 2 *(Binomial distribution)* [1]. Let for any $k = 0, ..., L$, $\pi_k = \binom{L}{k} p^k q^{L-k}$, $p(m) = 1 - q(m) > 0$. Then $g(z) = (q + zp)^L$, and 2.26 holds iff $L \ln(q + e^{-d} p/a) - d \to -\infty \;\; as\ m \to \infty$.

2.15 Proofs

Assume to have $(\Omega, \mathcal{F}, P)$ as a standard probability space. For every $B \in \mathcal{F}$, denote by $\overline{B} := \Omega \setminus B$. Let $N = \sum_{\alpha \in \Gamma} I_\alpha$, where I_α is the indicator of the event C_α and $P\{I_\alpha = 1\} = P(C_\alpha)$, $\alpha \in \Gamma$ and $M = |\Gamma|$. There are classic limit theorems for independent indicator random variables. In general, C_α, $\alpha \in \Gamma$, are dependent and it needs a different technique. The Stein-Chen method has been developed for establishing poisson approximation for sums of dependent indicator variables. One of the result of this approach is used at following lines.

Γ is denoted by $\Gamma_\alpha = \Gamma \backslash \{\alpha\}$, and λ is denoted by $\lambda = E[N] = Mq$. Assume that Γ_α is divided into Γ_α^0, and Γ_α^i such that I_α and $\{I_\beta; \beta \in \Gamma_\alpha^i\}$ are independent. We get $m_0 = |\Gamma_\alpha^0|$.

Proposition 3. [1] Let N be the sum of indicators, $q = P\{I_\alpha = 1\}$, and $E[I_\alpha I_\beta] \leqslant s$ for $\alpha \in \Gamma$ and $\beta \in \Gamma_\alpha^0$. Then

$$|P\{N = 0\} - e^{-\lambda}| \leqslant \lambda/M(m_0 + 1 + m_0 s/q^2). \tag{2.27}$$

following elementary inequality is also used,

$$|\ln(1 + x)| \leqslant 2|x|, \quad |x| \leqslant \frac{1}{2}. \tag{2.28}$$

2.15.1 Functional Dependencies in Random Databases

Proof of Theorem 1. [1] To apply the Stein-Chen method, note that in this case $\Gamma = \{(i, j) : i, j = 1, ..., m, i < j\}$, where $|\Gamma| = M = m(m-1)/2$. Further, write

$N(A,B) = \sum_{\alpha\in\Gamma} I_\alpha$, where I_α is the indicator of the event $C_\alpha = C_{ij} = \{t_i(A) = t_j(A), t_i(A) \neq t_j(A)\}$. The random variables I_α, $\alpha \in \Gamma$, are identically distributed, but dependent. In addition, $\Gamma_\alpha^0 = \{(i,k),(l,j); k \neq j, l \neq i\}$, and $|\Gamma_\alpha^0| = 2(m-1)$ for $\alpha = (i,j)$. Moreover, $P\{I_\alpha = 1 = q\}$ and $E[I_\alpha I_\beta] = s$ for all $\alpha \in \Gamma$ and $\beta \in \Gamma_\alpha^0$. Now, Proposition 3 yields

$$|P\{N=0\} - e^{-\lambda}| \leqslant \lambda/M(2(m-1)+1+2(m-1)s/q^2) = 4\lambda/m(\frac{1}{2}s/q^2). \quad (2.29)$$

Since R is uniform-type, $s \asymp q^2$ is obtained and, therefore,

$$|P\{N=0\} - e^{-\lambda}| \leqslant C_1\lambda/m \leqslant C_2 e^{-\lambda} exp\{\lambda + \ln\lambda - \frac{1}{2}\ln 2a\}, \quad (2.30)$$

where $0 < C_1$, $C_2 < \infty$, and the assertion follows.

Proof of Corollary 1. [1] For a finite value of λ_0 the assertion follows directly by theorem 1 and Eq. (2.30). If $\lambda_0 = 0$ the following estimate can be used:

$$P\{R \models A\} = 1 - P\{N \geqslant 1\} = 1 - P\{\bigcup_{i,j} C_{ij}\} \geqslant 1 - \lambda \to 1 \ \ as\ m \to \infty. \quad (2.31)$$

If $\lambda_0 = \infty$, then for every $\varepsilon > 0$ and $\lambda_\varepsilon > 0$, m_ε is found such that $m_\varepsilon(m_\varepsilon - 1)/2P\{C_{12}\} = \lambda_\varepsilon$. Then for every $n > n_\varepsilon$, $\lambda > \lambda_\varepsilon$, and $m > m_\varepsilon$, then the following is obtained

$$P\{A \to B\} = P\{N=0\} = p(m) \leqslant p(m_\varepsilon). \quad (2.32)$$

Applying now the assertion of the theorem, the following is obtained;

$$P\{A \to B\} \leqslant p(m_\varepsilon) \leqslant e^{-\lambda_\varepsilon} + \varepsilon/2 < \varepsilon \quad (2.33)$$

for every $m > m_\varepsilon$. This completes the proof.

2.15.2 Keys in Random Databases and Relations

The proofs after here is very similar with the ones which are for functional dependencies. Same notation of above is valid; $N(A) = \sum_{\alpha\in\Gamma} I_\alpha$, where I_α is the indicator of the event $C_\alpha = C_{ij} = t\{t_i(A) = t_j(A)\}$. Now the proofs of Theorem 2(i) and Corollary 2 repeat those of Theorem 1 and Corollary 1, respectively.

In the uniform case the following combinatorial argument is used. One can find $2^a = \prod_{i\in A} |D_i|$ variants for the first row $t(A)$ in R. For the second raw, there are $2^a - 1$ variants and so on. There exist 2^{ma} different matrices $m\text{x}|A|$, and whence,

$$p(m) = P\{R \models A\} = \frac{1}{2^{ma}} \prod_{j=0}^{m-1} (2^a - j) = \prod_{j=0}^{m-1} (1 - j2^{-a}) \leqslant e^{-\lambda} \tag{2.34}$$

Applying now (2.28) yields $e^{-\rho m} \leqslant e^{\lambda} p(m) \leqslant 1$, where

$$\rho_m = \sum_{j=1}^{m-1} \frac{j^2 2^{-2a}}{1 - j2^{-a}} \leqslant \frac{2^{-2a}}{1 - m2^{-a}} \sum_{j=1}^{m-1} j^2 \leqslant C_0 \frac{\lambda^2}{m}, \tag{2.35}$$

for some $0 < C_0 < \infty$, and (ii) follows, since $p(m) = e^{-\lambda}(1 + \delta_m)$, with $|\delta_m| \leqslant 1 - e^{-\rho m} \leqslant \rho_m$.

2.15.3 Minimal Keys

Proof of Theorem 4. [1] R be a standard Bernoulli database and whence $|A| = a$. Denote by $A_k = A \setminus \{k\}$, $\mathtt{A}_k = \{R \models A_k\}$, $k = 1, ..., a$, and $\mathtt{A} = \{R \models A\}$. Then it follows directly by the definition of a minimal key that $P\{R \models_{min} A\}$ can be represented in the following form: $\binom{a}{j}$

$$P\{R \models_{min} A\} = P(\mathtt{A}) - \sum_{j=1}^{a} (-1)^{j-1} \binom{a}{j} P\{\mathtt{A}_1, ..., \mathtt{A}_j\} \tag{2.36}$$

In fact, $\{R \models_{min} A\} = \{R \models A\} \bigcap_{j=1}^{a} \overline{\{R \models A_j\}} = \mathtt{A} \bigcap_{j=1}^{a} \overline{\mathtt{A}}_j$. Therefore, $P\overline{\{R \models_{min} A\}} = P\{\overline{\mathtt{A}}\} + P(\bigcup_{j=1}^{a} \mathtt{A}_j)$, since $\overline{\mathtt{A}} \bigcap \mathtt{A}_j = \emptyset$ for $j = 1, ..., a$. Then (2.36) follows by using the inclusion-exclusion formula.

For the proof of the theorem, first the Poisson approximation technique is applied to evaluate the probability $P\{\mathtt{A}_1 ... \mathtt{A}_j\}$; Then the Bonferroni inequality yields the assertion.

The events $\mathtt{A}_k$, $k = 1, ..., a$, are strongly dependent. Hence, the event $E_p = \mathtt{A}_1 ... \mathtt{A}_p$ is represented as follows, $E_p = \bigcap_{i<j} \overline{D}_{ij}$, where $D_{ij} = \bigcup_{l=1}^{p} D_{ij}^{l}$ and $D_{ij}^{l} = \{t_i(A_l) = t_j(A_l)\}$.

Lemma 1. [1] *Let* $1 \leqslant p \leqslant T$ *and* $0 < \lambda < ca$, *where* $1 \leqslant T \leqslant max(ca/\lambda - 1, a)$ *and* $0 < c < \frac{1}{2} \ln 2$. *Then there exists* $\gamma > 0$ *such that*

$$P(A_1 ... A_p) = e^{-(p+1)\lambda}(1 + O(e^{-\gamma a})) \quad as\ m \to \infty. \tag{2.37}$$

Proof. [1] Denote by I_α the indicator of the event $D_\alpha = D_{ij}$, $\alpha \in \Gamma$, and $N(A) = \sum_{\alpha \in \Gamma} I_\alpha$. First consider $P(D_{ij}) = P(D_{12})$, $i, j = 1, ..., m, i \neq j$. $P(D_{12})$ is found by using again the inclusion-exclusion formula. Namely, $|A_1| = a - 1$,

$$P(D_{12}^{l}) = P(D_{12}^{1}) = P\{t_1(A_1) = t_2(A_1)\} = 2^{-(a-1)}. \tag{2.38}$$

If $l_1 \neq l_2$ and $k \geqslant 2$, then $D_{12}^{l} D_{12}^{1} = \{t_1(A) = t_2(A))\}$ and, therefore,

$$P(D_{12}^{l_1} ... D_{12}^{l_k}) = P(D_{12}^{l_1} D_{12}^{l_k}) = P\{t_1(A) = t_2(A)\} = 2^{-a}. \tag{2.39}$$

Thus, the inclusion-exclusion formula yields

$$\begin{aligned} P(D_{12}) &= \sum_{l_1} P(D_{12}^{l_1}) - \sum_{l_1<l_2} P(D_{12}^{l_1} D_{12}^{l_2}) + ... \\ &= \frac{p}{2^{a-1}} - \binom{p}{2}\frac{1}{2^a} + \binom{p}{3}\frac{1}{2^a} + ... + (-1)^{p-1}\binom{p}{p}\frac{1}{2^a} \\ &= \frac{2p+1-p}{2^a} - (1-1)^p \frac{1}{2^a} = \frac{p+1}{2^a}, \end{aligned} \tag{2.40}$$

and, therefore,

$$\lambda_p = E[N(A)] = M(p+1)2^{-a} = (p+1)\lambda. \tag{2.41}$$

To estimate the probability $P(D_\alpha D_\beta)$ when $D_\beta \in \Gamma_\alpha^0$, the inclusion-exclusion formula is used again for the events $C_{kl} = D_{12}^k D_{13}^l$ for $k, l = 1, ..., a$. Then

$$P(D_\alpha D_\beta) = PD_{12}D_{13} = P(\bigcup_{k,l} D_{12}^k D_{12}^l) = P(\bigcup_{k,l} C_{kl}). \tag{2.42}$$

If $k = l$, then

$$P(D_{12}^1 D_{12}^1) = P\{t_1(A_1) = t_2(A_1), t_1(A_1) = t_3(A_1)\} = s_1, \tag{2.43}$$

where $s_1 = 2^{-2(a-1)}$. If $k \neq l$, then

$$\begin{aligned} P(D_{12}^1 D_{12}^2) &= P\{t_1(A_1) = t_2(A_1), t_1(A_2) = t_3(A_2)\} \\ &= P\{t_1(A_{12}) = t_2(A_{12}), t_1(A_{12}) = t_3(A_{12})\} P\{t_1(2) = t_2(2), t_1(1) = t_3(1)\} \\ &= s_1, \end{aligned} \tag{2.44}$$

where $A_{kl} = A \backslash \{k, l\}$, and, therefore,

$$\sum_{k,l} P(C_{k,l}) = (p + 2\binom{p}{2}) s_1 = p^2 s_1. \tag{2.45}$$

Applying again (2.39) yields that if $k_1 = k_2$ or $l_1 = l_2$, then $P(C_{k_1,l_1} C_{k_2 l_2}) = 2^{-2(a-1)-1} = s_1/2$, otherwise $P(C_{k_1,l_1} C_{k_2,l_2}) = 2^{-2(a-2)} = s_1/4$. Let order the set

$\{(k_i,l_i):(k_i,l_i)\neq(k_j,l_j), i\neq j,\ \ i,j=1,...,(p^22)\}$, and $T_i=C_{(k_1,l_1)}$, $i=1,...,(p^22)$. Hence,

$$\sum_{i<j}P(T_iT_j)=\sum_{i<j}P(C_{k_i,l_i}C_{k_j,l_j})=2p\binom{p}{2}s_1/2+(\binom{p^2}{2}-2p\binom{p}{2}s_1/4). \tag{2.46}$$

Finally, the following is obtained;

$$P(D_{12}D_{13})=P(\bigcup_{k,l}C_{kl})=(p-1)^2s_1/4+p^2s_1-p(p-1)s_1\leqslant Cp^2s_1 \tag{2.47}$$

for some $0<C<\infty$. Thus, from proposition 3 and Eq. (2.47) we get

$$|P(E_p)-e^{-\lambda_p}|\leqslant C_1\lambda_p/m, \tag{2.48}$$

where $\lambda_p=(p+1)\lambda\leqslant(T+1)\lambda\leqslant ca$, and $0<c<\frac{1}{2}\ln 2$. The assertion follows now as in Theorem 2(ii).

Further, the asymptotic function is represented in the following inclusion-exclusion form

$$e^{-\lambda}(1-e^{-\lambda})^a=\sum_{j=0}^{a}(-1)^j\binom{a}{j}e^{-(j+1)\lambda}. \tag{2.49}$$

Hence, the Bonferroni inequality implies

$$\begin{aligned}
&|P\{R\models_{min}A\}-e^{-\lambda}(1-e^{-\lambda})^a|\\
\leqslant\ &\binom{a}{T}(e^{-(T+1)\lambda}+P(E_T))+\sum_{j<T}\binom{a}{j}|e^{-(j+1)\lambda}-P(E_T)|\\
\leqslant\ &2\binom{a}{T}(e^{-(T+1)\lambda}+\sum_{j<T}\binom{a}{j}R_j(m)\\
=\ &e^{-\lambda}(1-e^{-\lambda})^a(I_1+I_2)
\end{aligned} \tag{2.50}$$

for every $T=1,...,a$, First consider I_1. Let T be as in Lemma 1 and choose $c_1>0$ such that $T=c_1a/\lambda\leqslant\lfloor(ca/\lambda-1)\rfloor$, and $\lambda\geqslant\lambda_0$. Then, for a sufficiently large λ_0, then $T\leqslant a$ is obtained. Applying (2.28) and Stirling's formula, the following is obtained

$$\begin{aligned} I_1 &\leqslant a^T/T!e^{-\lambda T}(1-e^{-\lambda})^{-a} \leqslant C_1 exp\{T\ln(a/T)+T-\lambda T+2ae^{-\lambda}\} \\ &\leqslant C_2 exp\{ac_1\lambda^{-1}\ln(\lambda/c_1)-ac_1+ac_1/\lambda+2ae^{-\lambda}\} \\ &\leqslant C_3 exp\{ac_1\lambda_0^{-1}\ln\lambda_0-c_1a+ac_1/\lambda_0+ae^{-\lambda_0}\} \leqslant e^{-\gamma_1 a}, \end{aligned} \tag{2.51}$$

for some γ_1 and $C_i > 0$, $i = 1, ..., 3$, and sufficiently large $\lambda_0 > 0$.

In the case of I_2 combining Lemma 1 and Eq. (2.28), the following is obtained

$$\begin{aligned} I_2 &\leqslant (1-e^{-\lambda})^{-a}e^{-\gamma a}\sum_{j=0}^{a} e^{-j\lambda}\binom{a}{j} \leqslant exp\{-\gamma a - a\ln(1-e^{-\lambda})+ae^{-\lambda}\} \\ &\leqslant exp\{-\gamma a+\gamma+3ae^{-\lambda_0}\} \leqslant e^{\gamma_2 a}, \end{aligned} \tag{2.52}$$

for some $\gamma_2 > 0$ and sufficiently large λ_0. Now the assertion follows (2.51) and (2.52).

Proof of Proposition 1. [1] First, let $\lambda > \lambda_0 > 0$, as in Theorem 4, then the following is obtained;

$$P(a) \sim e^{-\lambda}(1-e^{-\lambda})^a \leqslant e^{-1}/(a+1) = P_{max}(a), \tag{2.53}$$

since the $f(x) = x(1-x)^a, x > 0$, has the unique maximum point $x = 1/(a+1)$, i.e. $\lambda = \ln(a+1) + o(1) \quad as\ m \to \infty$. Namely,

$$f_{max} = f(1/(a+1)) = 1/(a+1)(1-1/(a+1))^a \sim e^{-1}/(a+1) \quad as\ m \to \infty. \tag{2.54}$$

If $\lambda > ca$, then $P(a) \leqslant e^{-\lambda} \leqslant e^{-ca}$. Finally, the asymptotic estimates in the assertion can be easily checked.

Proof of Corollary 3. [1] If $m^2 = 2^{a+1}(\ln a + d)$, then $m2^{-a/2} \to \infty$ and

$$\lambda = M2^{-a} = (\ln a + d)(1 + O(1/m)) = \ln a + c \geqslant \lambda_1, \tag{2.55}$$

where $c = d(1+O(e^{-\gamma a}))\ \ as\ m \to \infty$, $\lambda_1 > 0$. The assertion follows now by applying Theorems 2 and 4, since

$$P(a/K) = P\{R \models_{min} A, R \models A\}/P\{R \models A\} = P\{R \models_{min} A\}/P\{R \models A\}. \quad (2.56)$$

Proof of Proposition 2. [1] Denote by $p_i(a) = P\{R \models A_i\}$ and $p(a) = P\{R \models A\}$ for $i \in A$. By the definition of a minimal key as in the proof of (2.36) the following is obtained

$$p(a) \geqslant P(a) \geqslant p(a) - \sum_{i \in A} p_i(a) = p(a)(1+\delta_m). \quad (2.57)$$

To prove the assertion it is sufficient to verify that $\delta_m \to 0\ \ as\ m \to \infty$. Write $a_i = a - \log_2 |D_i|$ and $\lambda_i = M2^{-a_i} = \lambda |D_i|$ for $i \in A$. By assumption,

$$m \sim \sqrt{\lambda} 2^{a/2} \leqslant 2^{(a/2)(1+\varepsilon)} \leqslant 2^{(2/3-\gamma)a}, \ \ \gamma > 0, \quad (2.58)$$

where $a_i \sim a\ \ as\ m \to \infty$. Thus, $\lambda_i \leqslant 2^{\varepsilon_1 a_l}$, $0 < \varepsilon_1 < \frac{2}{3}$, uniformly in $i \in A$. Hence, the claims of Theorem 2 (ii) hold uniformly in $i \in A$. It is observed that there exists $\gamma_0 > 0$ such that

$$\begin{aligned} p_i(a) &= e^{-\lambda_l}(1+O(e^{-\gamma_0 a_i})) = e^{-(d_i+2)\lambda}(1+O(e^{-\gamma_0 a})), \\ p(a) &= e^{-\lambda}(1+O(e^{-\gamma_0 a})), \end{aligned} \quad (2.59)$$

$as\ m \to \infty$ uniformly in $i \in A$. Finally,

$$\begin{aligned} |\delta_m| &= 1/p(a) \sum_{i \in A} p_i(a) = (1+O(e^{-\gamma_0 a}))e^{-\lambda} \sum_{i \in A} e^{-d_l \lambda} \\ &\sim ae^{-\lambda} \sum_{k=0}^{L} e^{-k\lambda}\pi_k = ae^{-\lambda} g(e^{-\lambda}) = e^{-d} g(e^{-d}/a) \to 0\ \ as\ m \to \infty, \end{aligned} \quad (2.60)$$

and the assertion follows.

CHAPTER 3

METHODOLOGY

3.1 Usage of Methods

The methods, told previously, showed that numerical values for normalization and primary key can be obtained by using probability and entropy values.

The results of the methods can be used to establish a system to help the optimization of the tables. While establishing that methods, it must be paid careful attention for the worst and average case complexities, because mostly there is no need to take care for millions of registered data. Instead of checking all data, it is better to test the range, which is defined by little o notation, and obey the constraints which is defined by big O notation.

The formula, $\lambda = M2^{-a}(1 - 2^{-b})$, which is proved at the previous chapter, will be our main gateway to obtain methods for our applications. For calculation of information functions a and b, we are using the formula of Shannon entropy.

Entropy values help to find the degree of functional dependency (λ) between two sets of attributes. By using obtained λ, some basic programs can be established.

One of these programs displays a table, which shows the functional dependency of each field with each of the rest. This table gives the information about whether a normalization is needed.

Another program can contain the list of functional dependency value of each field to the rest of all. Also page contains a list of all fields with checkboxes. By choosing the fields due to some rules and submitting them, λ of the combination can be found. Comparing this λ value with the λ value of the primary key can show us the alternative multiple keys of the primary key.

3.1.1 Calculation of the Functional Dependency (λ)

Calculating the result of $\lambda = M2^{-a}(1-2^{-b})$ gives the functional dependency of one attribute set, whose entropy value is a, to another attribute set, whose entropy value is b.

The formula $\lambda = M2^{-a}(1-2^{-b})$ contains three unknown values; M, a and b. M is the average of the length of tuples which is defined by big O and little o notations and it is calculated by the formula $M = m(m-1)/2$ (*m: number of tuples*). Big O is the segment that affects the result most. Here big O is data segment which has the biggest entropy value. According to little o notation, we must choose a data segment which is bigger than the segment of big O.

For calculation of entropy values a and b; $a = h(A)$, $b = h(B)$, we use the formula $h(x) = -\sum p(i)\log_2 p(i)$. Here the probability value, $p(i)$, is the ratio of the same valued data to all data.

Following slice of a table *(Table 3.1)* is used to explain how λ value is calculated for functional dependency. The functional dependency value of last two fields, *course* and *classroom*, is calculated at the following lines.

Table 3.1: A Table Slice for Calculating the λ

day	course	classroom
Monday	calculus	B307
Tuesday	algebra	B307
Wednesday	algebra	B307
Thursday	algebra	B312
Friday	calculus	B312

course field has 2 kinds of totally 5 data; 2 *calculus* values and 3 *algebra* values. But 2 of the *algebra* values correspond to the same value, *B307*, at *classroom* field. That's why, we take just 1 of them and assume we have 2 algebra values. One is corresponding to *B307* and the other one is corresponding to *B312*. Finally we have totally 4 data for *course* field; 2 *calculus* and 2 *algebra*. Information value for the *course* field is calculated as following;

$$\begin{aligned} h(course) &= -\sum p(i)\log_2 p(i) \\ I(\frac{2}{4},\frac{2}{4}) &= (-\frac{2}{4}\log_2\frac{2}{4}) + (-\frac{2}{4}\log_2\frac{2}{4}) \\ &= 1 \end{aligned} \tag{3.1}$$

classroom field has 2 kinds of totally 5 data; 3 $B307$ values and 2 $B312$ values. Again 2 of the *B307* value correspond to the same value, *algebra*, and we take just 1 of them. Information value for *classroom* field is calculated as following;

$$\begin{aligned} I(\frac{2}{4},\frac{2}{4}) &= (-\frac{2}{4}\log_2\frac{2}{4}) + (-\frac{2}{4}\log_2\frac{2}{4}) \\ &= 1 \end{aligned} \tag{3.2}$$

m is 4 for this condition because both of the fields use totally 4 tuples. M value for this piece of the table is;

$$\begin{aligned} M &= m(m-1)/2 = 4(4-1)/2 \\ &= 6 \end{aligned} \tag{3.3}$$

Functional dependency of *course* field to *classroom* field is calculated as following;

$$\begin{aligned} \lambda &= M2^{-h(course)}(1-2^{-h(classroom)}) \\ &= 6 * 2^{-1}(1-2^{-1}) \\ &= 1.5 \end{aligned} \tag{3.4}$$

3.1.2 Method for Normalization

To analyze a table for normalization, all the fields of the table is evaluated and a functional dependency table (FDT) like multiplication table is established from functional dependency values of each field to each of the rest fields. When we evaluate the table according to the following rules, we can define its normalization needs.

- At an optimal table, which does not need normalization, each entry of columns have near values at FDT.

- The fields, which need to be transferred to another table, have obviously very smaller λ values to each other at their columns at FDT.
- Also the fields, which need to be transferred to another table, may have smaller λ values than the primary key of the table.
- The fields, which gets limited type of data, may be misleading at small sized tables. That's why they can be neglected.

The **second normal form** states that each field in a multiple field primary key table must be directly related to the entire primary key. Or in other words, each non-key field should be a fact about all the fields in the primary key [16]. So our method helps the designer about the application of the second normal form.

3.1.3 Evaluating the Combination of Fields for Multiple Key

If there is not a suitable field to be a primary key, adding a new field, which contains auto increased numbers, can create a primary key but this process increases the size of the table and table uses more space from the memory, however this extra field has ease of usage.

To find an alternative for the primary key, which is created just to be a primary key, designer may define the key from the combination of the table's own fields. Functional dependency value of the combination to the rest must be equal to the value of primary key. At this point, the most important question is *how do we decide which fields will be chosen for the combination of multiple key?* Answer is at the following process;

- A program, coded to find the multiple keys of the tables, displays a table containing functional dependency of each field to the rest. Also program lets the users choosing the component fields for multiple key combination and program displays the functional dependency value of this combination to the rest.

- To choose the components of multiple key, the field which has the smallest λ value is chosen. For other component of the combination, the next smallest value is found.

- Then λ value of the combination to the rest is calculated. If this value is equal to the value of primary key, this combination can be multiple key. If it is not equal, the field, which has the following smallest λ value, is added to the combination.

- If the size of database is too small, the fields, which have limited type of data, may be misleading. That's why such kind of fields may be omitted. For instance, a field called *grade* can only have the values $\{$*1, 2, 3,...,100*$\}$ and can not have another value. So this field is misleading at small sized databases.

Table 3.2 is a very basic table and it will be analyzed for multiple key at the following lines.

Table 3.2: A Test Table Slice for Multiple Key Test

name	surname	username	email	authority
ahmet	kabarcık	ahmet	a.kabarcik@cankaya.edu.tr	editor
ahmet	kabarcık	kabarcik	a.kabarcik@cankaya.edu.tr	referee
ertan	öztürk	ozturert	ozturert@gmail.com	referee
ahmet	kara	ahmet	akara@yahoo.com	referee

This table contains the information of the persons for a journal. The values *editor* and *referee* of the *authority* field define the different user rights. That's why every email can be kept twice but can not match with the same username or authority. On this slice of the table no field can be primary key. But the combination of {*username, email*} has unique values at each tuple and it can be the minimal key of the table. Also the combination of {*username, email*}, {*username, authority*} and {*email, authority*} can be the minimal key of that table.

The fields, which contain limited type of data, can be misleading. So we must make comments about them due to this situation. At this table *authority* field suits to this situation.

Assume we accept the combination of {*username, email*} as minimal key. Let's see its calculation process.

The entropy of *username* field;

$$\begin{aligned} h(username) &= -\sum p(i)\log_2 p(i) \\ I(\frac{2}{4},\frac{1}{4},\frac{1}{4}) &= (-\frac{2}{4}\log_2\frac{2}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) \\ &= 1.5 \end{aligned} \tag{3.5}$$

The entropy of *email* field;

$$\begin{aligned} h(email) &= -\sum p(i)\log_2 p(i) \\ I(\frac{2}{4},\frac{1}{4},\frac{1}{4}) &= (-\frac{2}{4}\log_2\frac{2}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) \\ &= 1.5 \end{aligned} \tag{3.6}$$

The entropy value of the combination of {*username, email*};

$$
\begin{aligned}
h(username + email) &= -\sum p(i)\log_2 p(i) \\
I(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) &= (-\frac{1}{4}\log_2\frac{1}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) \\
&= 2 \qquad (3.7)
\end{aligned}
$$

The entropy of the rest is like the following;

$$
\begin{aligned}
h(rest) &= -\sum p(i)\log_2 p(i) \\
I(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) &= (-\frac{1}{4}\log_2\frac{1}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) + (-\frac{1}{4}\log_2\frac{1}{4}) \\
&= 2 \qquad (3.8)
\end{aligned}
$$

and $M = m(m-1)/2 = 4(4-1)/2 = 6$;

The functional dependency of *username* field to the rest;

$$
\begin{aligned}
\lambda &= M2^{-h(username)}(1 - 2^{-h(rest)}) \\
&= 6 * 2^{-1.5}(1 - 2^{-2}) \\
&= 1.5910 \qquad (3.9)
\end{aligned}
$$

The functional dependency of *email* field to the rest;

$$
\begin{aligned}
\lambda &= M2^{-h(email)}(1 - 2^{-h(rest)}) \\
&= 6 * 2^{-1.5}(1 - 2^{-2}) \\
&= 1.5910 \qquad (3.10)
\end{aligned}
$$

The functional dependency for {*username, email*} field, in other words for our candidate of multiple key;

$$
\begin{aligned}
\lambda &= M2^{-h(username+email)}(1 - 2^{-h(rest)}) \\
&= 6 * 2^{-2}(1 - 2^{-2}) \\
&= 1.1250 \qquad (3.11)
\end{aligned}
$$

We create an imaginary primary key whose values are automatically increased and calculate its λ value. We compare this value with the upper calculated value. If they are equal, this combination can be accepted as multiple key.

As it is seen that (λ) value of the combination of *username* and *email* is smaller than the components. Another result of upper operations may be that *accuracy is direct, performance is indirect proportional with the amount of processed data.*

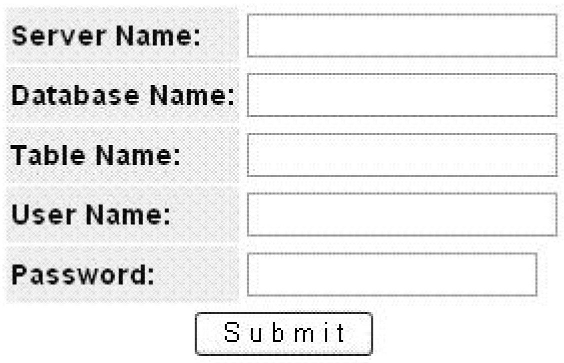
CHAPTER 4

APPLICATION

4.1 Application of Methods

The results that are obtained at the previous chapter are coded to establish an automated system to compile the tables for testing the relations of their attributes. Although the output of these tests have not 100 percentage accuracy for defining the requirements of the table, results are very helpful for the evaluation of the table for the designers who can read these outputs correctly.

To test the methods, we write programs by using PHP language with the database of MySQL on Apache server. Programs are established to be open for all MySQL server users to test their tables on the internet. After submitting the necessary information on the first page *(Figure 4.1)*, the output of the dependency values will display.



Server Name:	
Database Name:	
Table Name:	
User Name:	
Password:	

Figure 4.1: Necessary Information for Testing the Table

4.1.1 Application for Normalization

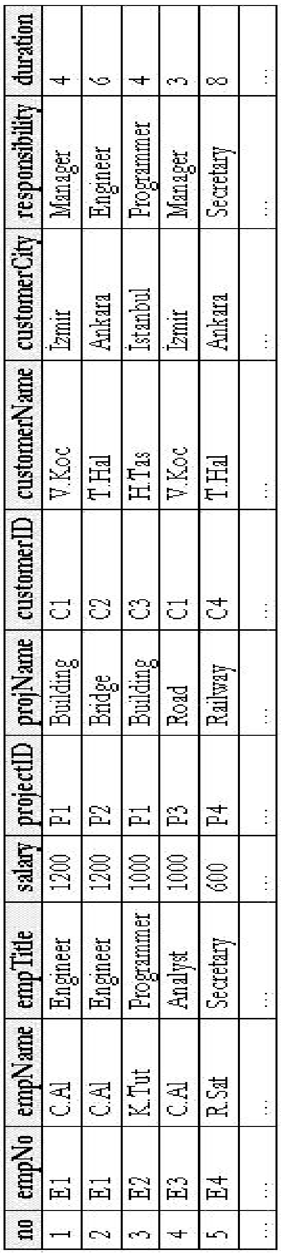
The program, which we write for analyzing the normalization needs of the tables, reveals the value of functional dependency of each field to each of the rest fields. This dependency describes whether the field needs to be transferred to another table or needs to be deleted from the table.

Again our base formulas are $\lambda = M2^{-a}(1-2^{-b})$ and the formula of entropy value, $H(a) = -\sum_{i=0}^{n} p(i) \log_2 p(i)$.

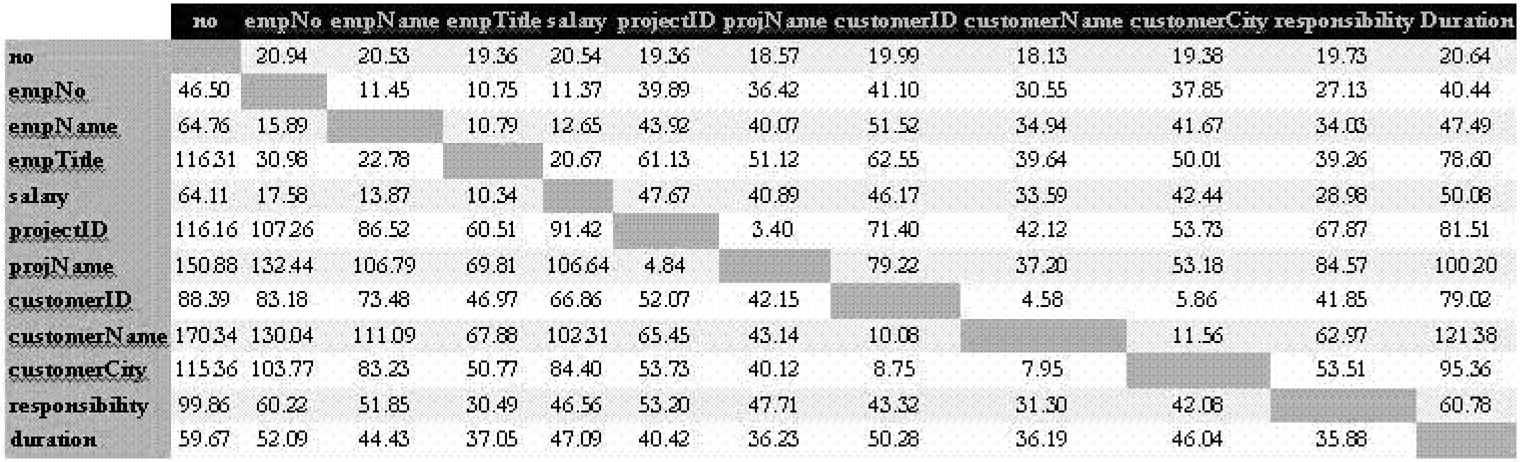
A table called *employee*, whose slice is seen at *(Table 4.1)*, is used to test our results. Table contains these fields; *no, empNo, empName, empTitle, salary, projectID, projName, customerID, customerName, customerCity, responsibility, duration.*

Our program displays a functional dependency table (FDT) like the one at *Figure 4.2*. This table contains the functional dependency values of each field to each of the rest fields. Analyzing process of normalization is like the following;

Table 4.1: A Slice of the *employee* Table for Testing Normalization



no	empNo	empName	empTitle	salary	projectID	projName	customerID	customerName	customerCity	responsibility	duration
1	E1	C.Al	Engineer	1200	P1	Building	C1	V.Koc	İzmir	Manager	4
2	E1	C.Al	Engineer	1200	P2	Bridge	C2	T.Hal	Ankara	Engineer	6
3	E2	K.Tut	Programmer	1000	P1	Building	C3	H.Tas	İstanbul	Programmer	4
4	E3	C.Al	Analyst	1000	P3	Road	C1	V.Koc	İzmir	Manager	3
5	E4	R.Sat	Secretary	600	P4	Railway	C4	T.Hal	Ankara	Secretary	8
...	...	...	...	...	...	...	...	...	...	...	...



	no	empNo	empName	empTitle	salary	projectID	projName	customerID	customerName	customerCity	responsibility	Duration
no		20.94	20.53	19.36	20.54	19.36	18.57	19.99	18.13	19.38	19.73	20.64
empNo	46.50		11.45	10.75	11.37	39.89	36.42	41.10	30.55	37.85	27.13	40.44
empName	64.76	15.89		10.79	12.65	43.92	40.07	51.52	34.94	41.67	34.03	47.49
empTitle	116.31	30.98	22.78		20.67	61.13	51.12	62.55	39.64	50.01	39.26	78.60
salary	64.11	17.58	13.87	10.34		47.67	40.89	46.17	33.59	42.44	28.98	50.08
projectID	116.16	107.26	86.52	60.51	91.42		3.40	71.40	42.12	53.73	67.87	81.51
projName	150.88	132.44	106.79	69.81	106.64	4.84		79.22	37.20	53.18	84.57	100.20
customerID	88.39	83.18	73.48	46.97	66.86	52.07	42.15		4.58	5.86	41.85	79.02
customerName	170.34	130.04	111.09	67.88	102.31	65.45	43.14	10.08		11.56	62.97	121.38
customerCity	115.36	103.77	83.23	50.77	84.40	53.73	40.12	8.75	7.95		53.51	95.36
responsibility	99.86	60.22	51.85	30.49	46.56	53.20	47.71	43.32	31.30	42.08		60.78
duration	59.67	52.09	44.43	37.05	47.09	40.42	36.23	50.28	36.19	46.04	35.88	

Figure 4.2: Functional Dependency Table for *employee* Table

- When we analyze the *projectID* column at FDT, it is seen that the value 4.84 is obviously smaller than the rest. Also it is smaller than the dependency value of primary key, *no*, which is the first entry of the *projectID* column, which is 19.36. The value 4.84 corresponds to the *projName* field at its row and this value is the λ value of *projName* field to *projectID* field. Then we analyze the column *projName* and it is seen that the value 3.40 is very smaller than the rest. 3.40 shows the λ value of *projID* field to the *projName* field. So these 2 fields need to be normalized according to these results.

- *customerID* column at FDT table has 2 values which are smaller than the λ value, 19.99, of *no* field, which is primary key. These values are 10.08 and 8.75 corresponding to *customerName* and *customerCity* fields respectively. *customerName* column has also 2 values which are very smaller than the rest and primary key. These values are 4.58 and 7.95 corresponding to the *customerID* and *customerCity* fields. Also *customerCity* column has 2 small values corresponding to *customerID* and *customerName* fields. So these 3 fields need to be normalized and the primary key of the new table must be kept instead of that 3 fields.

- Functional dependencies of *empNo, empName, empTitle, salary* to each other are also very smaller than the rest and near or smaller than the primary key. That's why they also need to be normalized.

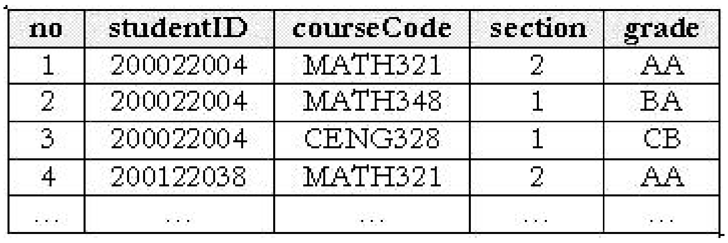
At this situation, our process results with the operation of second normal form. At the beginning, we had 1 table and at the end of the process, we get 4 table.

4.1.2 Application of the Multiple Primary Key

Assume that a table has not got any suitable field for being a primary key and a field is created just to be a primary key. On the other side, combination of more than one field might be an alternative for that primary key. Here, choosing the components of that combination is the most important point.

The rules were defined for multiple keys at the previous chapter. The program, which is coded due to that rules, is applied to the table *studentGrade*, which is seen at *(Table 4.2)*. Program displays 2 figures for our test table, *studentGrade.* First figure *(Figure 4.3)* contains the functional dependency value of each filed to the rest. Second figure *(Figure 4.4)* contains a list of attributes with checkboxes.

Table 4.2: *studentGrade* Table for Analyzing Multiple Key Property

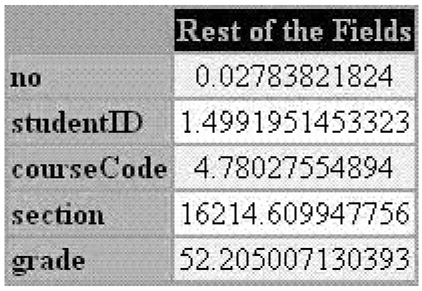


no	studentID	courseCode	section	grade
1	200022004	MATH321	2	AA
2	200022004	MATH348	1	BA
3	200022004	CENG328	1	CB
4	200122038	MATH321	2	AA
...	...	...	...	...

At the program of multiple primary key, again same formulas are used to find functional dependency values. The formulas are; $\lambda = M2^{-a}(1-2^{-b})$ and $H(a) = -\sum_{i=0}^{n} p(i)\log_2 p(i)$.

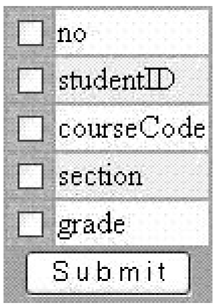
The processes to find the multiple key of the *studentGrade* table is like the following;

- Firstly, we find the attribute, which has the smallest λ value, from *Figure 4.3.* It is seen that 1.4991951453323 value of the *studentID* field is the smallest one.



	Rest of the Fields
no	0.02783821824
studentID	1.4991951453323
courseCode	4.78027554894
section	16214.609947756
grade	52.205007130393

Figure 4.3: Functional Dependency Table of Each Field to the Rest at *studentGrade* Table



☐ no
☐ studentID
☐ courseCode
☐ section
☐ grade

Figure 4.4: Field List of the *studentGrade* Table with Checkboxes

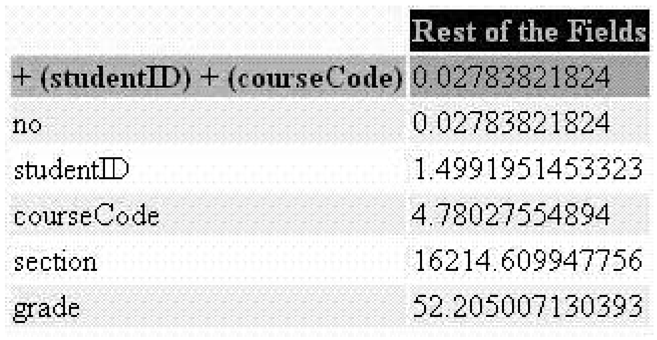
The value of *no* field is omitted because we are looking for its alternative. Then we choose the checkbox next to *studentID* field from *Figure 4.4*. It means, we find the first component of multiple key.

- Then we choose the second field, which has the following smallest value. The following value is *courseCode* field at this table. Then we choose the checkbox next to *courseCode* field from *Figure 4.4*. So we find the second candidate of the components.

- After choosing the checkboxes *studentID* and *courseCode* from *Figure 4.4*, we submit them to find the λ value of the combination of {studentID, courseCode}. If its value is equal to the value of *no* field, which is created just to be a primary key, this combination can be a multiple key. If this is not equal to that value, we add next field according to the rules.

- After the submission, a new λ table occurs like the one at *Figure 4.5*. At this table, it is seen that the value of the combination, {studentID, courseCode}, is equal to the value of primary key. So this combination can be an alternative for primary key, *no*.

- As it is mentioned before the fields, which take data from a limited source, may be misleading at small sized databases. Here the fields *section* and *grade* suit that situation. That's why we would not choose them although if they had suitable values.



	Rest of the Fields
+ (studentID) + (courseCode)	0.02783821824
no	0.02783821824
studentID	1.4991951453323
courseCode	4.78027554894
section	16214.609947756
grade	52.205007130393

Figure 4.5: Functional Dependency Table for Combined Fields and All Fields of *studentGrade* Table

If no λ value equal to the primary key is found after all tests of the combinations, it means there is no multiple key and a primary key has to be created.

CHAPTER 5

CONCLUSIONS

Database systems, formed from unpredictable data, need a system that can eliminate the configuration errors of the attributes of tables and that can find the optimum minimal key. Such a kind of system can increase the efficiency and the flexibility of the database, so wasting time due to the organization will be minimal. It also supplies the best arrangement of the data, so wasting memory space due to data duplications will be minimal too.

There is no parallel studies with this thesis. The studies of J. Demetrovics, et. al. is the main source of this work. They define basics of functional dependency by using the entropy of the data of fields. We use the same way for functional dependency. They have studies to find the length of minimal key but these studies does not define the components of the minimal key. This thesis aimed to find the elements of the minimal key. Although the studies of J. Demetrovics, et. al. does not contain any application for normalization, this thesis contains a system for normalization by using the result of their studies.

REFERENCES

[1] **DEMETROVICS, P., et. al.** (1998), *Asymptotic Properties of Keys and Functional Dependencies in Random Databases*, Theoretical Computer Science 190, 151-156.

[2] http://www.15seconds.com/issue/020522.htm.

[3] http://www.cs.jcu.edu.au/Subjects/cp1500/1998/Lecture_Notes/normalisation/fd.html.

[4] http://en.wikipedia.org/wiki/Sperner%27s_theorem.

[5] **SELEZNJEV, O., THALHEIM, B.** (2002) *High-Dimensional Random Databases*, NordStat2002, Sockholm.

[6] http://en.wikipedia.org/wiki/Information_entropy.

[7] **RUSSELL, S. J., NORVIG, P.** (1995), *Artificial Intelligence*, Prentice Hall Inc., United States of America.

[8] http://www.guides.sk/CRCDict/HTML/bigOnotation.html.

[9] http://www.eecs.harvard.edu/∼ ellard/Q-97/HTML/root/node8.html .

[10] http://www.guides.sk/CRCDict/HTML/littleOnottn.html.

[11] http://projecteuclid.org/Dienst/UI/1.0/Summarize/euclid.jap/1101840553 .

[12] http://mathworld.wolfram.com/MonteCarloMethod.html.

[13] http://en.wikipedia.org/wiki/Monte_Carlo_method.

[14] http://www.itl.nist.gov/div898/handbook/prc/section4/prc473.htm .

[15] **DEMETROVICS, P. et. al.** (1995) *The Average Length of Keys and Functional Dependencies in (Random) Databases* In: *Proc. ICDT95* (Gottlob, G. and Vardi, M., eds). Lecture Notes in Computer Science 893, Springer. Berlin, Germany, 266-279.

[16] http://www.devhood.com/Tutorials/tutorial_details.aspx?tutorial_id=104.

[17] http://en.wikipedia.org/wiki/Big-O_notation.

[18] http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK/NODE13.HTM.

[19] **ATZENI, P. et. al.** (1999), *Database Systems Concepts, Languages and Architecture*, The McGraw-Hill Companies, United Kingdom.

[20] **DATE, C. J.** (2004), *An Introduction to Database Systems*, Addison Wesley, United States of America.

[21] **DEMETROVICS, J. et. al.** (2003), *Recent Combinatorial Results in the Theory of Relational Databases*, European Mathematical Society, FIZ Karlsruhe & Springer-Verlag, Mathematical and Computer Modeling 38, 763-772.

[22] **SELEZNJEV, O.** (2004) *Random Databases with Approximate Record Matching and ϵ-Entropy*, Umea University, Sweden.

[23] **SELEZNJEV, O., THALHEIM, B.** (1998), *Behavior of Keys in Random Databases*, IEEE Computer. In: Proceedings of 18th International Conf. SCCC'98, Chile.

[24] **STAMPER, D., PRICE, W.** (1990), *Database Design & Management An Applied Approach*, McGraw Hill, United States of America.

[25] **YAMANO, T.** (2001), *A Possible Extension of Shannon's Information Theory*, Entropy 3, Köln, Germany, 280-292.

[26] http://databases.about.com/cs/administration/g/primarykey.htm.

[27] http://www.rsolutions.net/RSweb/Normalization/ppframe.htm.

[28] http://databases.about.com/od/specificproducts/a/normalization.htm.

[29] http://www.gslis.utexas.edu/ wyllys/DMPAMaterials/normover.html.