APPLICATION OF TECHNOLOGY ACCEPTANCE MODEL ON DATABASE NORMALIZER

Ahmed Abdulhakim Ahmed Al-Absi

Universiti Utara Malaysia
2011

APPLICATION OF TECHNOLOGY ACCEPTANCE MODEL ON DATABASE NORMALIZER

A project submitted to Dean of the Awang Had Salleh Graduate School of Arts and Sciences in partial Fulfillment of the requirements for the degree Master of Science of Information Technology
Universiti Utara Malaysia

PERMISSION OF USE

ABSTRACT

Normalization is one of the most important activities in database designing. The good database design is the database that meets user requirements and designed its structure carefully. Therefore, this study focused on developing a database normalization application that helps database designers to perform the normalization process automatically and improves the database designing by avoiding the problems of carrying out normalization manually which has many drawbacks such as time consuming, prone to errors and requires more than one skilled user. The main objective of this study is to develop a database normalizer application to normalize the database tables up to the third normal form (3NF). This study provides a normalization algorithm to perform the 1NF, 2NF, and 3NF automatically based on Microsoft Access and SQL Server databases. Experiment was conducted to check the functionality in performing the normalization process. The experiment result showed that the prototype achieved the result successfully as expected and fulfills the requirements and rules of normalization processes. Moreover, a questionnaire based on the Technology Acceptance Model technique has been adopted to ensure of the prototype level in terms of easiness of use, and satisfaction.

Dedication

Specially dedicated to

My beloved father and mother

To my siblings and family

Thanks for all the encouragement and support

ACKNOWLEDGEMENT

Alhamdulilah. All thanks and praise to Allah for giving me the strength to pursue and complete this project.

I would like express my deepest gratitude to my supervisor Assoc. Prof. Abd Ghani B. Golamdin for his support, guidance and ideas given to me throughout this research and for finding time and patience reading my drafts repetitively are very much appreciated.

Many thanks go to my evaluator Dr. Massudi Mahmuddin for his tremendous help in providing me the valuable support, time and feedback are much appreciated.

I am grateful to all lecturers of the College of Arts and Sciences at Universiti Utara Malaysia where I gained a lot of experience, information and knowledge and learnt the most valuable things in the world of research.

Special thanks to my dear friend Mr. Ahmed Talib for his help in giving me valuable ideas and sharing me his experience.

My thanks to Mr. Mustafa Muwafak and SerindIT UUM for their help and giving me the permission to do the interview and the evaluation.

I would like to thank my colleagues and friends who helped me directly or indirectly for the completion of this project.

Finally, my gratitude and love goes out to my family. This project will not have been possible without their help and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

1NF	First Normal Form
2NF	Second Normal Form
3NF	Third Normal Form
4NF	Fourth Normal Form
5NF	Fifth Normal Form
BCNF	Boyce-Codd Normal Form
DB	Database
DBMS	Database Management System
DBNP	Database Normalizer Prototype
ERD	Entity-Relationship Diagram
FD	Functional Dependency
GUI	Graphical User Interface
IT	Information Technology
TAM	Technology Acceptance Model
UML	Unified Modelling Language
UNF	Un-Normalized Normal Form
UUM	Universiti Utara Malaysia

CHAPTER ONE

INTRODUCTION

1.1 Background

Data has become one of the important strategic resources for many organizations from industry, and government. The tradition data resource had been managed by a file processing system that requires no special data management techniques. Now, data has been stored and manipulated through database management systems (DBMS) as the need for information processing has become necessary.

In 1972, Relational databases has been proposed by Dr. Codd as stated in Connolly and Begg (2004) which are widely used in almost commercial applications to store, manipulate and use huge data for a specific enterprises and decision making. The success of relational database modeled for any enterprise is depending on the design of relational schema (Bahmani, Naghibzadeh, & Bahmani, 2008). Process of designing databases is referring to the activities that are related to the design of the database structure for storing and managing end-user data. The good database design is that database which meets all user requirements and designed its structure carefully (Rob & Coronel, 2009). Database design is an essential phase of working with databases where it affects a good DBMS to work poorly with a badly designed database. Therefore, to have a proper database design, database designer should identify exactly the expected use of database such as process of designing a data warehouse database that requires identifying the historical data also designing a centralized database involve using a centralized approach which is differs from that one in distributed database (Rob & Coronel, 2009).

One of the essential steps in designing stage of any relational database is Normalization, which defined as the technique that can used to produce a set of relations with desirable properties for specifying enterprises data requirements (Connolly & Begg, 2010). The goal of normalization is to create a set of relational tables with minimum data redundancy and to avoid insertion and deletion anomalies. In case if the redundant data cannot be avoided in databases, then reducing the redundancy should be the objective for any database designer to achieve with the help of data normalization (Ram, 2008).

Normalization is achieved in steps. Every step has its own name and requirements, these steps names are First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), Fourth Normal Form (4NF) and Fifth Normal Form (5NF). However, the last two normal forms deals with situation that is very rare (Connolly & Begg, 2010).

1.2 Problem Statement

In most of software industries, process of normalization is still done manually which requires more than one experienced user to do normalization (Dongare, Dhabe, & Deshmukh, 2011). In large database enterprises, there is large number of relations that contains many attributes and functional dependencies which involves normalization process. The problem of carrying out normalization manually is in its time consuming, and it’s prone to errors on dealing with large number of attributes (Dongare, Dhabe, & Deshmukh, 2011). Another problem is that performing normalization manually is costly because it requires more than one skilled user in database design. Moreover, according to Yazici and Karakaya (2007) the available commercial database tools are not providing a full solution for normalization and requiring a programming and data structure skills.

1.3 Research Questions

The research questions that represent this study are:

1. What is the requirement to develop a database normalizer application systematically?

1.4 Objectives

1. To develop a prototype tool that can do database normalization systematically.

1.5 Significance of the Study

The developed normalization application is based on a windows application for faster performance, normalization process will generate a normalized relational tables based on relations that been created on a given database after entering table's records. The database normalizer prototype is a contribution to the society especially for users who work on database normalization. This prototype can easily be used by any user with little background on database normalization like database relations attributes and functional dependencies.

1.6 Scope of the Study

The study focused on the first three types of normalization: first normal form (1NF), second normal form (2NF), and third normal form (3NF). This prototype work based on Microsoft Access and SQL Server databases.

1.7 Organization of the Report

This report is divided into six chapters. The first chapter provided an introduction about database designing and normalization. It includes research problem statement, research questions, objectives, significance and scope of the study. Chapter Two is a review of related literatures pertaining to this project which is divided into three sections: introduction to database, database normalization, and database normalization applications. Chapter Three presented the research methodology that been used in this project in details. The fourth chapter discussed the normalization normalizer prototype analysis and implementation. Chapter Five presents the results and findings of the experiment along with the questionnaire. Finally, the conclusions and recommendations for future research are stated in the Chapter Six. The used references along with the required appendices have been attached at the end of this study.

CHAPTER TWO

LITERATURE REVIEW

This chapter presents a brief introduction to database and concept of normalization as well discusses previous literatures that are related to database normalization applications.

2.1 Introduction to Database

Database is a collection of related data that is created and maintained either by a set of applications written specially for that task or by database management system where this database contains a complete description and definition of the database structure and constraints (Srikanth & Sudarshan, 2001). The data in database are represented in form of tables/relations as a set rows/tuples and columns/attributes. The database columns are called attributes and have a name and are ordered in the table. A table contains functional dependencies (FDs) between its attributes. Functional dependency is a constraint between two sets of attributes from the database and describes the relationship between attributes (Connolly & Begg, 2010). The functional dependencies and attributes determine the normal form of the table. For instance, if there are attributes called A and B in a relation called R, we say attribute B is functionally dependent on attribute A (represented as A → B), if each value of attribute A in relation R is associated with exactly one value of attribute B in R as shown in Figure 2.1.

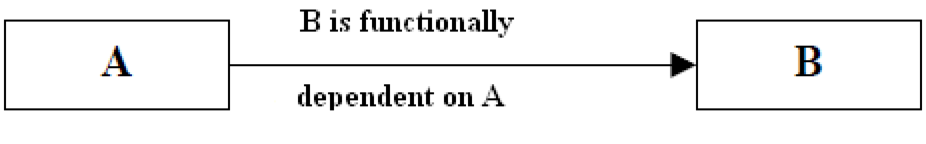
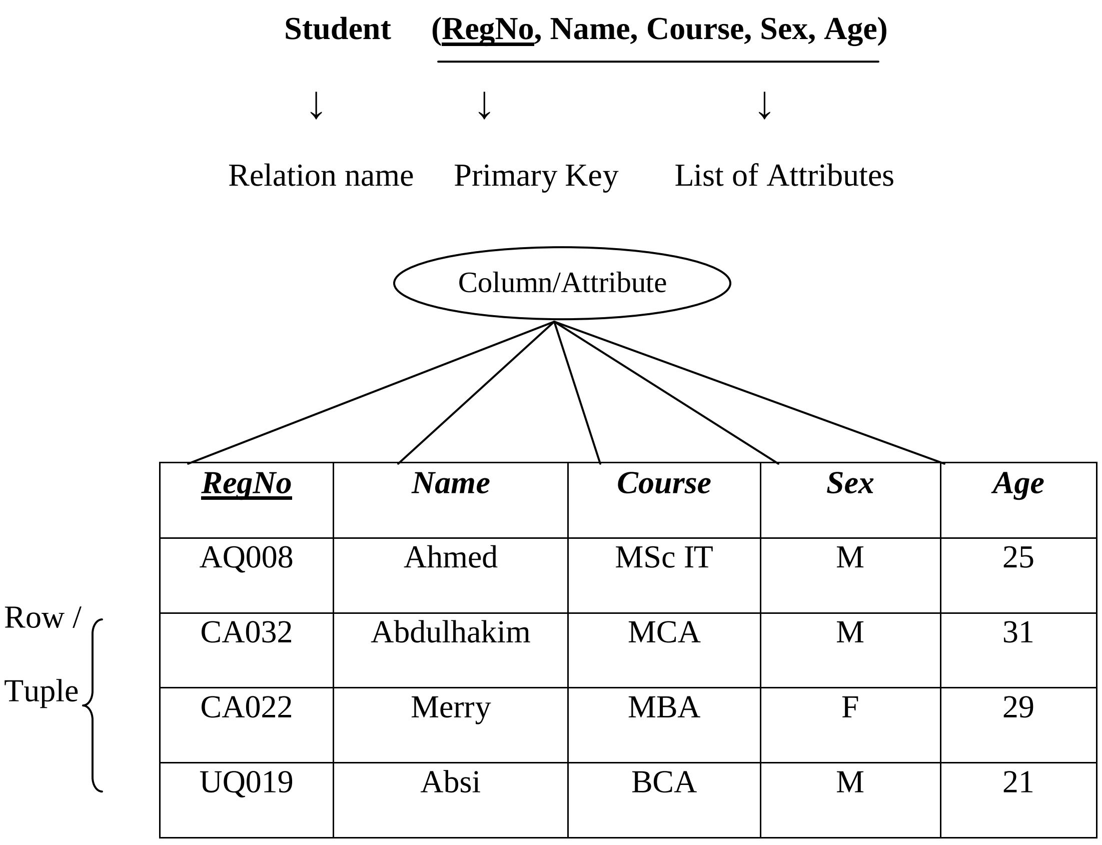


Figure 2.1: Functional dependency diagram.

It is essential in any database relation to have attributes where every row in the relation must be unique to provide enough information. These unique values are known as the key. One relation may have more than one key called candidate key. For example, in the Figure 2.2 below several columns might serve as a key.   Either students register number or students name, both are candidate keys. One of the candidate keys are considered as primary key which is preferred to be register number rather than name because same name is possible for each student. Figure 2.2 below shows an example of a database relation called student contains set of attributes that represents students' information such as RegNo, Name, Course, Sex and Age. To identify the data of this relation every row must has a unique value, in this case RegNo is the primary key.



RegNo	***Name***	***Course***	***Sex***	***Age***
AQ008	Ahmed	MSc IT	M	25
CA032	Abdulhakim	MCA	M	31
CA022	Merry	MBA	F	29
UQ019	Absi	BCA	M	21

Figure 2.2: A student relation example.

2.2 Database Normalization

Normalization is the most applied technique for analyzing database relation based on their primary key and functional dependencies to reduce data redundancy and file storage space of a given relation (Srikanth & Sudarshan, 2001). There are several forms of normalization, in practice, databases are normalized up to third normal form, including BCNF (Bahmani, Naghibzadeh, & Bahmani, 2008). This study focuses on the first three normal forms and not addressing higher order of normalization.

Before starting with first normal form, there is normalization process case called Unnormalized Normal Form (UNF) which refers to a relation that holds one or more repeating groups. This case requires transforming the data from the information sources like form into table format with columns and rows. The output of this process is unnormalized data in form of table (Connolly & Begg, 2010). The next section discusses process of transferring the unnormalized table to 1NF and the following example comes from Connolly and Begg (2010, Chapter 14) which shows the ClientRental relation normalization process for UNF, 1NF, 2NF and 3NF relations. This ClientRental relation is one of five other relations that have been chosen to undergone to experiment to check the functionality of the prototype in fulfilling the requirements and rules of 1NF, 2NF and 3NF. Chapter Five provides more details on the experiment and the chosen relations.

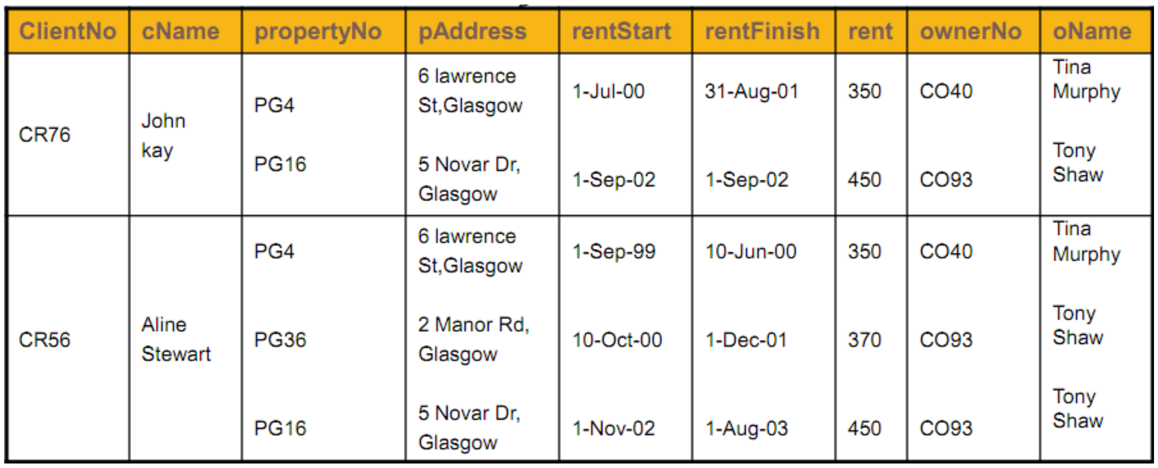
2.2.1 First Normal Form (1NF)

A relation can be in 1NF if every cell on the relation contains exactly one value where all the repeated data has to be removed. According to Connolly and Begg (2010) transforming UNF to 1NF involves the following:

- Choosing table attributes that acts as the key values.
- Identifying the repeating groups in the unnormalized table.
- Removing the repeated group by one of two approaches: Either by inserting data into the empty rows that hold the repeated data or by placing the repeated data in a separate table with a copy of the original key attributes.

In Table 2.1, the data is in unnormalized form because it contains a repeated group for attributes (propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, and oName). Moreover, client named John has two values for propertyNo (PG4 and PG16).

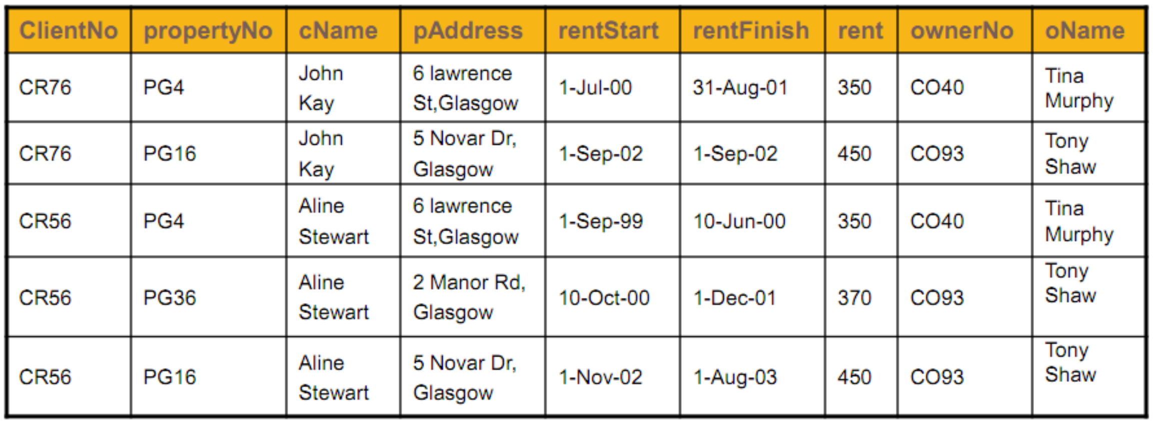
Table 2.1: ClientRental unnormalized data table.



ClientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John kay	PG4	6 lawrence St,Glasgow	1-Jul-00	31-Aug-01	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-02	1-Sep-02	450	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 lawrence St,Glasgow	1-Sep-99	10-Jun-00	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	370	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-02	1-Aug-03	450	CO93	Tony Shaw

Therefore, to transfer Table 2.1 to 1NF, each cell must have a single value by inserting data into each row of CilentRental relation. The output of this process is shown in Table 2.2.

Table 2.2: 1NF ClientRental data table.



ClientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 lawrence St,Glasgow	1-Jul-00	31-Aug-01	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-02	1-Sep-02	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 lawrence St,Glasgow	1-Sep-99	10-Jun-00	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	370	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-02	1-Aug-03	450	CO93	Tony Shaw

2.2.2 Second Normal Form (2NF)

A relation can be in 2NF if it is in 1NF and all non-primary attribute be fully functionally dependent on the primary key. This form requires removing the partial dependencies attributes that are related to only part of the primary key and assign a new table to place the functionally dependent attributes along with a copy of their determinant.

In order to transfer Table 2.2 to 2NF, this requires checking the functional dependencies of ClientRental relation through identifying existence of any partial dependencies on the primary key as following:

- cName is partially dependent on the clientNo primary key.
- pAddress, rent, ownerNo and oName are partially dependent on the propertyNo primary key.
- rentStart and rentFinish are fully dependent on the clientNo and propertyNo primary key.

The output of this process will be creation of three new tables called Client, PropertyOwner, and Rental as in Table 2.3.

Table 2.3: 2NF tables derived from ClientRental data table.

Client

ClientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

ClientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-00	31-Aug-01
CR76	PG16	1-Sep-02	1-Sep-02
CR56	PG4	1-Sep-99	10-Jun-00
CR56	PG36	10-Oct-00	1-Dec-01
CR56	PG16	1-Nov-02	1-Aug-03

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 lawrence St,Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	370	CO93	Tony Shaw

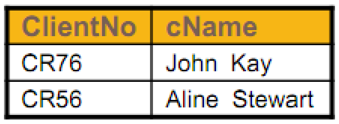
2.2.3 Third Normal Form (3NF)

A relation can be in 3NF if it is in 1NF and 2NF and every non primary attribute is functionally dependent on just the primary key. This normal form is based on the transitive dependency and requires assigning the transitive dependences in a new table along with a copy of their determinant.

From Table 2.3, the propertyOwner table holds transitive dependency where the oName attribute is dependent on ownerNo attribute, so this transitive dependency must be placed in a new table called Owner as in Table 2.4.

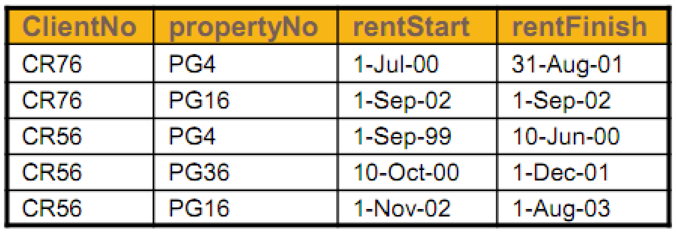
Table 2.4: 3NF tables derived from propertyOwner table.

Client



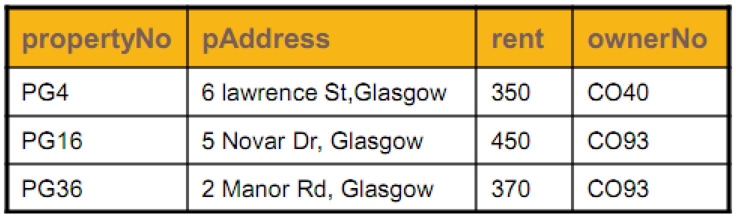
ClientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental



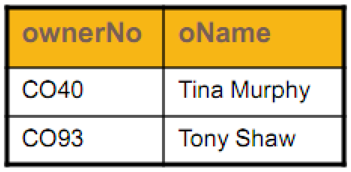
ClientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-00	31-Aug-01
CR76	PG16	1-Sep-02	1-Sep-02
CR56	PG4	1-Sep-99	10-Jun-00
CR56	PG36	10-Oct-00	1-Dec-01
CR56	PG16	1-Nov-02	1-Aug-03

PropertyForRent



propertyNo	pAddress	rent	ownerNo
PG4	6 lawrence St,Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	370	CO93

Owner



ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

2.2.4 Boyce-Codd Normal Form (BCNF)

A relation can be in Boyce-Codd Normal form if every determinant is a candidate key. If a relation has only one candidate key then 3NF and BCNF are equivalent and if not then the functional dependencies must be removed and placed in a new relation.

In fact, the Table 2.4 of 3NF is equivalent to BCNF because Client, PropertyForRent, and Owner tables have a single determinant which is the candidate key while Rental table contains 3 determinants so because of that all tables are candidate keys then in BNCF.

Figure 2.3 below illustrates process of normalization diagram.

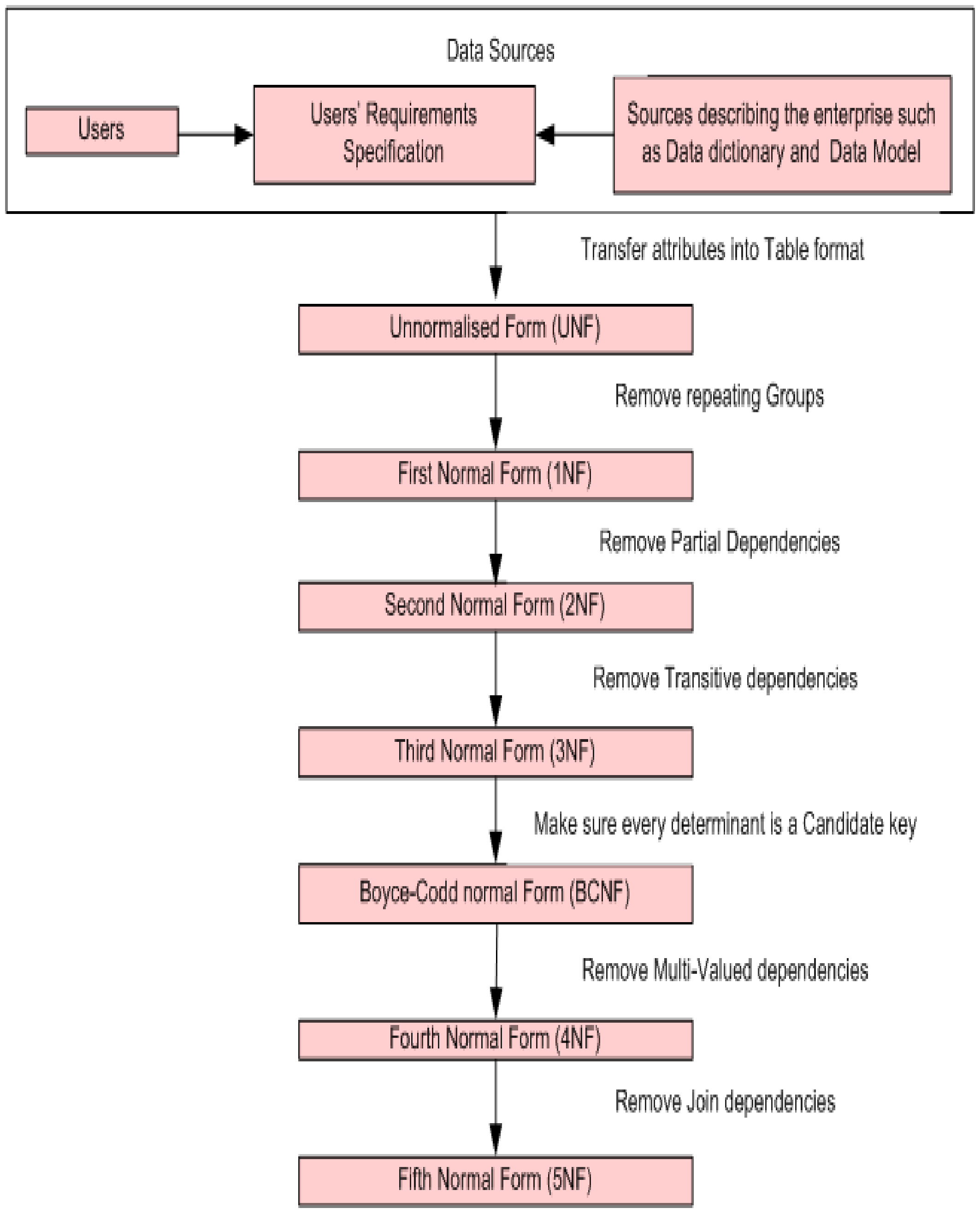
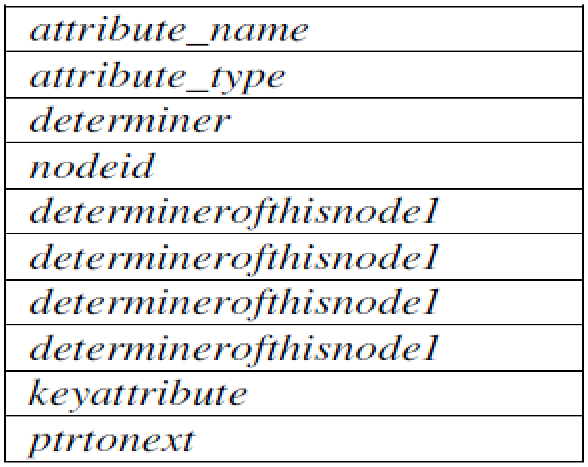

Data Sources
Users
Users' Requirements Specification
Sources describing the enterprise such as Data dictionary and Data Model
Transfer attributes into Table format
Unnormalised Form (UNF)
Remove repeating Groups
First Normal Form (1NF)
Remove Partial Dependencies
Second Normal Form (2NF)
Remove Transitive dependencies
Third Normal Form (3NF)
Make sure every determinant is a Candidate key
Boyce-Codd normal Form (BCNF)
Remove Multi-Valued dependencies
Fourth Normal Form (4NF)
Remove Join dependencies
Fifth Normal Form (5NF)

Figure 2.3: Normalization process diagram (Connolly & Begg, 2010).

2.3 Related Works

To eliminate the drawbacks of the manual normalization process that been mentioned in the problem statement, many researchers already attempted to come out with automation for normalization process.

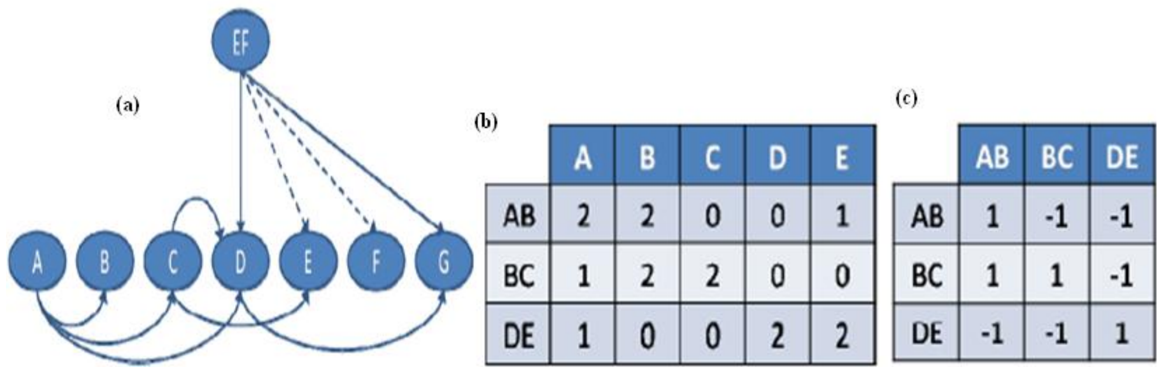
Recently, Dongare, Dhabe, and Deshmukh (2011) proposed a semi-automated normalization tool using single linked list to represent relations as data structure as shown in Figure 2.4. The tool works at schema level for 2NF and 3NF of normalization which means the tool normalizes tables before defining a table and inserting records to it. Although this approach is convenient but still it is not fully automated where user is required to set empty database relations to be normalized manually at schema level. Therefore, in this work, the prototype deals with inputted tables that been already stored with records which mean normalization will be carried out for 1NF, 2NF and 3NF after defining and entering table's records of a given database.



attribute_name
attribute_type
determiner
nodeid
determinerofthisnode1
determinerofthisnode1
determinerofthisnode1
determinerofthisnode1
keyattribute
ptrtonext

Figure 2.4: Linked list Node structure.

Bahmani, Shekofteh, Naghibzadeh, and Deldari (2010) proposed database normalization application using parallel algorithm to compute and reduce time of database normalization process. The proposed application is for 2NF and 3NF normalization process. The process of the algorithm was based on using three data structures: Dependency Graph, Dependency Matrix and Directed Graph Matrix to represent and manipulate dependencies amongst attributes of a relation as shown in Figure 2.5. The proposed application has shown a good result in reducing the time of normalization using parallel algorithm than any other algorithm. However, their scope was limited to 2NF and 3NF only. In addition the application deals with relations that has no repeated data and already normalized in its 1NF. Therefore, in this research, the prototype eliminates the repeated data and performs the 1NF systematically in addition to 2NF and 3NF. According to Hoffer, Prescott, and McFadden (2007) normalization is process of converting complex data structures into simple and stable data structures. Therefore, in this study, the prototype was implemented without using any complex data structures.



(b)

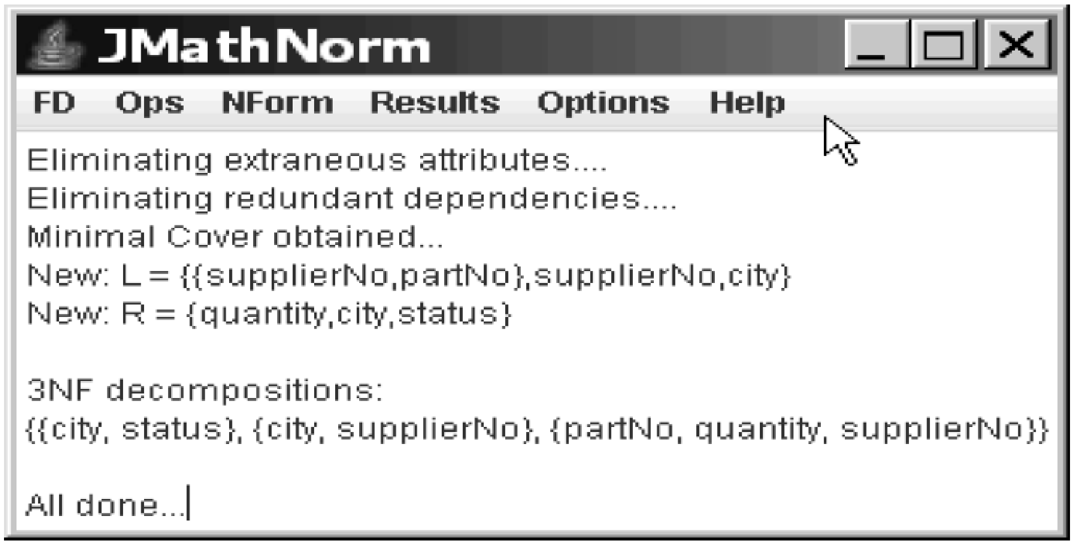
	A	B	C	D	E
AB	2	2	0	0	1
BC	1	2	2	0	0
DE	1	0	0	2	2

(c)

	AB	BC	DE
AB	1	-1	-1
BC	1	1	-1
DE	-1	-1	1

Figure 2.5: (a) Dependencies graphical representation. (b) Dependency Matrix. (c) Directed Graph Matrix. (Bahmani et al., 2010)

Yazici and Karakaya (2007) proposed another automatic normalization tool called JMathNorm designed using built-in functions provided by Mathematica with user designed interface using Java language**.** The important aspect in this tool is that it provides facility to normalize relations up to Boyce-Codd Normal Form (BCNF). However, normalization process of this tool is represented by the symbolic nature of Mathematica as shown in Figure 2.6. Moreover, JMathNorm tool does not provide facility of table creations for the normalized schema of any DBMS. Therefore, in this research, the prototype generates normalized relations in form of structured tables of a given database depending on a DBMS.

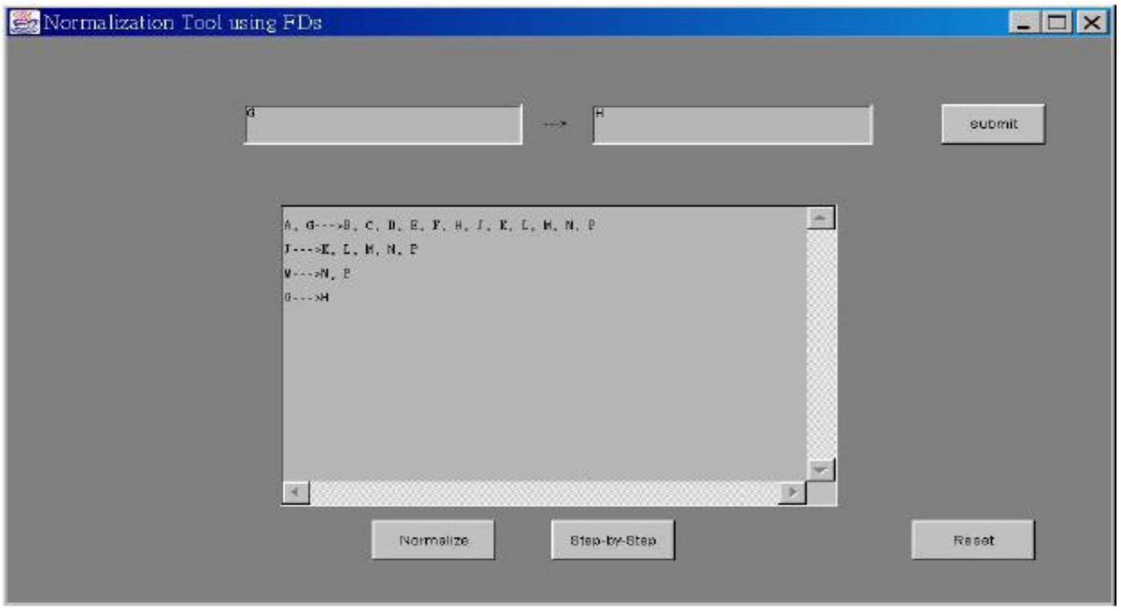


```
Eliminating extraneous attributes....
Eliminating redundant dependencies....
Minimal Cover obtained...
New: L = {{supplierNo,partNo},supplierNo,city}
New: R = {quantity,city,status}

3NF decompositions:
{{city, status}, {city, supplierNo}, {partNo, quantity, supplierNo}}

All done...
```

Figure 2.6: A screenshot run of JMathNorm tool for 3NF decomposition.

According to Yazici and Karakaya (2007) the available commercial database tools are not providing a full solution for normalization and requiring a programming and data structure skills. One of these tools is performing the normalization through Metamodelling in which the Unified Modelling Language (UML) is used to access Object Constraint Language (OCL) to construct expressions that encode Functional Dependency's using classes at a meta-level (Akehurst, Bordbar, Rodgers, & Dalgliesh, 2002). In the study, the process of automation is achieved by using two declarative specifications over the UML meta-model.

Kung and Tung (2006) proposed a web-based normalization tool using Java applet in which it can be accessed through the Internet. The tool developed for purpose of enhancing teaching and learning of database normalization to students. The study had evaluated the students' perceptions of the tool through a questionnaire for 45 students. The result of evaluation found the tool easy to use and the step-by-step feature helped students gain understanding of database normalization process. However, this tool does deal with functional dependencies in form of symbols rather than a structured database relation as shown in Figure 2.7. The tools do not provide visual aid for normalization. Another issue is that developing a database tool in a web-based platform is not recommended because web application performance is slower than windows application especially in dealing with huge database relations (Bhavsar, 2008). Therefore, in this research, the developed prototype will be based on a windows application for faster performance and normalization process will be carried out based on relations that been created by user on a database as a structured tables where the prototype will generate a normalized relations for the given database.



```
A, G--->B, C, D, E, F, H, J, K, L, M, N, P
J--->K, L, M, N, P
M--->N, P
G--->H
```

Figure 2.7: A screenshot of the main window of the web-based normalization tool.

Mitrovic (2002) also developed a web enabled data normalization tutor called NORMIT. NORMIT is a system proposed to teach university students concept of database normalization and solve problems. The student is required to register first where the system provides the student a general description about the system and data normalization then student has to select a predefined normalization problem to normalize from list. NORMIT involves the student to determine prime attributes, candidate keys, compute the set of attributes closure, simplify functional dependencies, and the normal form the table is in. However, these requirements made the approach to be difficult for learners.

2.4 Summary

This chapter presented a brief introduction to database in terms of database tables and functional dependencies, moreover discussed in details the concept of normalization along with its rules and requirements till the third normal form. The chapter discussed as well the previous literatures that were related to database normalization application. The next chapter presents the research methodology that defines the layout phases of this study to achieve the objectives.

CHAPTER THREE

RESEARCH METHODOLOGY

In order to complete and answer the research questions of this study, researcher used a research methodology that defines the layout of the research phases. According to Hoffer, George, and Valacich (2002) the best methodology is that which ensures a reliable approach in all research phases and facilities achieving the identified objectives easily. The research approach of this study is inspired from the General Methodology for Design Research (GMDR) which is advocated by Vaishnavi and Kuechler (2008). Applying this methodology helps in producing the proposed prototype in a better quality and achieves the objectives perfectly.

The methodology has five phases as shown in Figure 3.1:

i. Problem Understanding.

ii. Prototype Design.

iii. Prototype Development.

iv. Experiment phase.

v. Prototype Evaluation.

The study carried out in several steps as illustrated in Figure 3.1 This methodology has been carefully designed to make sure that it is flexible and more suitable for developing the database normalization prototype. The aims of this study accomplished when the prototype is developed and evaluated.

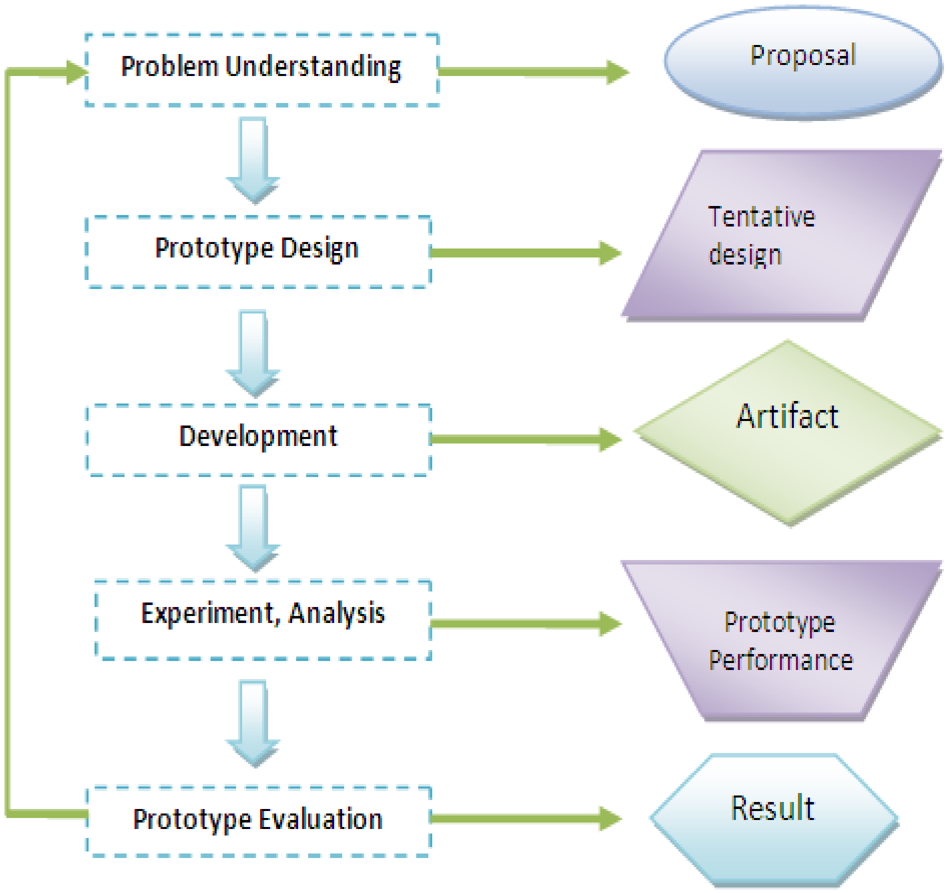


Figure 3.1: Methodology flowchart of the database normalization prototype.

3.1 Problem Understanding

The first stage of this methodology is to understand the objectives and scope of this study. This stage required understanding and analyzing the problem in order to define the problem (Whitten, Bentley, & Dittman, 2001). Therefore, this case required a clear understanding of normalization concept in all of its forms (1NF, 2NF, & 3NF) along with database relations background. The researcher checked the previous related work on the normalization concepts and tools and gathers required information through interview as a primary data sources and from secondary data sources as well. The aim of the interview was to explore how do IT company’s database designers achieve normalization process and what is the most used types of normalization process along with its importance to projects work. The interview (using unstructured interview with open ended questions) was carried out in IT Company called SerindIT at Sintok city and conducted with three employees who were working as Database designer, Database administrator and software programming and developer. The interviews generally produce riches detailed data about a much smaller number of people where interviewees should be knowledgeable and experienced on the topic they provide information on (Patton, 2002; Rubin & Rubin, 2005). Therefore, the participants of the interview were chosen to provide more information and knowledge. According to the information that gathered during the interview, the conclusion was that SerindIT company developers are performing the normalization process manually with the help of Entity-relationship diagram approach (ERD) Moreover, the employees go up to the 3NF and stated that the 4NF doesn’t occur. Appendix A provides more details on the interview questions and summary of participants responses.

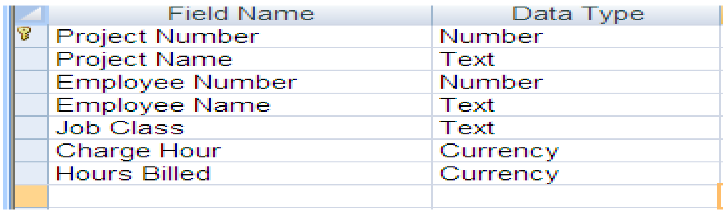
The secondary data sources were also used in this study such as standard texts books like (Connolly & Begg, 2010; Teorey et al., 2011; Srikanth & Sudarshan, 2001), articles, journals, proceedings and websites. The gathered data have been analyzed and helped the researcher to have a clear idea and more understanding of normalization process.

3.2 Prototype Design

This phase started by identifying the inputs/outputs and the functional system requirements for the proposed prototype. The prototype has been designed using the Rational Rose 2010 software to construct the Unified Modeling Language (UML) diagrams and design the use case diagram along with sequence diagrams and class diagrams as shown in Chapter Four. The prototype designed process carried out carefully according to the prototype needs.

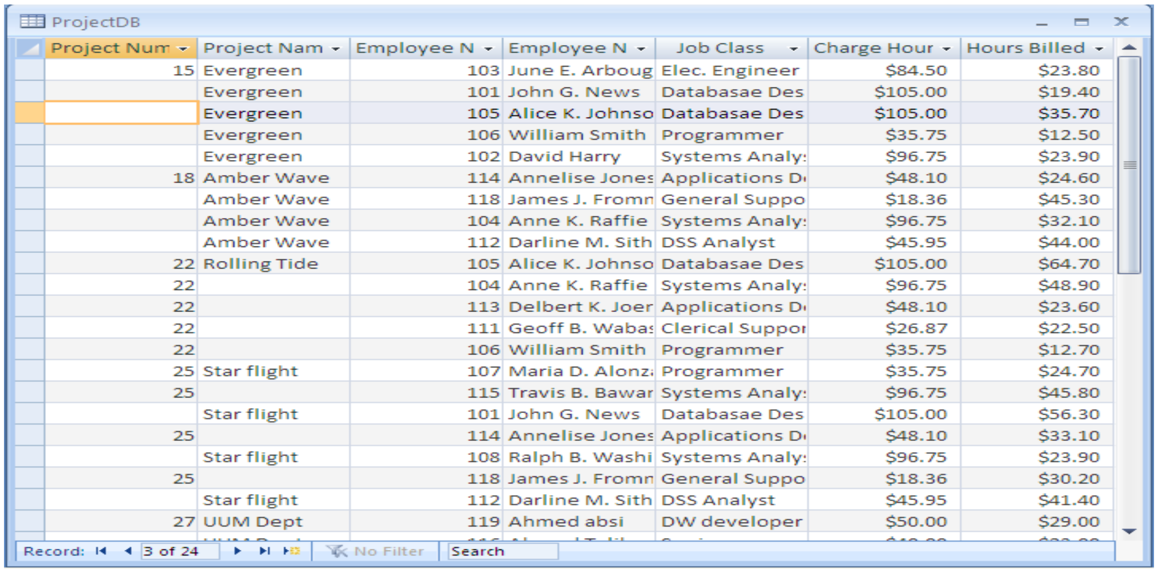
3.2.1 Data Input (Unnormalized form)

The process of normalization in this prototype involves the user to specify and retrieve the required database sources to be normalized in a form of structured database relation from Microsoft Access as in Figure 3.2 and Figure 3.3. This stage of design identifies the Unormalized Form (UNF) where the attributes of the retrieved relation are all in a single relation. The prototype will check the redundancy of the inputted relation and remove the repeating groups. The prototype identifies the retrieved database structure in terms of relations, attributes, data types, and size. The following shows example of database structure of a database relation called Project used in this prototype. Details of all the used databases relations of this study are available in Chapter Five.



Field Name	Data Type
Project Number	Number
Project Name	Text
Employee Number	Number
Employee Name	Text
Job Class	Text
Charge Hour	Currency
Hours Billed	Currency

Figure 3.2: Structure of Report relation.



ProjectDB

Project Num	Project Nam	Employee N	Employee N	Job Class	Charge Hour	Hours Billed
15	Evergreen	103	June E. Arboug	Elec. Engineer	$84.50	$23.80
	Evergreen	101	John G. News	Databasae Des	$105.00	$19.40
	Evergreen	105	Alice K. Johnso	Databasae Des	$105.00	$35.70
	Evergreen	106	William Smith	Programmer	$35.75	$12.50
	Evergreen	102	David Harry	Systems Analy:	$96.75	$23.90
18	Amber Wave	114	Annelise Jones	Applications D	$48.10	$24.60
	Amber Wave	118	James J. Fromn	General Suppo	$18.36	$45.30
	Amber Wave	104	Anne K. Raffie	Systems Analy:	$96.75	$32.10
	Amber Wave	112	Darline M. Sith	DSS Analyst	$45.95	$44.00
22	Rolling Tide	105	Alice K. Johnso	Databasae Des	$105.00	$64.70
22		104	Anne K. Raffie	Systems Analy:	$96.75	$48.90
22		113	Delbert K. Joer	Applications D	$48.10	$23.60
22		111	Geoff B. Wabas	Clerical Suppor	$26.87	$22.50
22		106	William Smith	Programmer	$35.75	$12.70
25	Star flight	107	Maria D. Alonz	Programmer	$35.75	$24.70
25		115	Travis B. Bawar	Systems Analy:	$96.75	$45.80
	Star flight	101	John G. News	Databasae Des	$105.00	$56.30
25		114	Annelise Jones	Applications D	$48.10	$33.10
	Star flight	108	Ralph B. Washi	Systems Analy:	$96.75	$23.90
25		118	James J. Fromn	General Suppo	$18.36	$30.20
	Star flight	112	Darline M. Sith	DSS Analyst	$45.95	$41.40
27	UUM Dept	119	Ahmed absi	DW developer	$50.00	$29.00

Record: 3 of 24 No Filter Search

Figure 3.3: Sample of relation in UNF to be normalized by the prototype.

3.2.2 Normalization Algorithms

The normalization algorithms of 1NF, 2NF and 3NF of this prototype are depending on a table that is in unnormalized form and contains a collection of records. The algorithms check the matched values and normalize the given table to 1NF, 2NF, and 3NF.

This prototype has been designed based on the requirements and rules of each normalization forms process that been discussed and illustrated in the literature review for Connolly and Begg (2010) where if each row contains single value then it is in 1NF. The table has removed partial dependency then it is in 2NF. Lastly, eliminating the transitive functionally dependency then it is in 3NF.

3.2.2.1 Select database table function

The Select Database Table function is used to read all the tables that are available in a selected database. This function works as the prototype input function that inputs the unnormalized table to be normalized. The function ignores reading system files and temporary files. The output of this function is displaying the contents of the selected table in a grid. This can be represented in pseudo code as follows:

```
Step 0: Open database for input;
Step 1: Read tables of the database;
Step 2: Set database Open (filename);
Step 3: Initialize table definitions and table name type of String;
Step 4: For each table definitions in database (td,i=0,…,n) do Step 5
        Step 5: Assign table name by table definitions.Name;
Step 6: Ignore system tables and temporary tables;
Step 7: Display RecordSources in DBGrid;
```

Algorithm 1: Pseudo code of selecting database tables.

3.2.2.2 Select Primary Key Function

This function is used to allow user to specify the primary keys of the select table. Process of this requires user to select table first and then according to the selection, all the fields of the table will be shown to user in form of list to allow user to select the primary keys of the given table. The output of this process is two lists: first holds the table's fields and the other list hold the select primary keys list. The following algorithm describes the process:

```
Step 0: Initialize tabledefinition, field, and fieldname type of String;
Step 1: Check if database table is selected,
          IF selected, do Step 2-6
          Step 2: Set database Open (filename);
          Step 3: Set table definitions (MainTableName)
          Step 4: For each field in table definitions. Fileds
                    Do Step 5-6;
                    Step 5: Assign field to fieldname;
                    Step 6: Add fieldname to list
          IF not selected do Step 7
          Step 7: Display select table message;
```

Algorithm 2: Pseudo code of selecting table's primary keys.

3.2.2.3 First Normal Form Algorithm (1NF)

Process of transferring the selected table from unnormalized status to INF requires from the user to specify the primary keys of the given table. The output of this algorithm is creating a new table with same name of the table name prefixed by "1NF". The algorithm applies the rules of the first normal form by assuring that each field in the selected table does not has empty values and contains single value. According to these rules the algorithm will check each row in the selected table and fill up the required empty fields automatically. So by this, the rules of 1NF are applied successfully and the table becomes ready to move to 2NF. A flag (NF1) is used to indicate the result of 1NF to be used in the rules of 2NF algorithm. The following algorithm describes the process:

```
Step 0: Check selection of primary key
        IF selected, do Step 1-4
        Step 1: Create new relation with name ("1NF" + MainTableName)
        Step 2: Initialize one-dimensional Array (AA) type of String for each field
        Step 3: Check if RecordSource not in EndOfFile then do Step 4-9;
            Step 4: For each Field in RecordSource (field i, i=0,…,n) do Step 5
                Step 5: IF fields are text and
                    IF fields value not null and single value do Step 6
                    Step 6: Return Array fields
                    Else do Step 7
                    Step 7: Edit RecordSource;
                    Step 8: Fill empty fields;
                    Step 9: Update RecordSource;
            Step 10: IF EndOfFile do step 11
                Step 11: Close RecordSource
                Step 12: Display INF successful
                Step 13: Set NF1= True
        IF not selected, do Step 14
Step 14: Display message select primary key first
```

Algorithm 3: Pseudo code of 1NF algorithm.

3.2.2.4 Second Normal Form Algorithm (2NF)

The 2NF algorithm works depending on the result of the 1NF, the flag (NF1) indicates that the table is in INF and ready to be normalized according to 2NF rules. The inputs of 2NF algorithm are the created table of the 1NF. The algorithm checks the functional dependencies of the fields on the selected primary keys. The two-dimensional array of type string is used to hold rows and columns of the table, the purpose of using string is to allow accepting all types of data types. A function Canseparate is used to separate the 1NF table into sub tables according to their fields' dependencies on primary keys. The algorithm creates the new sub tables with the same name of the original table prefixed by "Table $_n$ _". The separation is done according to the similarities of the fields' values. After creating the new tables the algorithm eliminates the redundant data. So by this, the rule of 2NF is applied successfully and the new created tables become ready to move to 3NF. A flag (NF2) is used to indicate the result of 2NF to be used in the rules of 3NF algorithm. The following algorithm describes the process:

```
Step 0: Initialize two-dimensional Array type of String
Step 1: Initialize Primarykey, Recordset, and Myindex as integer
Step 2: keyInfo As String, otherInfo As String, CanSeparate As Boolean
Step 3: IF NF1 is false then do Step 4
            Step 4: Display message not in 1NF
        Else do Step 5 to
            Step 5: Open Recordset (“1NF” & MainTableName)
            Step 6: Get index of the selected Primarykey
            Step 7: For each Primarykey selected (pk,i=0,…,n) do Step 8
                Step 8: Check all fields dependency on pk(i);
                Step 9: While Recordset not EOF and Canseparate =True do Step 9
                    Step 10: IF field type = “Datatype” then do Step 10
                        Step 11: value of field (index) equal to keyInfo
                        Step 12: value of field (Recordset) equal to otherInfo
                                 (Datatype)
                        Else do Step 13
                        Step 13: IF otherInfoInt Not equal to value of field
                                 (Recordset)  Then CanSeparate = False
                    While end
                Step 14: Create new relation with name contains pk(i)
                         (“Table”&NextTableIndex&”_”& MainTableName)
                Step 15: Fill the values OpenRecords ((“Table”&i&”_”&
                          MainTableName)
                Step 16: Remove redundancies of the created table
                Step 17: Create new relation with all pk(i) and otherInfo fields with
                         no primary key dependencies
                         (“Table”&i&”_”& MainTableName)
                Step 18: Display 2NF successful
                Step 19: Set NF2 = True
```

Algorithm 4: Pseudo code of 2NF algorithm.

3.2.2.5 Third Normal Form Algorithm (3NF)

In order to normalize the selected table to 3NF, the algorithm apply the first rule of 3NF by checking first whether the inputted table is in 2NF or not through the flag, if the flag is True then the 2NF table will be opened and show all of its attributes, the user in this case is required to choose the attribute that work as a Transitive key for the table, then the algorithm will apply the second 3NF rule which is about removing the transitive dependencies by separating the table according to the similarities of the fields’ values and creating a new relation based on the

Transitive key of the table along with its filled data. The following algorithm describes the process:

```
Step 0: Check the NF2, IF NF2= false then do Step 1
        Step 1: Display message not in 2NF
      Else do Step 2
Step 2: Open Recordset (2NF table)
Step 3: For each field in table definitions. Fileds, do Step 4-12;
      Step 4: Assign fields to list1;
      Step 5: Add fieldname (Transitivekey) to list2
      Step 6: Get index of the selected TransitiveKey
      Step 7: Create a new table TD.Name = TheTableName & NextTableIndex
      Step 8: Fill up the values Move newRecordset (TheTableName & i)
      Step 9: Check redundant data, For each field in new table Recordset, do Step10-12
            Step 10: IF newRecordset (Fields).Type = (Datatype) Then do Step 11
                     Step11: Move value of Recorset.fields to NewRecordsetpoition
                     Else do Step 12
            Step 12:  Close newRecordset
Step 13: Display TheTableName (i)
Step 14: Set NF3 = True
```

Algorithm 5: Pseudo code of 3NF algorithm.

3.2.3 Graphical User Interface (GUI) Design

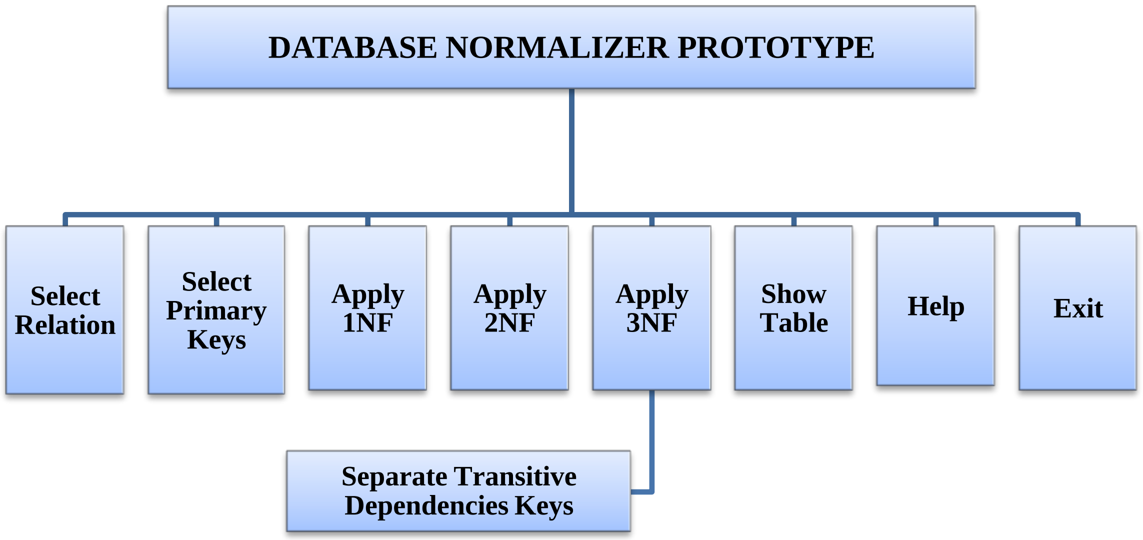


Figure 3.4: Database Normalizer Prototype GUI.

Figure 3.5 illustrates the architecture of the database normalizer prototype.

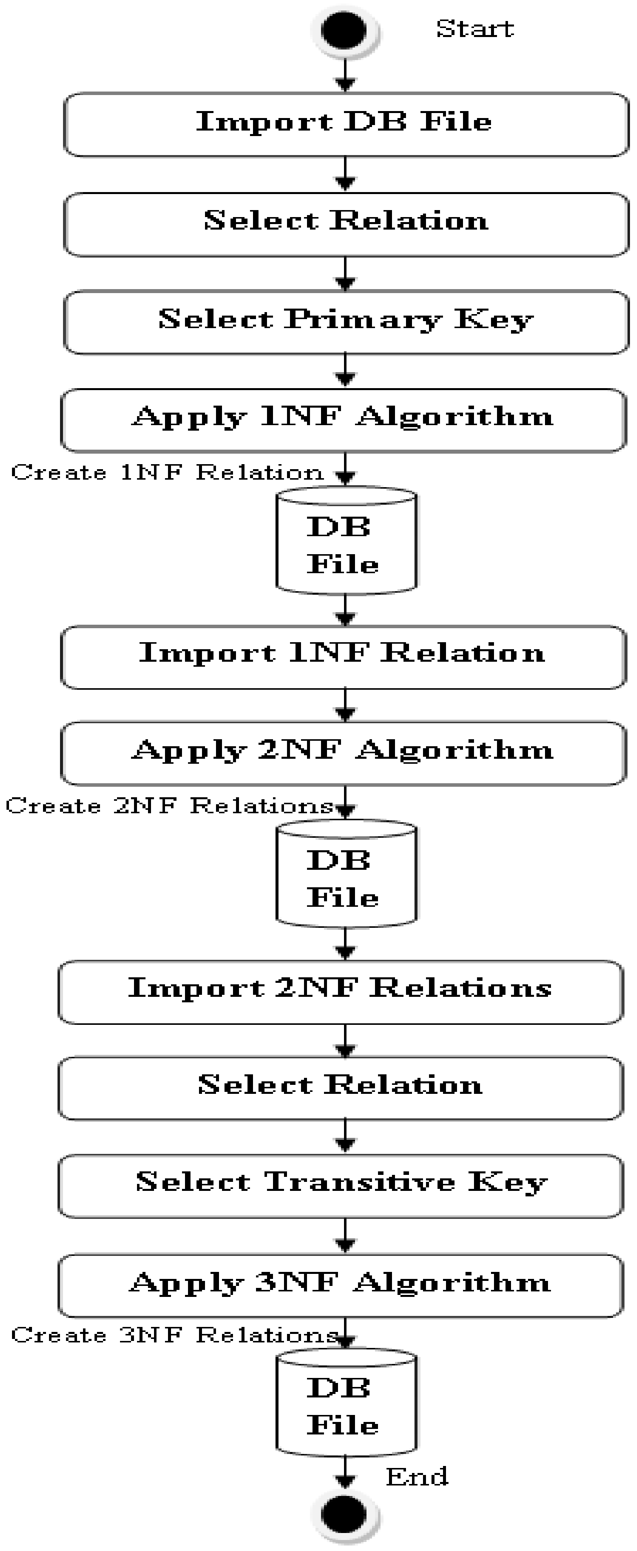


Figure 3.5: Database normalizer prototype architecture.

3.3 Prototype Development

In this phase, the prototype design of the previous phase has been translated into the program code to build the prototype. The prototype developed on windows platform as windows application using Visual Basic programming as front-end and Microsoft Access and SQL Server as back-end.

In this phase, a prototype development methodology called Code-and-Fix has been used. This methodology is appropriate for this prototype which is of type throwaway prototypes that used for answering certain type of questions (Erdil et al., 2003; Norshuhada & Shahizan, 2010). The Code-and-fix methodology chosen because of its iterative nature where it is easy to modify the functionality of the prototype whenever is necessary. This methodology is achieved in two stages: programming some code and then fix the problem in the code which is efficient for early defect detection and correction (McConnell, 1999).The developing process started and as coding error occurs it fixed until the prototype is complete. The developed code is available in Appendix C.

3.4 Experiment

Testing is one of the most critical steps in implementation (Bahrami, 1999). In this step, the prototype undergone to experiment to check the functionality of the prototype in performing the normalization process. Five relations have been checked from chosen databases with different number of attributes and number of rows to be validated and normalized. The experiment considered the result of the manual normalization process and compared it with the prototype normalization result of the 1NF, 2NF and 3NF of each relation. Details of all the used databases relations of this study are available in Chapter Five.

3.5 Evaluation

The evaluation phase is one of the important phases of this study. In this phase, a usability testing technique has been utilized and used to evaluate the prototype by testing it by users. The usability testing evaluation gives direct input and feedback on how real users use the prototype (Nielsen & Levy, 1994).

Questionnaire method has been used to check the prototype easiness of use and satisfaction. The questionnaire designed according to the Technology Acceptance Model (TAM) by Davis (1986) which is an information systems theory that models how users come to accept and use a technology. TAM model suggests that when users are presented with a new technology, a number of factors influence their decision about how and when they will use it, the factors are: Perceived usefulness which defines the degree to which a person believes that using a particular system would enhance his job performance, the other factor is the Perceived ease-of-use which defines the degree to which a person believes that using a particular system would be free from effort (Davis, 1986).

The questionnaire was answered on a five-point Likert scale. The Likert Scale categorized from strongly disagrees to strongly agree to facilitate the data analysis (Best & Kahn, 2006). According to Chan and Teo (2007), the TAM model proved to be a robust in predicting the user acceptance in the IT field and applied usually in understanding issues in computer and software adoption. Easiness of use and satisfaction are important measures for acceptance model (Babar, Winkler, & Biffl, 2007).

Participates in this evaluation were of total 30 Participants (10 developers and 20 students) from database designers in the IT Company SerindIT and IT background students in UUM. The questionnaire questions are available in Appendix B.

3.6 Summary

This chapter presented the methodology that been used to develop the database normalize prototype. The methodology involves five phases: Problem understanding, prototype design, development, experiment, and prototype evaluation. The next chapter presented the design process of the prototype using the UML notation.

CHAPTER FOUR

ANALYSIS OF THE SYSTEM AND DESIGN

4.1 Introduction

This chapter discussed more details on analysis design and proposed prototype design. Among others, this chapter explains on the requirements determination and structuring activity as well as the production of system's design according to functional requirements.

This chapter is elaborating more on analysis. The analysis process should not be taken easily as most observers agreed that many errors occurs in an information system was a consequence of inadequate efforts in analysis and design phase. For that reason, any requirements for the proposed application was thoroughly defined as to make sure that the system meets the needs of the database designer. The eventual aim of this phase is to identify what designer would require from the **Database Normalier Prototype (DBNP)**. In order to come out with the result, the steps for analysing the requirements had been started since at the early stage of the development. The project initiation and planning phase had boosted up the decision of pursuing the study thus ignited the analysis process.

4.2 Tools for System Design

There are several tools available. For this project to designed audio visual system, Unified Modeling Language (UML) Rational Rose 2010 has been selected.

4.2.1 Unified Modeling Language (UML)

According to Krishna and Samuel (2010) the Unified Modeling Language is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process (Berenbach et al., 2009). According to Barclay and Savage (2004) an object-oriented approach using the UML was employed in the development of the online food order management. The primary benefits of such an approach are that it leads to software that demonstrates the following qualities; Reliability, Robustness, Reparability, Resolvability, Maintainability, and Reusability one of the goals of Object-Oriented System Analysis and Design (OOSAD) is to describe several major alternative methodologies for developing Information System (Kern & Garrett, 2003). An object-oriented approach using the UML was employed in the development.

4.2.2 Rational Rose 2010

Rational Rose 2010 is a designed to provide the software developer with a complete set of visual modeling tools and automates parts of the software development process use of Unified modeling Language (UML). It provides a very robust notation, which grows from analysis into design. The output from Rational Rose 2010 case tools are use case diagram, sequence diagram, collaboration diagram, class diagram, object diagram, and activity diagram (Dennis, 2005).

4.3 Database Normalizer Prototype Requirement

System design is the activity of proceeding from an identified set of requirements for a system to a design that meets those requirements (Daintith, 2009), therefore the first step should be exploring the requirements of the system, where the system requirements are the start key and foundation upon which systems are constructed.

4.3.1 Functional Requirements of DBNP

Functional requirements are intended to capture the anticipated behavior of the system. There are several functional requirements to the proposed system (Bennett, 2002). The system consists of the one user who is a database designer; the designer will interact with the system through interfaces. As well as the requirements appear it based on the designer needs. Table 4.1 summarizes the functional requirements for the prototype and gives a brief description of the different requirements.

Table 4.1: Functional Requirements.

Requirement_ ID	**Function Requirement**	**Priority**
DBNP -01	**Import Database**	**Mandatory**
	By the DBNP the designer can import the required database file to apply the normalization process.	
DBNP -02	**Select primary key**	**Mandatory**
	The designer shall define the data base tables' primary key trough the use of this function	
DBNP -03	**Apply 1st Normal Form**	**Mandatory**
	This function allows to designer to apply the **1st Normal Form**	
DBNP -04	**Apply 2nd Normal Form**	**Mandatory**
	This function allows to designer to apply the **2nd Normal Form**	
DBNP – 05	**Apply 3rd Normal Form**	**Mandatory**
	This function allows to designer to apply the **3rd Normal Form**	

4.3.2 Non-Functional Requirements of DBNP

The non-functional requirements try to capture properties of the system that has to do with performance, quality or features that are not fundamental for the system to work (Kern & Garrett, 2003). They are however very important because they are often properties that highly desired by the user and can help the system gain competitive advantage over other systems (Krishna & Samuel, 2010). The list of the nonfunctional requirements for the system as follow:

- Secure data handling: The data has to be stored in a way that they cannot be compromised.
- User friendly: The graphical user interface has to be easy to understand.
- Reliability: Availability of the system, rate of failure occurrence very low.
- Speed: The system will increase the speed of all daily activities.
- Navigation: The system offering the opportunity to go to other parts of the application.
- Help & Support: Support workflow in the system and support the user to fulfill their missions.
- Error handling: Errors are avoided as much as possible.

4.4 Modeling and System Design

This section illustrates the design of the system. The design of this system includes the use of UML diagrams. According to Barclay and Savage (2004) UML diagram is designed to let developers and users view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modeling tools include. It includes use case diagram, sequence diagrams and class diagram.

4.4.1 Use Case Diagram

A use case diagram is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases (Egeberg, 2006). The two main components of a use case diagram are use cases and actors. The following Figure 4.1 shows the DBNP use case diagram.

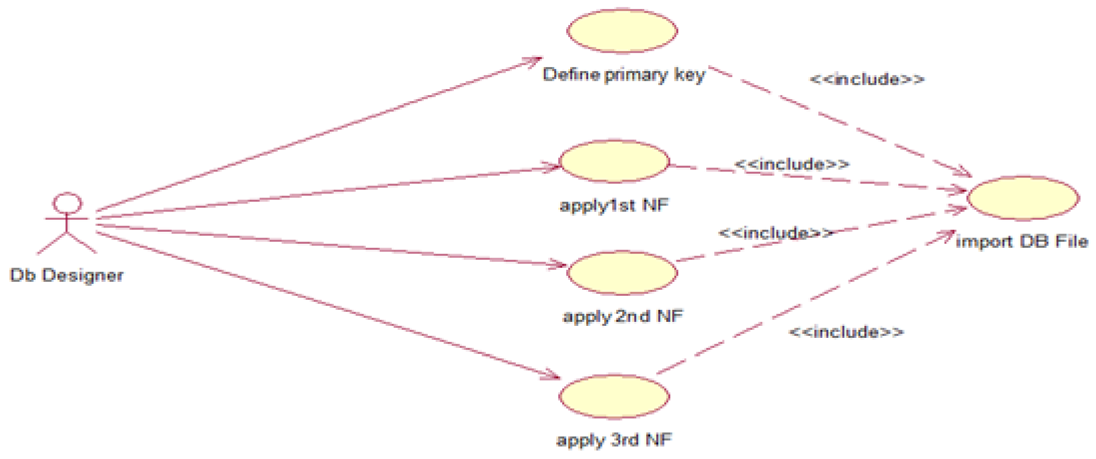


Figure 4.1: DBNP use case Diagram.

4.4.2 Use Case Specification

The use case specification shows the essential specifications of the prototype based on the uses case diagram (Jivan & Gruner, 2009). The DBNP use cases specifications for 1NF, 2NF and 3NF are depicted in Table 4.2, Table 4.3 and Table 4.4 respectively.

Table 4.2: Apply First Normal Form use case specification.

Use Case Name: **Apply 1st Normal Form**	ID: **DBNP_03**	Importance Level: **High**
Primary Actor: **Designer**		
Short Description: Through this function the designer can apply the 1st Normal Form on the selected database file.		
Type: **External** / Temporal.		
Basic Flow of Events **Designer** **1.** Designer imports the database file. **2.** Select the primary key of the selected database. **3.** Press on apply 1st normal form button.	**System** **4.** The system applies the 1st Normal Form algorithm. **5.** The system displays the confirmation message.	
Alternatives Not applicable		
Exceptions : E1: “ Primary key not selected ”		
Characteristic of Activation Click apply button.		
Pre-conditions Database File.		
Post-conditions 1st Normal form done.		

Table 4.3: Apply Second Normal Form use case specification.

Use Case Name: **Apply 2nd Normal Form**	ID: **DBNP_04**	Importance Level: **High**
Primary Actor: **Designer**		
Short Description: Through this function the designer can apply the 2nd Normal Form on the selected database file which is on the first normal form.		
Type: **External** / Temporal		
Basic Flow of Events **Designer**	**System**	
1. Designer imports the database file on 1st NF.	**3.** The system applies the 2nd Normal Form algorithm.	
2. Press on apply 2nd normal form button.	**4.** The system displays the confirmation message.	
Alternatives Not applicable		
Exceptions: E1: “Database file on 1 NF not selected”.		
Characteristic of Activation Click apply button		
Pre-conditions Database File on 1st NF		
Post-conditions 2nd Normal form done.		

Table 4.4: Apply Third Normal Form Use Case Specification.

Use Case Name: **Apply 3nd Normal Form**	ID: **DBNP_05**	Importance Level: **High**
Primary Actor: **Designer**		
Short Description: Through this function the designer can apply the 3rd Normal Form on the selected database file which is on the second normal form.		
Type: **External** / Temporal		
Basic Flow of Events		
Designer	**System**	
1. Designer imports the database file on 2nd NF (designer required to choose the required table to apply rules of 3NF)	**3.** The system applies the 3rd NF algorithm.	
2. Press on apply 3rd Normal form button.	**4.** The system displays confirmation message.	
Alternatives Not applicable		
Exceptions: E2: “database file on 2 NF not selected”.		
Characteristic of Activation Click apply button.		
Pre-conditions Database File on 2nd NF.		
Post-conditions 3nd Normal form done.		

4.4.3. DBNP Sequence Diagrams

The sequence diagram kind of Interaction diagram that used to describe the object interaction (Johan, 2004). The sequence diagram shows the interactions among objects that participate in a use case and the message that pass between them over time for one use case. In this prototype, there are three main sequence diagrams that represent 1NF, 2NF, and 3NF as shows in Figure 4.2, 4.3 and 4.4 respectively.

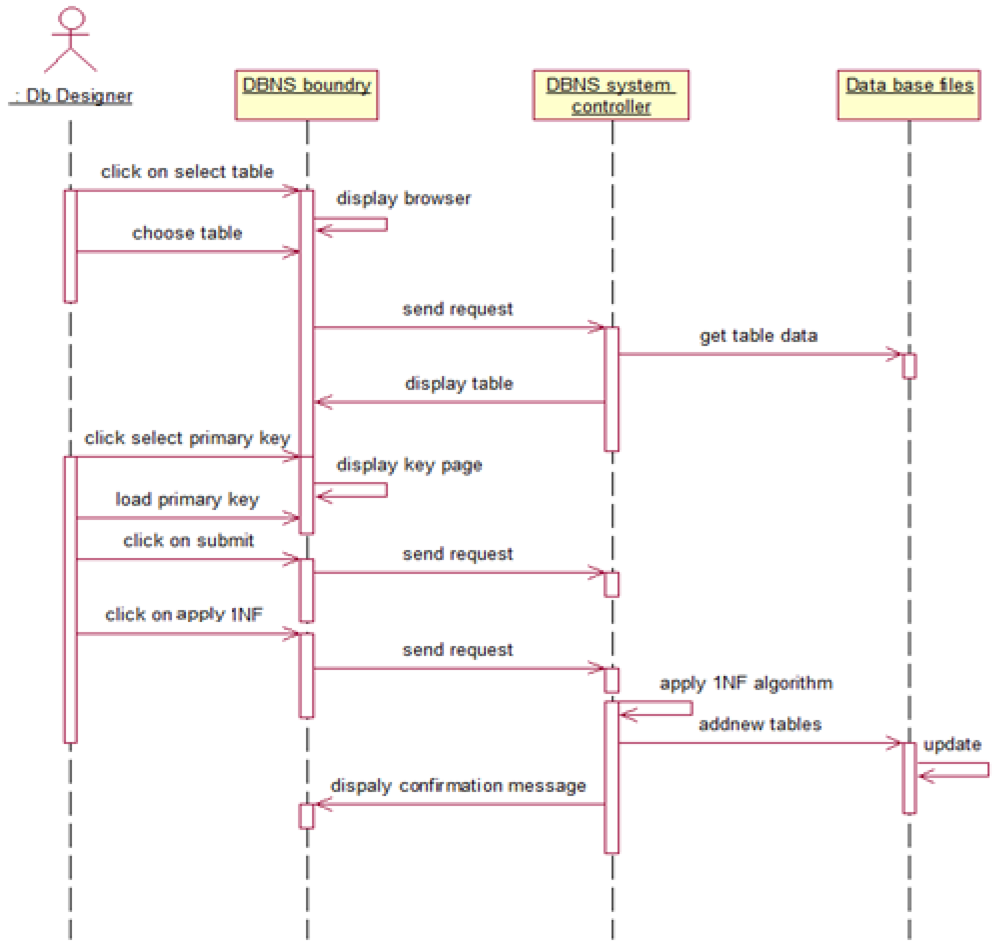


Figure 4.2: Apply 1NF Sequence Diagram.

Figure 4.3 illustrates the process of applying the second normal form (2NF) sequence diagram. This function allows the database designer to apply the 2NF process.

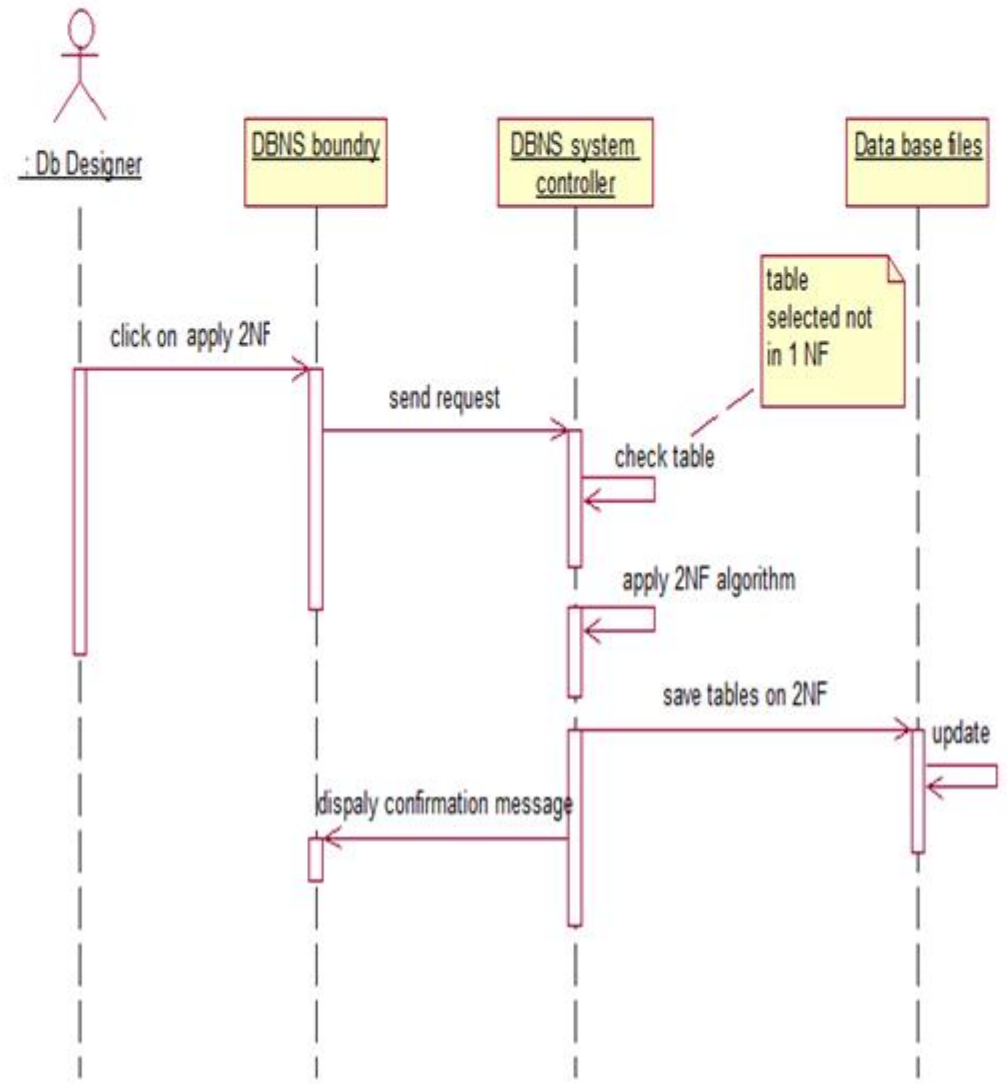


Figure 4.3: Apply 2NF Sequence Diagram.

Figure 4.4 illustrates the process of applying the third normal form (3NF) sequence diagram. This function allows the database designer to apply the 3NF process.

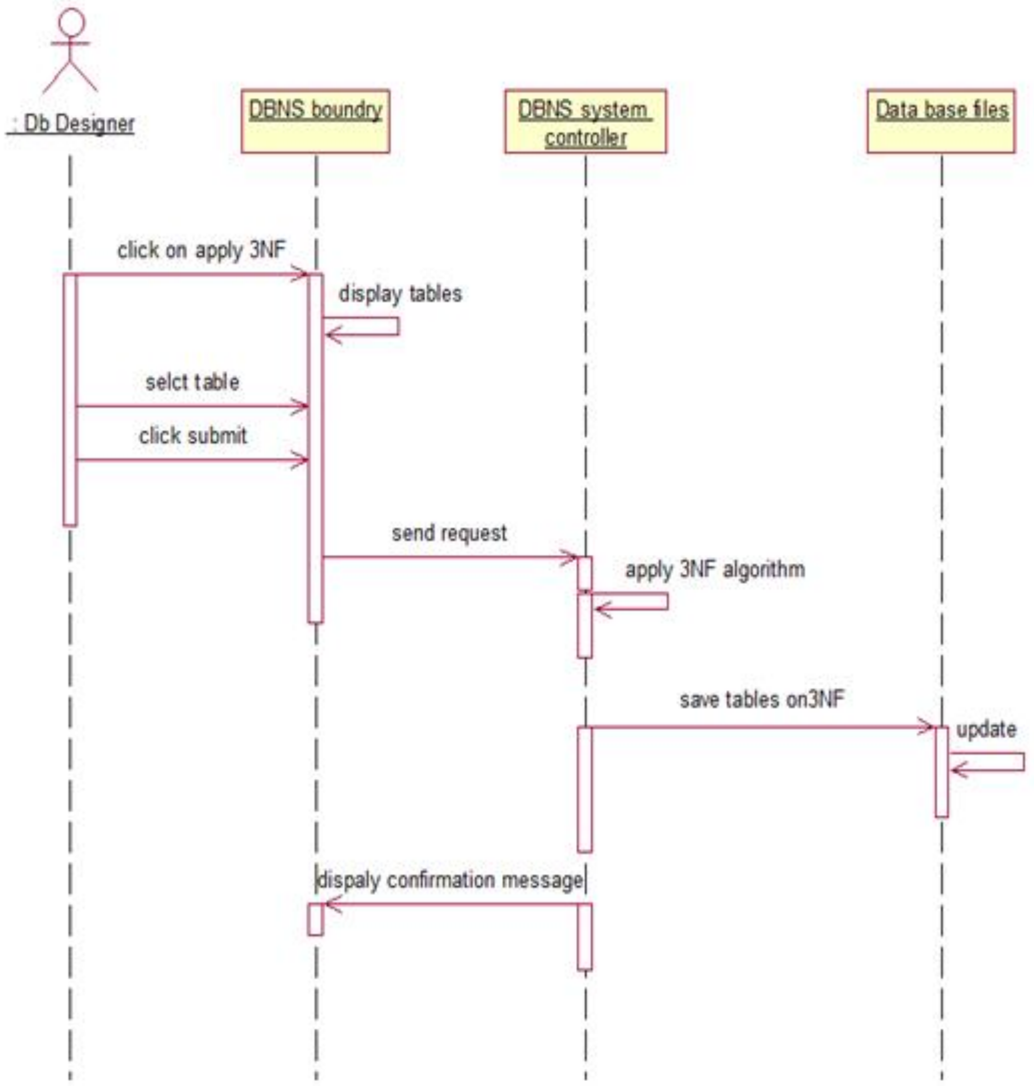


Figure 4.4: Apply 3NF Sequence Diagram.

4.4.4 Class diagram

Class diagram one of UML diagrams and type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes (Yi, Wu & Gan, 2005). Class diagrams for system are shown in Figure 4.5 bellow.

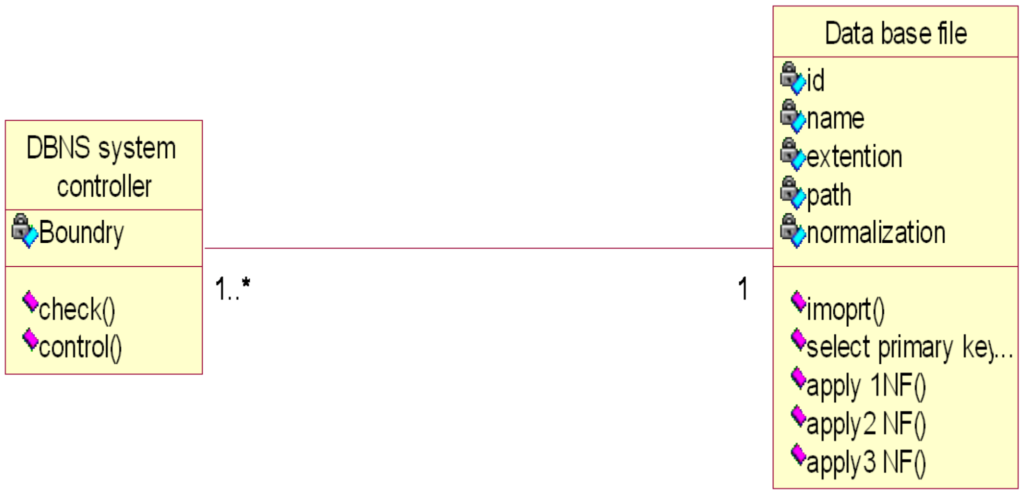


Figure 4.5: DBNP class diagram.

4.5 Prototype Implementation

This phase involves the development of the prototype database normalizer. Prior to implementing the prototype, the prototype requirements have been reviewed to be sure that the prototype behavior is correct (Peters & Parnas, 2002). The design was translated into program code using VB language. Microsoft Access and SQL Server used as the databases to store and retrieve all the required tables that needs to be normalized. The aesthetics of the appearance of the user interface was granted a prime attention to make the user experience as pleasant as possible. The following section provides snapshots of the prototype user interfaces.

4.6 Graphical User Interface

The developed prototype has many interfaces based on the requirements that have been discussed in Chapter Three. The following are the prototype interfaces that show the normalization process functionalities and operations:

(A) Home Page

From the home page, the database designer starts performing the normalization process where this page contains the following functions as illustrated in Figure 4.6:

- Select Relation: this used for purpose of selecting the required relation that's need to be normalized.
- Select Primary Key: this is used for purpose of identifying the primary keys of the selected table.
- Apply First Normal Form: this is for purpose of applying the 1NF rules on the selected table where the INF algorithm will be applied automatically.
- Apply Second Normal Form: this is for purpose of applying the 2NF rules on the selected table where the 2NF algorithm will be applied automatically.
- Apply Third Normal Form: this is for purpose of applying the 3NF rules on the selected table where the 3NF algorithm will be applied.
- Help: This will show the prototype manual for the user.
- Exit: To exit from the prototype.
- Show table: this is used for purpose of showing the normalization generated tables for 1NF, 2NF, and 3NF.

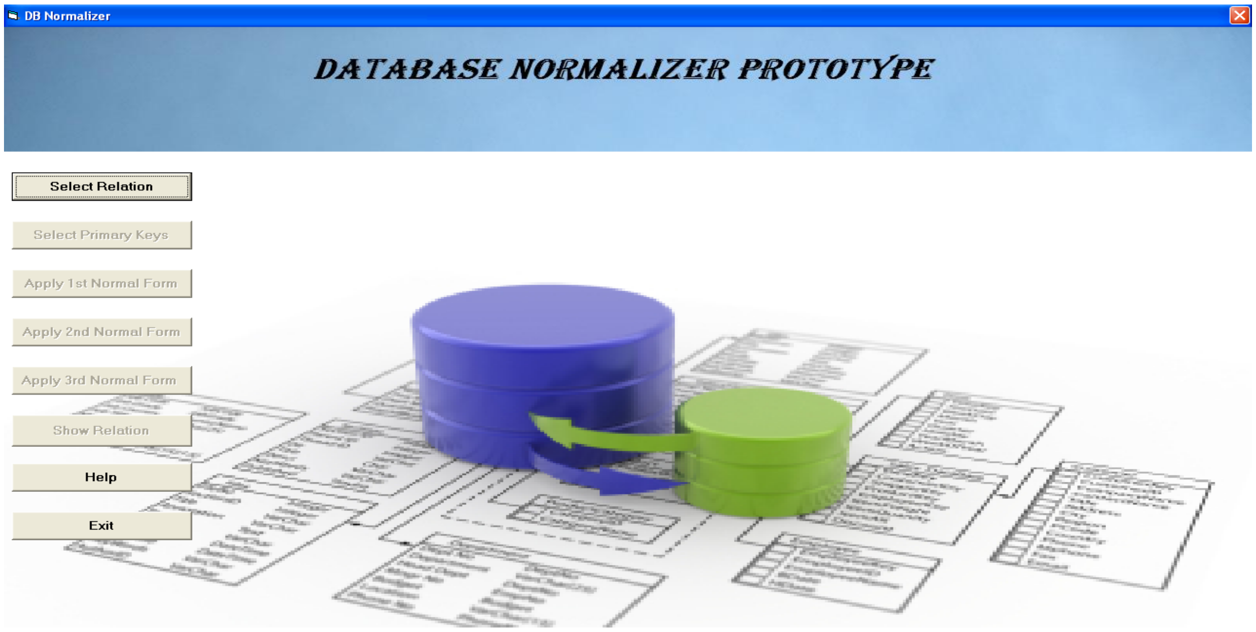
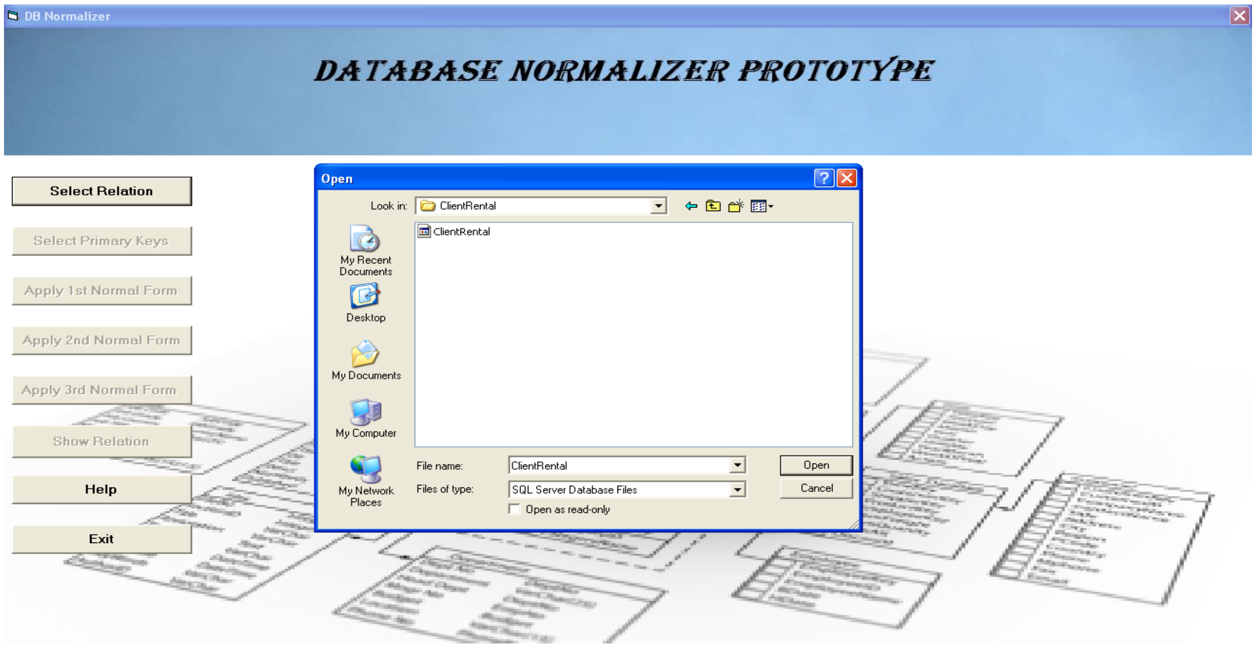


Figure 4.6: Database Normalizer Home Page.

(B) Select Table

Figure 4.7 illustrates the screen of selecting a database where the user is required to select a database to be normalized through the open wizard.



Look in: ClientRental

ClientRental

File name: ClientRental

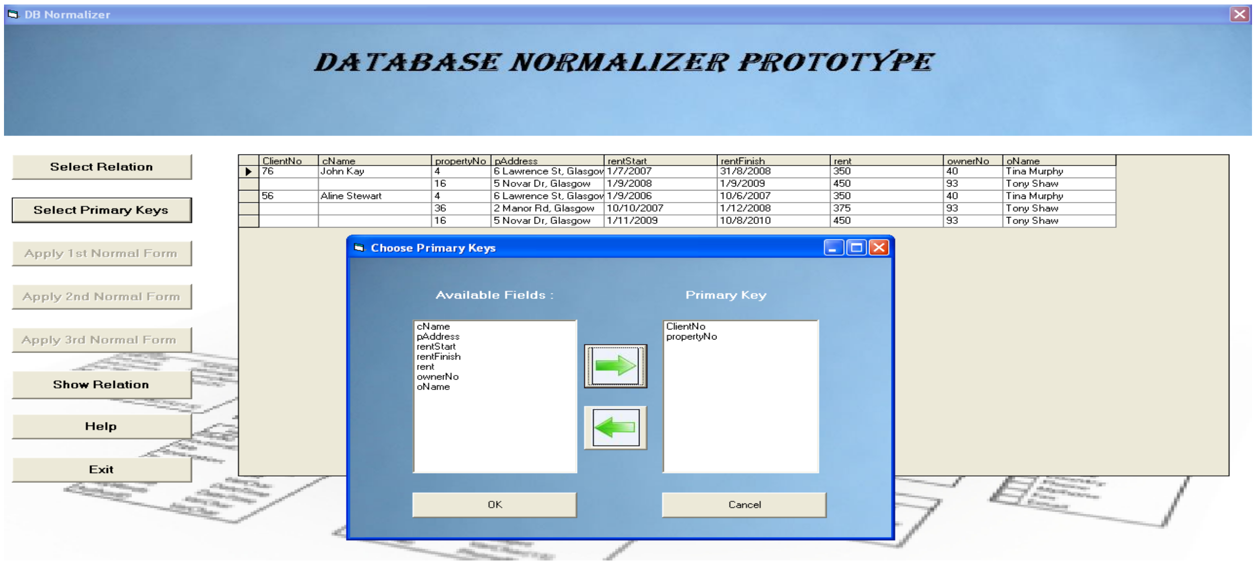
Files of type: SQL Server Database Files

Open as read-only

Figure 4.7: Selecting database to be normalized.

(C) Select Primary Key

Figure 4.8 illustrates the screen of selecting the selected table primary keys from a list that contains all tables' attributes.



ClientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
76	John Kay	4	6 Lawrence St, Glasgow	1/7/2007	31/8/2008	350	40	Tina Murphy
		16	5 Novar Dr, Glasgow	1/9/2008	1/9/2009	450	93	Tony Shaw
56	Aline Stewart	4	6 Lawrence St, Glasgow	1/9/2006	10/6/2007	350	40	Tina Murphy
		36	2 Manor Rd, Glasgow	10/10/2007	1/12/2008	375	93	Tony Shaw
		16	5 Novar Dr, Glasgow	1/11/2009	10/8/2010	450	93	Tony Shaw

Available Fields :

cName
pAddress
rentStart
rentFinish
rent
ownerNo
oName

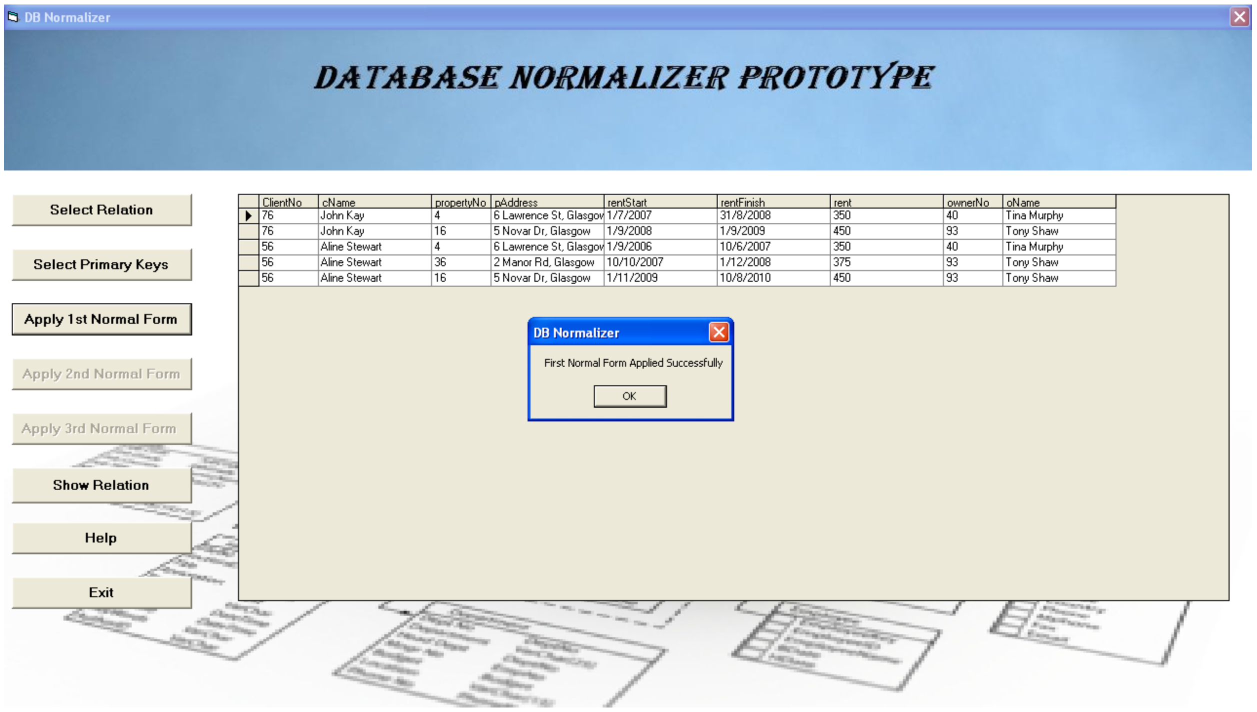
Primary Key

ClientNo
propertyNo

Figure 4.8: Selecting table primary keys to be normalized.

(D) 1NF Form

Figure 4.9 illustrates the screen of performing the 1NF for the selected table by pressing 1NF button and the system show the process result in the grid.



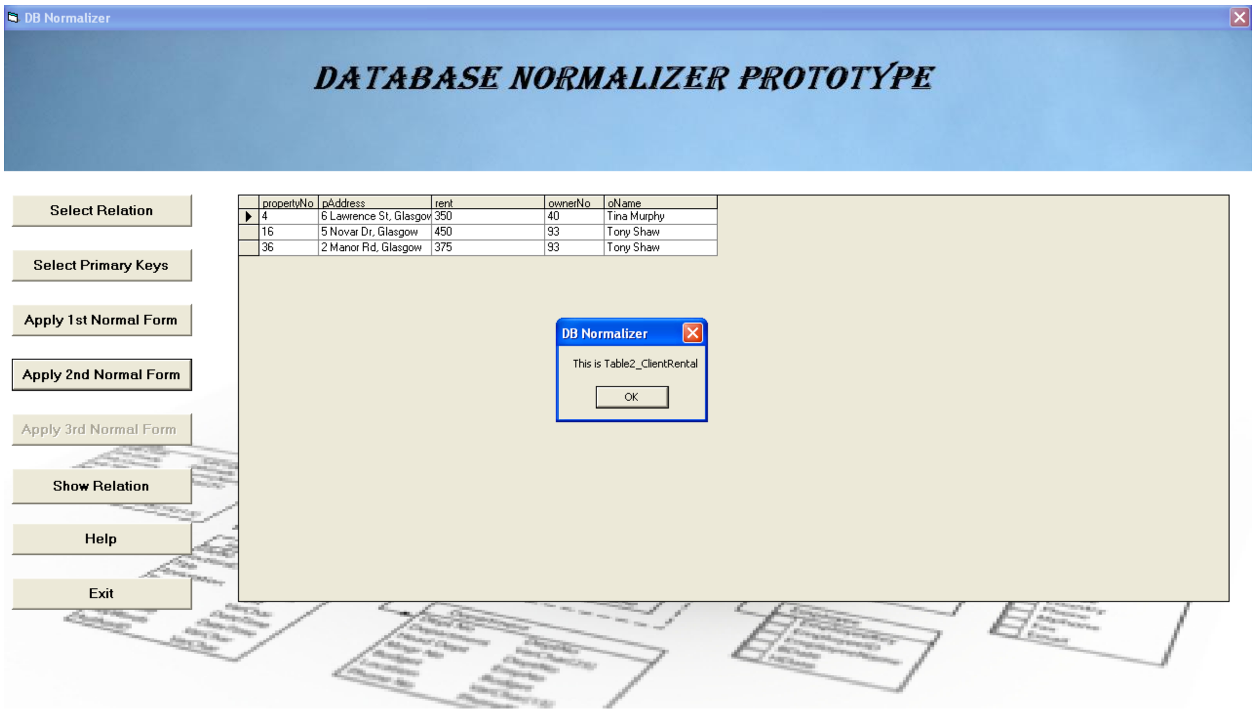
ClientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
76	John Kay	4	6 Lawrence St, Glasgow	1/7/2007	31/8/2008	350	40	Tina Murphy
76	John Kay	16	5 Novar Dr, Glasgow	1/9/2008	1/9/2009	450	93	Tony Shaw
56	Aline Stewart	4	6 Lawrence St, Glasgow	1/9/2006	10/6/2007	350	40	Tina Murphy
56	Aline Stewart	36	2 Manor Rd, Glasgow	10/10/2007	1/12/2008	375	93	Tony Shaw
56	Aline Stewart	16	5 Novar Dr, Glasgow	1/11/2009	10/8/2010	450	93	Tony Shaw

First Normal Form Applied Successfully

Figure 4.9: Performing the 1NF operation.

(E) **2NF Form**

Figure 4.10 illustrates the screen of performing the 2NF for the selected table by pressing 2NF button and the system perform the normalization automatically and shows the process result in the grid.



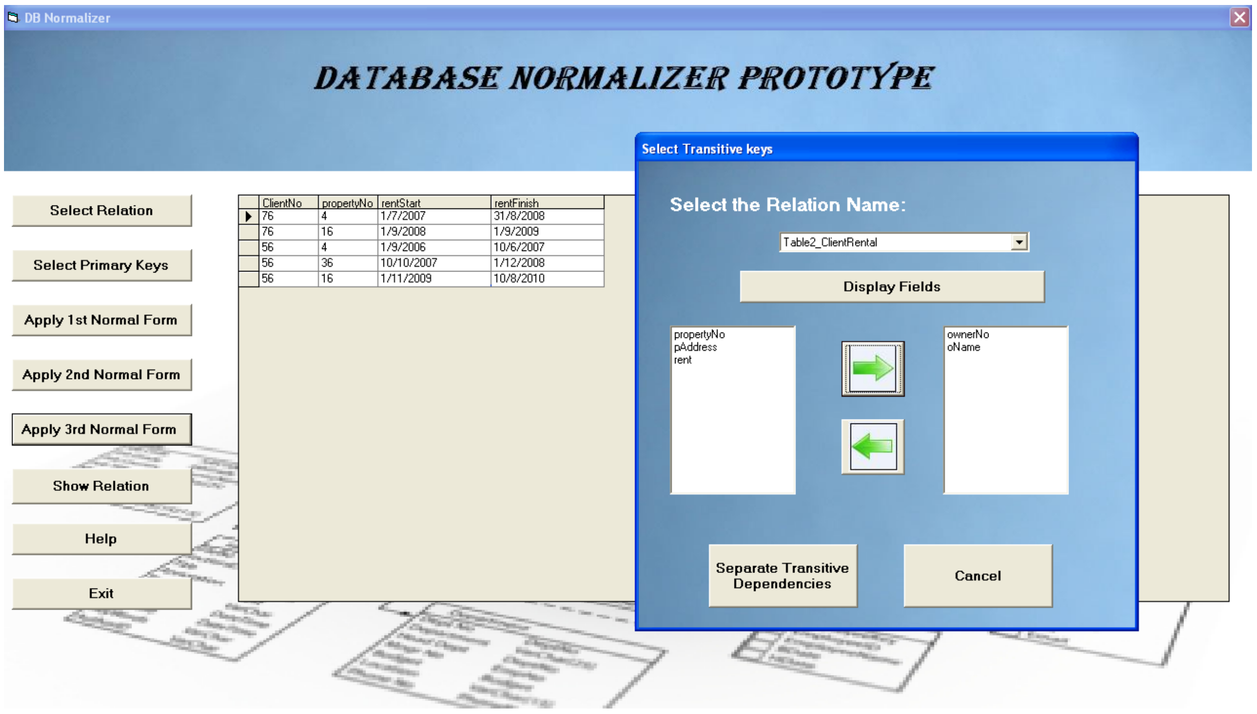
propertyNo	pAddress	rent	ownerNo	oName
4	6 Lawrence St, Glasgow	350	40	Tina Murphy
16	5 Novar Dr, Glasgow	450	93	Tony Shaw
36	2 Manor Rd, Glasgow	375	93	Tony Shaw

This is Table2_ClientRental

Figure 4.10: Performing the 2NF operation.

(F) **3NF Form**

Figure 4.11 illustrates the screen of performing the 3NF process for a table in a 2NF in which the user is required to specify the transitive keys of the table and the system perform the normalization automatically and shows the process result in the grid.



ClientNo	propertyNo	rentStart	rentFinish
76	4	1/7/2007	31/8/2008
76	16	1/9/2008	1/9/2009
56	4	1/9/2006	10/6/2007
56	36	10/10/2007	1/12/2008
56	16	1/11/2009	10/8/2010

Select the Relation Name:

Table2_ClientRental

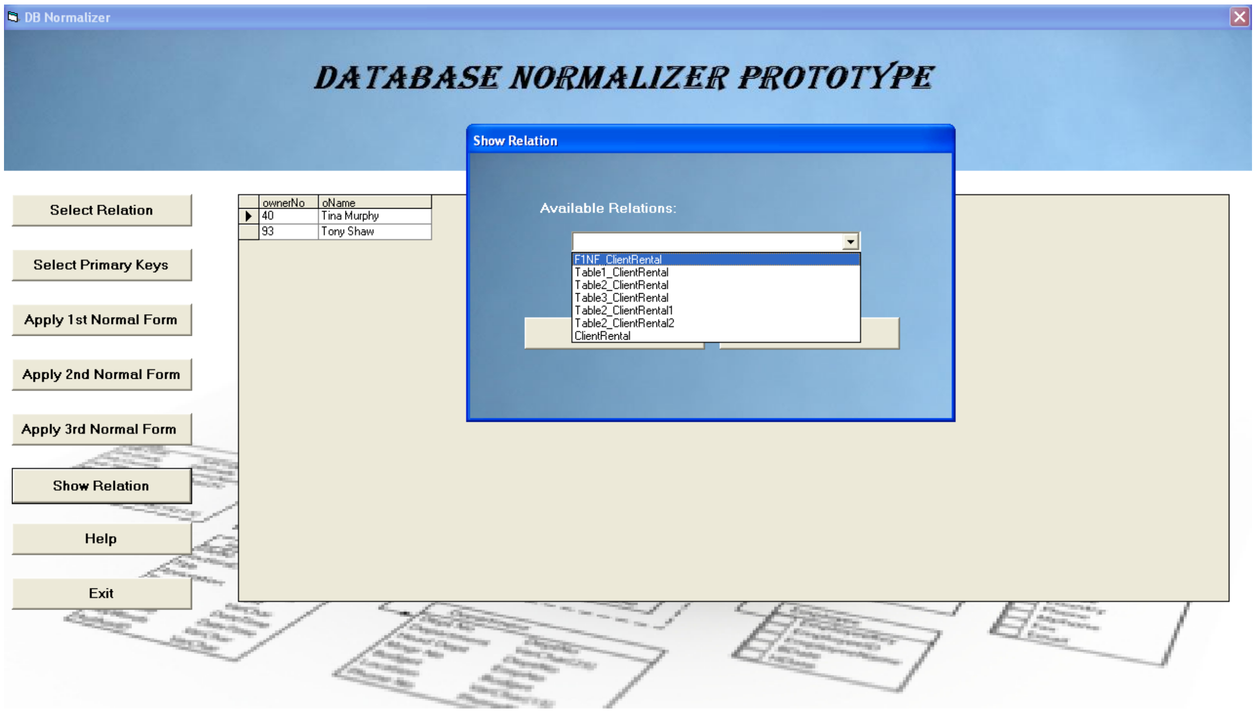
propertyNo
pAddress
rent

ownerNo
oName

Figure 4.11: Performing the 3NF operation and selecting the table transitive keys.

(G) **Show Table Form**

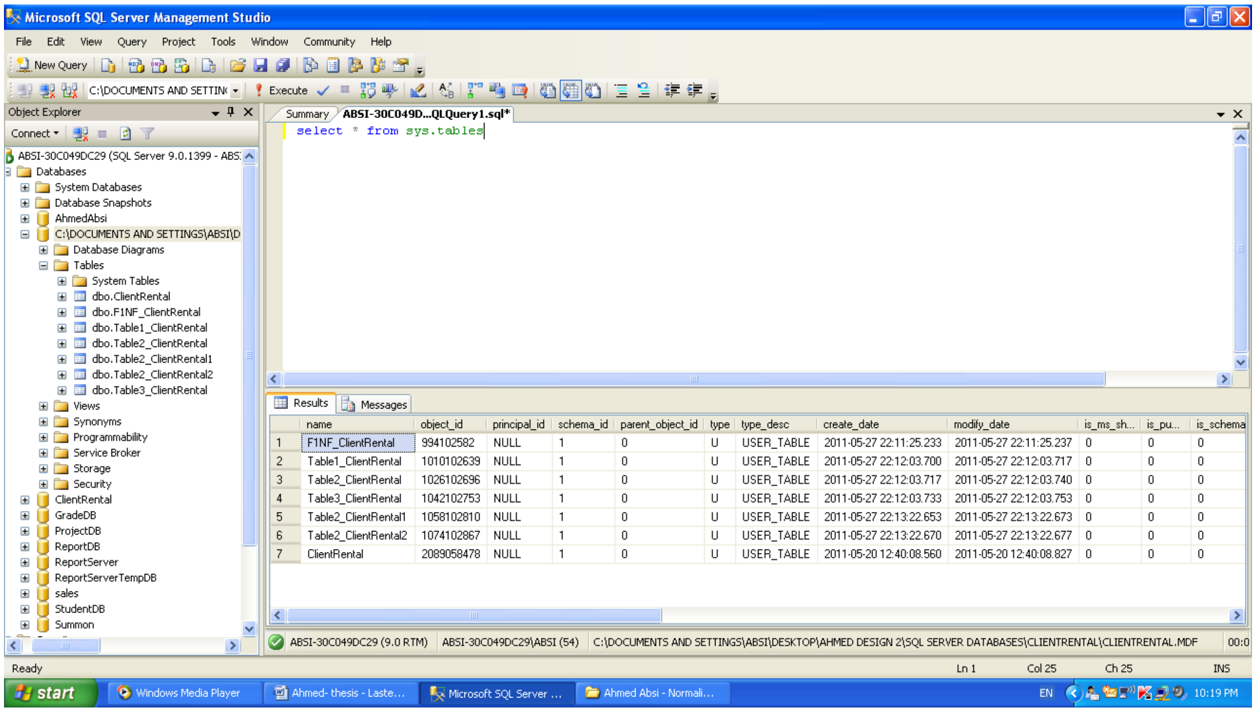
Figure 4.12 illustrates the output tables of each normal form process “1NF, 2NF, and 3NF”. The system creates these tables in the SQL Server database automatically at the same time of performing the normalization.



ownerNo	oName
40	Tina Murphy
93	Tony Shaw

F1NF_ClientRental
Table1_ClientRental
Table2_ClientRental
Table3_ClientRental
Table2_ClientRental1
Table2_ClientRental2
ClientRental

Figure 4.12: Shows the tables that been created at 1NF, 2NF and 3NF levels.



```
select * from sys.tables
```

ABSI-30C049DC29 (SQL Server 9.0.1399 - ABSI
- Databases
  - System Databases
  - Database Snapshots
  - AhmedAbsi
  - C:\DOCUMENTS AND SETTINGS\ABSI\D
    - Database Diagrams
    - Tables
      - System Tables
      - dbo.ClientRental
      - dbo.F1NF_ClientRental
      - dbo.Table1_ClientRental
      - dbo.Table2_ClientRental
      - dbo.Table2_ClientRental1
      - dbo.Table2_ClientRental2
      - dbo.Table3_ClientRental
    - Views
    - Synonyms
    - Programmability
    - Service Broker
    - Storage
    - Security
  - ClientRental
  - GradeDB
  - ProjectDB
  - ReportDB
  - ReportServer
  - ReportServerTempDB
  - sales
  - StudentDB
  - Summon

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_sh...	is_pu...	is_schema
1	F1NF_ClientRental	994102582	NULL	1	0	U	USER_TABLE	2011-05-27 22:11:25.233	2011-05-27 22:11:25.237	0	0	0
2	Table1_ClientRental	1010102639	NULL	1	0	U	USER_TABLE	2011-05-27 22:12:03.700	2011-05-27 22:12:03.717	0	0	0
3	Table2_ClientRental	1026102696	NULL	1	0	U	USER_TABLE	2011-05-27 22:12:03.717	2011-05-27 22:12:03.740	0	0	0
4	Table3_ClientRental	1042102753	NULL	1	0	U	USER_TABLE	2011-05-27 22:12:03.733	2011-05-27 22:12:03.753	0	0	0
5	Table2_ClientRental1	1058102810	NULL	1	0	U	USER_TABLE	2011-05-27 22:13:22.653	2011-05-27 22:13:22.673	0	0	0
6	Table2_ClientRental2	1074102867	NULL	1	0	U	USER_TABLE	2011-05-27 22:13:22.670	2011-05-27 22:13:22.677	0	0	0
7	ClientRental	2089058478	NULL	1	0	U	USER_TABLE	2011-05-20 12:40:08.560	2011-05-20 12:40:08.827	0	0	0

ABSI-30C049DC29 (9.0 RTM) | ABSI-30C049DC29\ABSI (54) | C:\DOCUMENTS AND SETTINGS\ABSI\DESKTOP\AHMED DESIGN 2\SQL SERVER DATABASES\CLIENTRENTAL\CLIENTRENTAL.MDF | 00:0

Figure 4.13: Shows the automatically created normalized tables in SQL Server database.

4.7 Summary

This chapter discussed the design, development and implementation of the system at the prototype level. The functional and non functional requirements of the system were initially determined and then the system modeling was carried out the Unified Modeling Language (UML). The sequence diagrams and the class diagrams were also presented in this chapter as design stage. The system was implemented using VB as front-end and Microsoft Access and SQL Server as back-end. The screen shots of the user interfaces for the different normalization processes functionalities and operations have been presented.

CHAPTER FIVE

RESULTS AND FINDINGS

5.1 Introduction

This chapter aims to discuss the evaluation process of the database normalize prototype. The evaluation phase is one of the important phases of this study. In this phase, a usability testing technique has been utilized and used to evaluate the prototype. According to Holzinger (2005) usability testing with end users is one of the most essential methods in usability evaluation. The usability testing evaluation gives direct input and feedback on how real users use the prototype (Nielsen & Levy, 1994).The prototype has undergone to an experiment for purpose of checking the functionality of the prototype in performing the normalization process. Moreover, a questionnaire method has been used to ensure of the prototype level in terms of easiness of use and satisfaction.

5.2 Experiment Design

As the development phase completed the prototype undergone to experiment to check the functionality in performing the normalization process. For this purpose, five examples of relations up to the 3NF have been collected from standard books and research papers.

The chosen relations are from different databases with different number of attributes, rows and functional dependencies to be validated and normalized. The experiment considered the result of the manual normalization process that mentioned in standard books and research papers i.e. Connolly & Begg (2010), Teorey et al. (2011), Srikanth & Sudarshan (2001), and compared it with the prototype normalization result of the 1NF, 2NF and 3NF of each relation. Details and description of the 5 benchmark databases relations in this study are presented in Table 5.1 where the functional dependencies of each relation are separated by semicolon. To test the normalized relations output of the prototype, Table 5.2 shows the expected output of the prototype.

Table 5.1 Relations used in the experiment.

R No.	**Relation Name**	**Relation Description**	**Number of Attributes**	**Number of Functional Dependencies**
1	**Report Relation** (Teorey et al., 2011)	(ReportNo, Editor, DeptNo, DeptName, DeptAddress, AuthorID,AuthorName, AuthorAddress) **Functional Dependencies are:** - ReportNo → Editor, DeptNo; - DeptNo → DeptName, DeptAddress; - AuthorID → AuthorName; - AuthorID → AuthorAddress;	8	6
2	**ClientRental Relation** (Connolly & Begg, 2010)	(clientNo, propoertyNo, cName, pAddress, rentStart, rentFinish, rent, ownerNo, oName) **Functional Dependencies are:** - clientNo, propoertyNo →	9	17

		rentStart, rentFinish; - clientNo → cName; - ownerNo → oName; - prpoertyNo → pAddress, rent, ownerNo, oName; - clientNo, rentStart → prpoertyNo, pAddress, rentFinish, rent, ownerNo, oName; - propoertyNo, rentStart → clientNo, cName, rentFinish;		
3	**Student Relation** (Srikanth & Sudarshan, 2001)	(MatricNo,SurName,DateOfBirth, MentorID,MentorSurName, MentorOffice,CourseCode, CourseName,Credits,Grade) **Functional Dependencies are:** - MatricNo → SurName, DateOfBirth,MentorID, MentorSurName, MentorOffice; - MentorID → MentorSurName	10	12

		,MentorOffice; - MatricNo → SurName , DateOfBirth; - CourseCode → Credits, CourseName; - MatricNo,CourseCode → Grade;		
4	**Project Relation** (Rob & Coronel, 2009)	(ProjectNumber,ProjectName, EmployeeNumber,EmployeeName, JobClass,ChargeHour,HoursBilled) **<u>Functional Dependencies are:</u>** - ProjectNumber → ProjectName; - EmployeeNumber → EmployeeName,JobClass, ChargeHour; - JobClass → ChargeHour; - ProjectNumber, EmployeeNumber → HoursBilled;	7	6

5	**Grade Relation** (Shelly, Cashman, & Rosenblatt, 2009)	(StudentNo,StudentName, TotalCredits,GPA, AdvisorNo , AdvisorName , CourseNo, CourseDESC , NumCredits , Grade) **<u>Functional Dependencies are:</u>** - StudentNo → StudentName, TotalCredits,GPA,AdvisorNo ,AdvisorName; - CourseNo → CourseDESC, NumCredits; - AdvisorNo → AdvisorName; - StudentNo → StudentName, TotalCredits,GPA;	10	12

Table 5.2 Experimentation expected results.

R No.	Relation Name	2NF Result	3NF Result

1	Report Relation	ReportNo(ReportNo,Editor, DeptNo,DeptName, DeptAddress)
		
		AuthorID (AuthorID,AuthorName, AuthorAddress)
		
		(ReportNo,AuthorID)
			ReportNo(ReportNo,Editor, DeptNo)
			
			DeptNo(DeptName, DeptAddress)
			
			AuthorID(AuthorID, AuthorName,AuthorAddress)
			
			(ReportNo,AuthorID)

2	ClientRental Relation	Client (clientNo, cName) Rental (clientNo, propertyNo, rentStart, rentFinish) PropertyOwner (propertyNo, pAddress,rent, ownerNo, oName)	Client (clientNo, cName) Rental (clientNo, propertyNo,rentStart, rentFinish) PropertyForRent (propertyNo, pAddress, rent, ownerNo) Owner (ownerNo, oName)

3	Student Relation	Student (MatricNo,SurName, DateOfBirth,MentorID, MentorSurName,MentorOffice) Course (CourseCode, CourseName,Credits) Grade (MatricNo, CourseCode,Grade)	Student (MatricNo,SurName, DateOfBirth) Mentor (MentorID, MentorSurName, MentorOffice) Course (CourseCode, CourseName,Credits) Grade (MatricNo, CourseCode,Grade)
4	Project Relation	Project (ProjectNumber, ProjectName) Employee (EmployeeNumber, EmployeeName,JobClass, ChargeHour) ProjectEmployee (ProjectNumber, EmployeeNumber, HoursBilled)	Project (ProjectNumber, ProjectName) Employee (EmployeeNumber, EmployeeName,JobClass) (JobClass, ChargeHour) ProjEmp (ProjectNumber, EmployeeNumber, HoursBilled)

5	Grade Relation	(StudentNo,StudentName, TotalCredits,GPA,AdvisorNo, AdvisorName) (CourseNo,CourseDESC, NumCredits) (StudentNo,CourseNo,Grade)	(StudentNo,StudentName, TotalCredits,GPA) (AdvisorNo,AdvisorName) (CourseNo,CourseDESC, NumCredits) (StudentNo,CourseNo,Grade)

5.2.1 Experiment Result

In this experiment, the functionality of the prototype has been checked through comparison between output result of the prototype and expected result in Table 5.2. The experiment showed that the prototype achieved the result successfully as expected and fulfills the requirements and rules of 1NF, 2NF and 3NF of normalization processes that been discussed in chapter two.

5.3 Questionnaire

Questionnaire method has been used to check the prototype easiness of use and satisfaction. The questionnaire designed according to the Technology Acceptance Model (TAM) by Davis (1989) and was answered on a five-point Likert scale. Participates in this evaluation were of total 30 Participants (10 developers and 20 students) from database designers.

The questionnaire measured using the Likert Scale format ranging from strongly disagree to strongly agree and included three main sections, firstly the demographic questions, secondly prototype easiness of use questions and lastly satisfaction questions. The questionnaire consisted of 11 questions exclude the participant's demographical information. The questionnaire questions are available in Appendix B. All statistical analysis was carried out using SPSS program, version 17 (SPSS Inc, Chicago, Il, USA).

5.3.1 Questionnaire Analysis

The prototype has been set along with the questionnaire for the participants to evaluate it in two places at SerindIT company and at UUM-CAS. The questionnaires have been collected after participants' answered the questions where the participants' response against each question and each group has been analyzed and calculated. Graphs and tables were used to represent the statistical data obtained from questionnaires.

According to the analysis, majority of the participants were of male gender with percentage of 83.3% while female formed 16.7% in this study.

The Figure 5.1 below shows the sample of the study (type of the participants) which were for a total of 30 participants. The analyzing showed the following types of evaluators: Students formed 66.67% of the participation, 13.33% were DB developers and designers, 16.67% were programmers and 3.33% participated as analyst who were working at SerindIT company.

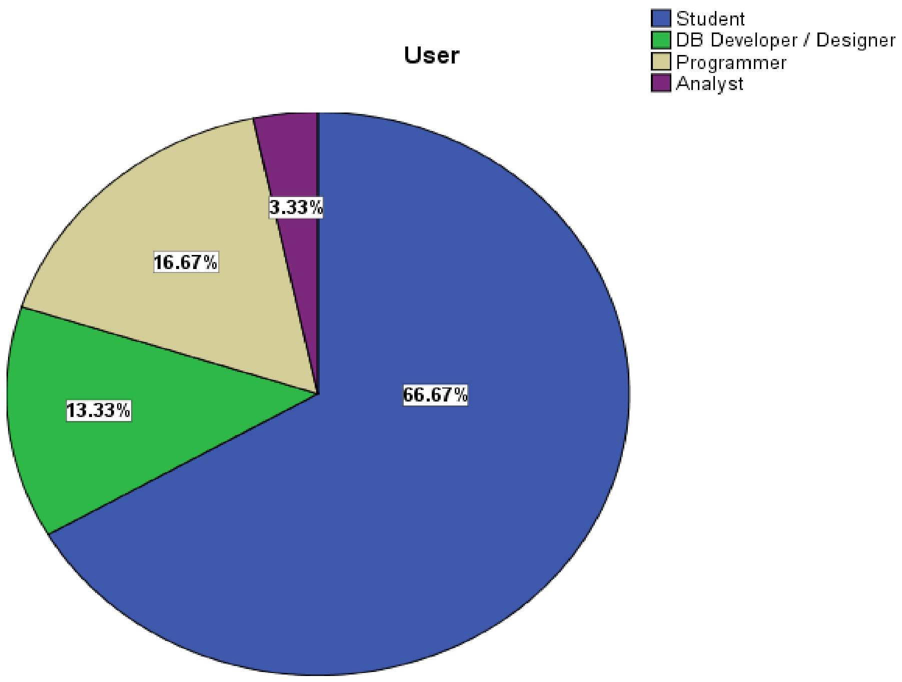


Figure 5.1: Type of participants.

Age of the participants were 46.67% for the age interval 18-29, 50% for the interval 30-39 and 3.33% were in the interval 40-50 as shown in Figure 5.2.

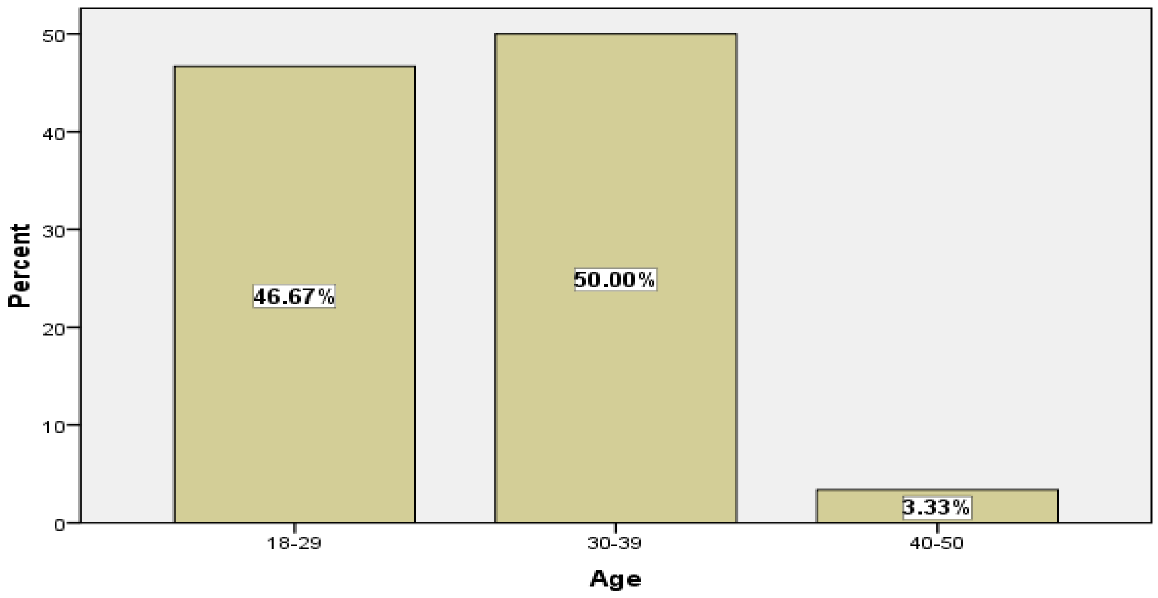


Figure 5.2: Age of the participants.

The educational background of the participants in this study varies as following: 80% as Master students, 13.33% as Bachelor and 6.67% as PhD as shown in Figure 5.3.

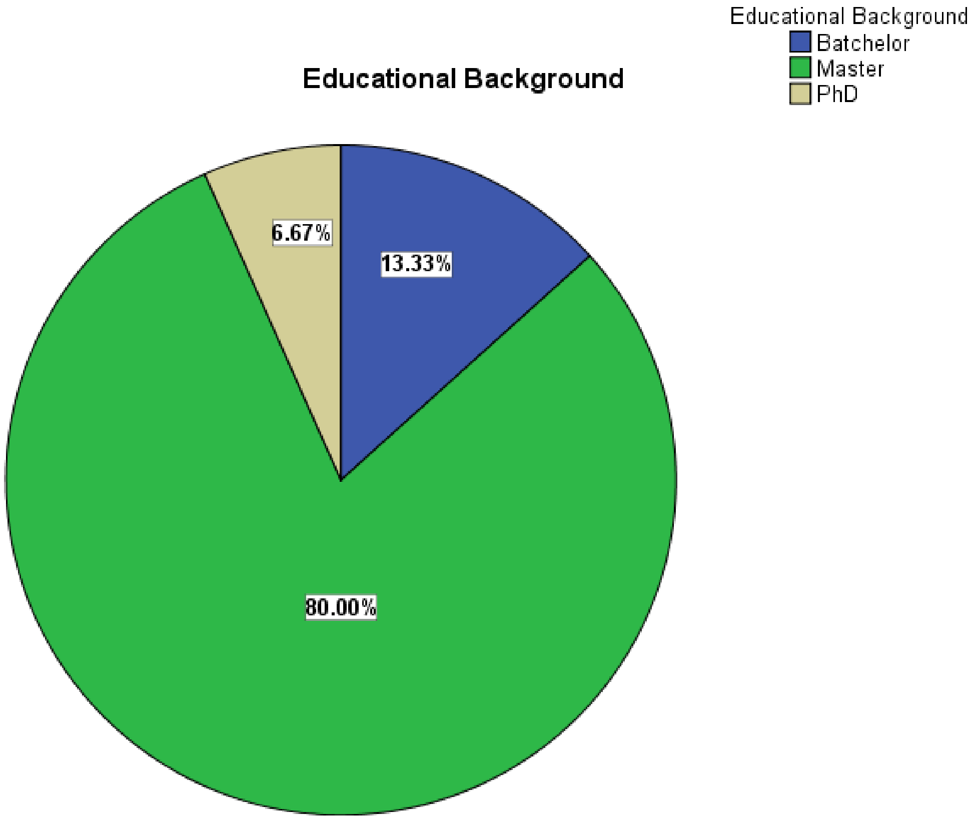


Figure 5.3: Participants educational background.

5.3.2 Easiness Evaluation

The first section of the questionnaire questions was to evaluate the prototype easiness of use. This section consists of six questions; Table 5.3 describes the number of the respondent, the minimum and maximum answer, the mean and the STD deviation for this section.

Table 5.3: Descriptive Statistics (Easiness of use)

Descriptive Statistics (Easiness Of Use)

	N	Minimum	Maximum	Mean	Std. Deviation
Q1	30	1	5	3.83	1.020
Q2	30	2	5	3.60	.814
Q3	30	2	5	4.07	.828
Q4	30	1	5	4.10	.960
Q5	30	3	5	4.20	.714
Q6	30	2	5	4.17	.834
Valid N (listwise)	30				

Since the six questions measure the prototype easiness of use, the summation of the corresponding values of the (mean) row of each question has been divided by the total number of questions i.e. six questions, so the mean of the all mean values corresponding to the easiness of use questions is 3.995 which equal almost 66.58% indicates that measuring the prototype easiness of use is high and agreed altitude.

5.3.3 Satisfaction Evaluation

The second section of the questionnaire questions was to evaluate the prototype satisfaction. This section consisted of five questions; Table 5.4 describes the number of the respondent, the minimum and maximum answer, the mean and the STD deviation for this section.

Table 5.4: Descriptive Statistics (Satisfaction)

Descriptive Statistics (Satisfaction)

	N	Minimum	Maximum	Mean	Std. Deviation
Q7	30	2	5	4.10	.803
Q8	30	2	5	4.23	.898
Q9	30	3	5	4.30	.750
Q10	30	3	5	4.17	.531
Q11	30	4	5	4.60	.498
Valid N (listwise)	30				

Since the five questions measure the prototype satisfaction, the summation of the corresponding values of the (mean) row of each question has been divided by the total number of questions i.e. five questions, so the mean of the all mean values corresponding to the easiness of use questions is 4. 28 which equal almost 86.6% indicate that measuring the prototype satisfaction is high and agreed altitude.

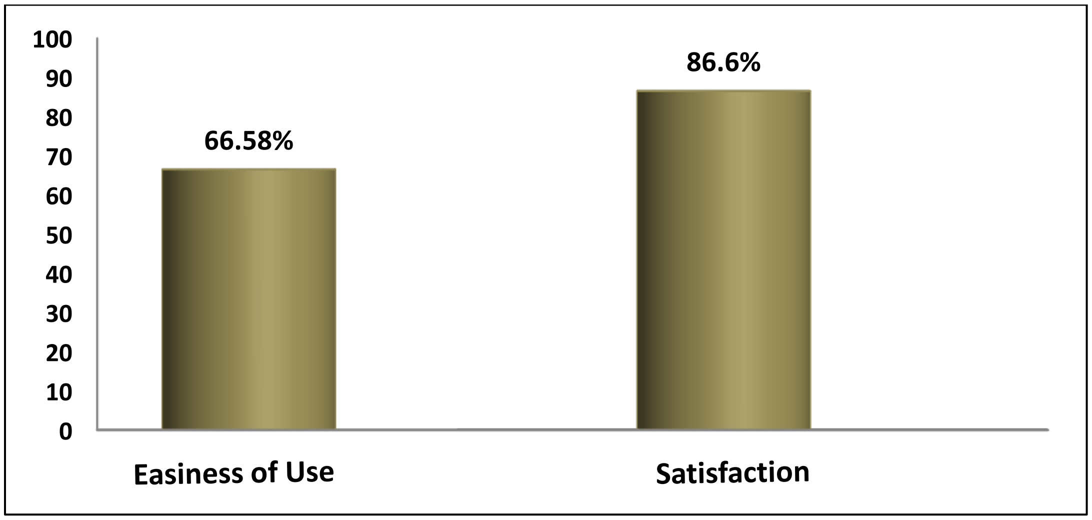


Figure 5.4: Evaluation result.

5.3.4 Reliability Statistics

According to Field (2006) reliability is the degree to which measure are free from error and therefore yield consistent results. Cronbach's alpha based on standardized items used to measure the reliability scale of the prototype usability. The closer the reliability coefficient gets to 1.0, the better it is, and those values over .80 are considered as good and those value in the .70 is considered as acceptable and those reliability value less than .60 is considered to be poor (Yu, 2000).

5.3.4.1 Reliability Statistics for Easiness Evaluation

As shown in Table 5.5, Alpha value for the easiness evaluation is above 0.8 which is considered as good.

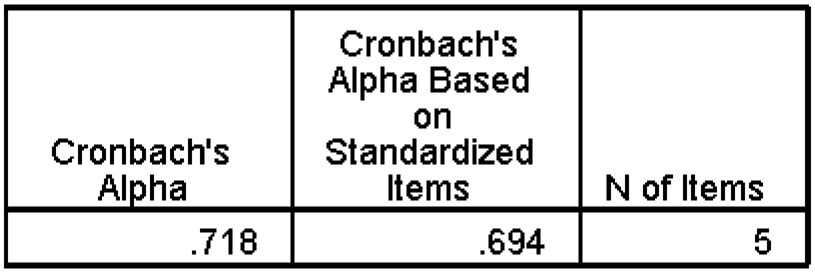
Table 5.5: Reliability Statistics (Easiness Evaluation)

Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.814	.818	6

5.3.4.2 Reliability Statistics for Satisfaction Evaluation

Table 5.6 illustrates Cronbach's alpha based on standardized items for satisfaction evaluation. The alpha value for the satisfaction evaluation is above 0.7 which is considered as acceptable.

Table 5.6: Reliability Statistics (Satisfaction Evaluation)

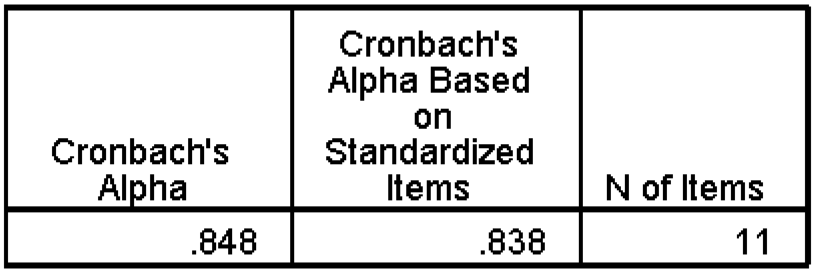


Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.718	.694	5

5.3.4.3 Reliability Statistics for All Sections

Table 5.7 illustrates Cronbach's alpha to all questions (Easiness and Satisfaction). The alpha value for the database normalizer prototype usability evaluation is above 0.8 which is considered as good.

Table 5.7: Reliability Statistics

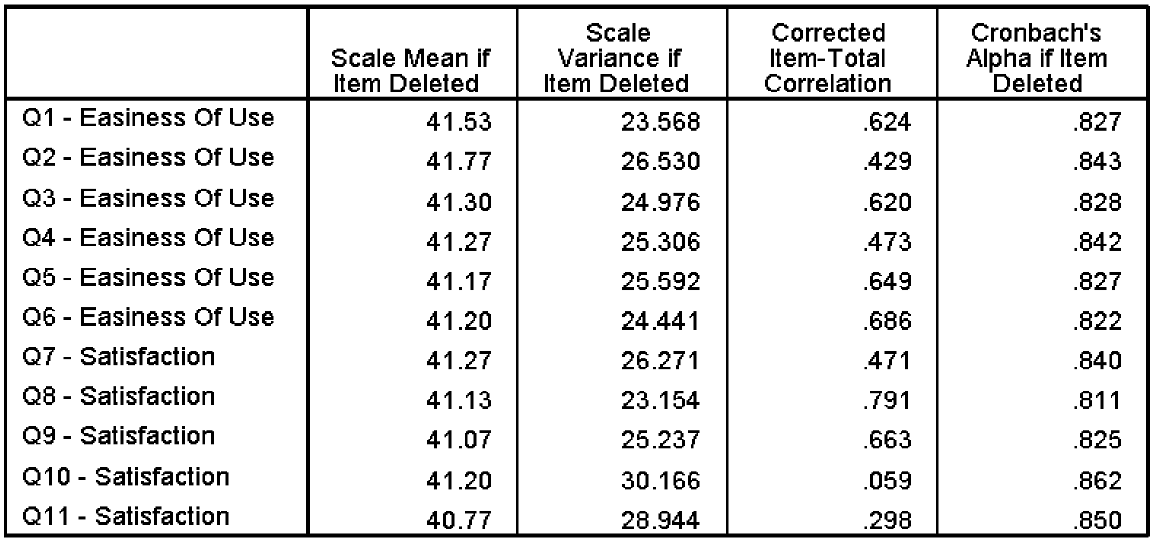


Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.848	.838	11

5.3.4.4 Item-Total Statistics

According to Field (2006) one of the most important for questionnaire reliability the scale if item deleted. The questionnaire is reliable if the value of Cronbach's alpha around 0.8 or higher. Table 5.8 shows the percentage of scale mean if item deleted and the scale variance if item deleted and the corrected item total correlation and the Cronbach's alpha if item deleted for the prototype usability questions.

Table 5.8: Item-Total Statistics



	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
Q1 - Easiness Of Use	41.53	23.568	.624	.827
Q2 - Easiness Of Use	41.77	26.530	.429	.843
Q3 - Easiness Of Use	41.30	24.976	.620	.828
Q4 - Easiness Of Use	41.27	25.306	.473	.842
Q5 - Easiness Of Use	41.17	25.592	.649	.827
Q6 - Easiness Of Use	41.20	24.441	.686	.822
Q7 - Satisfaction	41.27	26.271	.471	.840
Q8 - Satisfaction	41.13	23.154	.791	.811
Q9 - Satisfaction	41.07	25.237	.663	.825
Q10 - Satisfaction	41.20	30.166	.059	.862
Q11 - Satisfaction	40.77	28.944	.298	.850

5.4 Summary

This chapter presented the evaluation process of the database normalize prototype in terms of three factors: functionality, easiness of use, and satisfaction. The evaluation process has been conducted by two usability testing, the first one was by an experiment for purpose of checking the functionality of the prototype in performing the normalization process. The second usability test was by a questionnaire to evaluate the prototype easiness of use and satisfaction.

The experiment showed that the prototype achieved the result successfully as expected and fulfills the requirements and rules of 1NF, 2NF and 3NF. Moreover, the questionnaire reveals that 66.58% of the participants indicated that the prototype easiness of use and 86.6% indicated that the prototype satisfaction which is high and agreed altitude.

The analysis of the evaluation process revealed that the prototype fulfils the requirements needed to normalize databases relations systematically. This led to conclude that the usability of the prototype was very high on aspects of functionality, easiness and satisfaction.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

This chapter concludes and summaries the findings of this study and present the research contribution along with the limitations and future work.

6.1 Conclusion

This study addressed the problem of performing the database normalization process manually by the database designers as mentioned in Chapter One. The main objective of this study is to develop a prototype tool that can do database normalization systematically. The prototype has been developed to process the first three forms of normalization (1NF, 2NF, &3NF). The prototype undergone to experiment to check the functionality of the algorithms in performing the normalization process and the experiment showed that the prototype achieved the result successfully as expected and fulfill the requirements and rules of normalization processes, which were mentioned in Chapter Two. In terms of measuring the prototype easiness of use and satisfaction, the questionnaire reveals that 66.58% of the participants indicated that the prototype easiness of use and 86.6% indicated that the prototype satisfaction which is high and agreed altitude.

6.2 Research Contribution

The developed database normalizer prototype would assist the database designers and developers to achieve the process of database tables' normalization systematically. Additionally, the developed prototype would also significantly contribute to the society especially for users who work on database normalization. The prototype helps in reducing time of database designing and errors of normalization process especially once dealing with large number of attributes. In addition, the database designers can reduce their effort on normalizing the tables by not spending long time in performing the normalization. Moreover, this research contributes to the area of academic in educational environment to assist the students understanding the database normalization process.

This prototype is easily used by any user with little background on database normalization like database relations attributes and functional dependencies. Hence, this prototype will offer extensive benefits to all database designers.

6.3 Problems and Limitations

This study focused on developing a database normalizer prototype to perform the normalization process on databases automatically. The prototype has been developed successfully. However there are some limitations as following:

- Although the prototype can cover a wide area of scope of database management systems types like Oracle, MySQL, and Sybase but the prototype focused on the

Microsoft Access and SQL Server databases only. Due to time constrained, other types of DBMS did not consider in this prototype.

- The prototype can normalize up to the third normalization only.
- Time delay problem has been reported during the creating table process of the first normal form in Microsoft Access databases while no delay in SQL Server databases.

6.4 Future work

The development for normalizing database tables automatically has been achieved in this study. However, a lot of enhancements still can be made on the prototype in the future as following:

- Currently, the prototype can handle tables till the third normal form; the prototype can be extended to 4NF, and 5NF.
- Supporting all types of database management systems.
- Support of visualization of functional dependencies constructed.
- Evaluating the prototype in educational environment to know the positive impact of the prototype to assist the students in understanding the database normalization process.

6.5 Summary

This chapter summaries the whole research processes included the findings of the study, research contribution, problems and limitations faced during the study, and the recommended future works to enhance the study in future.

REFERENCES

Akehurst, D., Bordbar, B., Rodgers, P., & Dalgliesh, N. (2002). Automatic normalization via Metamodelling, In *Proceedings of the ASE Declarative Meta Programming to Support Software Developmen,* held on September 23-24, 2002 at Edinburgh, UK (pp. 23-27). Edinburgh: IEEE.

Babar, M., Winkler D., & Biffl, S. (2007). Evaluating the usefulness and ease of use of a groupware tool for the software architecture evaluation process. In *Proceedings of the first international symposium on empirical software engineering and measurement, ESEM 2007*, held on September 20-21, 2007 at Madrid, Spain (pp. 430-439). CA: IEEE Computer Society.

Bahmani, A., Naghibzadeh, M., & Bahmani, B. (2008). Automatic database normalization and primary key generation. In *proceedings of the 21st Canadian Conference on Electrical and Computer Engineering,* held on May 4-7, 2008 at Ontario, Canada (pp. 11-16). CA: IEEE CCECE.

Bahmani, A., Shekofteh, S., Naghibzadeh, M., & Deldari, H. (2010). Parallel algorithms for automatic database normalization. *Computer and Automation Engineering*, *2*(1), 157-161

Bahrami, A. (1999). *Object-Oriented Systems Development: Using the Unified Modeling Language*. New York: McGraw-Hill.

Barclay, K., & Savage, J. (2004). *Object-Oriented design with UML and Java*. Burlington, USA: Elsevier Butterworth-Heinemann.

Bennett, S., McRobb, S., & Farmer, R. (2002). *Object-oriented system analysis and design* (2nd ed.). UK: McGraw Hill.

Bhavsar, C. (2008). Comparison between Windows Forms and Web Applications. Retrieved March 30, 2011, from http://www.eggheadcafe.com/community/aspnet/2/10036174/whats the major difference between windows and web applications.aspx

Berenbach, B., Paulish, D., Kazmeier, J., & Rudorfer, R. (2009). *Software & systems requirements engineering in practice*. New York: McGraw-Hill.

Best, J., & Kahn, J. (2006). *Research in Education* (10th ed.). New York: Pearson Education Inc.

Chan, H.C., & Teo, H.-H. (2007). Evaluating the boundary conditions of the technology acceptance model: An exploratory investigation. *ACM Transactions on Computer-Human Interaction, 14*(2), 1-22.

Connolly, T., & Begg, C. (2004). *Database solutions: A step-by-step approach to building databases* (2nd ed.). Boston: Pearson.

Connolly, T., & Begg, C. (2010). *Database Systems: A practical approach to design, implementation, and management* (5th ed.). Boston: Pearson.

Daintith, J. (2009). Systems design a dictionary of computing. Retrieved May 15, 2011, from http://www.encyclopedia.com/doc/1O11-systemdesign.html.

Davis, F. (1989). *Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results*. Boston, MA: Massachussetts Institute of Technology.

Dennis, A., Wixom, B., & Tegarden, D. (2005). *System analysis and design with UML version 2.0: an object-oriented approach with UML* (2nd ed.). Hoboken, NJ: John Wiley and Sons, Inc.

Dongare, Y., Dhabe, P., & Deshmukh, S. (2011). RDBNorma: A semi-automated tool for relational database schema normalization. *International Journal of Database Management Systems*, *3*(1), 133-154.

Egeberg, M. (2006). *The mobile phone as a contactless ticket.* Master's thesis, Norwegian University of Science and Technology, Norway.

Erdil, K., Finn, E., Keating, K., Meattle, J., Park, S., & Yoon, D. (2003). *Software maintenance as part of the software life cycle* (Department of Computer ScienceTufts University Technical Report No. Comp-180). Retrieved March 29, 2011, from http://www.hepguru.com/maintenance/Final_1.pdf

Field, A. P. (2006). Discovering statistics using SPSS (2nd ed.). London: Sage.

Jivan, E. & Gruner, S. (2009).Tool support for more precise use-case specifications. In *Proceedings of Warm-Up Workshop for ACM, WUP/ISS 2009*, held on April 1-3, 2009 at Cape Town, South Africa (pp. 29-32). Cape Town: ACM.

Johan, K. (2004). Information system analysis and design. Retrieved May 11, 2011, from http://www.cs.toronto.edu/jm/3405/slides2/sequenceD.pdf.

Hoffer, J., George, J., &Valacich, J. (2002). *Modern Systems Analysis and Design* (3rd ed.). Upper Saddle River, New Jersey: Prentice Hall.

Hoffer, J., Prescott, M., & McFadden, F. (2007). *Modern Database Management* (8th ed.). Upper Saddle River, New Jersey: Prentice Hall.

Holzinger, A. (2005). Usability Engineering Methods for Software Developers. *Communications of the ACM*, *48*(1), 71-74.

Kern, J., & Garrett, C. (2003). Effective Sequence Diagram Generation: Effective Use of Options with Borland Together Technologies. Retrieved May 27, 2011, from http:www.borland.com/resources/en/pdf/white_papers/20263.pdf

Krishnan, H., & Samuel, P. (2010). Relative Extraction Methodology for Class Diagram Generation using Dependency Graph. In *Proceedings of the International Conference on Communication, Control and Computing Technologies, ICCCCT 2010,* held on October 7-9, 2010 at Kanyakumari, Tamilnadu (pp. 815-820). Tamilnadu: IEEE.

Kung, H., & Tung, H. (2006). A web based tool to enhance teaching/Learning database normalization. In *Proceedings of international conference of southern association for information system, SAIS 2006.*30-38.

Martin, F., & Kendall, S. (2000). *UML Distilled: Brief guide to the standard object modeling language* (2nd ed.). Boston, USA: Addison-Wesley Longman Publishing.

McConnell, S. (1999, August). Open-Source Methodology: Ready for Prime Time? [Electronic version]. *IEEE Software*, *16*(4). 6-11.

Mitrovic, A. (2002). NORMIT: a web-enabled tutor for database normalization. In *Proceedings of the International Conference on Computers in Education ICCE, 2002,* held on December 3-6, 2002 at Auckland, New Zealand (pp. 275-80). CA: IEEE Computer Society.

Nielsen, J., & Levy, J. (1994). Measuring usability: Preference vs. performance. *Communications of the ACM*, *37*(4), 66-75.

Norshuhada, S., & Shahizan, H. (2010). *Design research in software development: constructing and linking research questions, objectives, methods and outcomes*. Sintok: University Utara Malaysia Press.

Patton, M. Q. (2002). *Qualitative research and evaluation methods* (3rd ed.). Thousand Oaks, CA: Sage.

Peters, D. & Parnas, L. (2002). Requirements based monitors for real time systems. *IEEE Transactions on software engineering, 28*(2), 146-158.

Ram, S. (2008). Teaching data normalization: Traditional classroom methods versus online visual methods – a Literature review. In *Proceedings of the 21st Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2008)*, held on July 4-7, 2008 at Auckland, New Zealand (pp. 327-330). Auckland: NACCQ.

Rob, P., & Coronel, C. (2009). *Database Systems: Design, Implementation, and Management* (8th ed.). Boston: Course Technology.

Rubin, H. J. & Rubin, I. S. (2005). *Qualitative interviewing: The art of hearing data*. Thousand Oaks, CA: Sage.

Shelly, G. B., Cashman, T. J., & Rosenblatt, H. J. (2009). *Systems Analysis and Design* (8th ed.). Boston, MA: Course Technology.

Srikanth, S., & Sudarshan, D. (2001). *Database management Systems*. (1st ed.). Bangalore: Subhas.

Teorey, T., Lightstone, S., Nadeau, T., & Jagadish, H. (2011). *Database Modeling and Design: Logical Design* (5th ed.). Morgan Kaufmann: Morgan Kaufmann.

Vaishnavi, V. & Kuechler, W. (2008). *Design science research methods and patterns innovating information and communication technology research in information systems*. New York: Auerbach.

Whitten, J., Bentley, L., & Dittman, K. (2001). Systems analysis and design methods (5th ed.). New York: McGraw-Hill.

Yazici, A., & Karakaya, Z. (2007). JMathNorm: A database normalization tool using mathematica. In *Proceedings of International conference on computational science, 2007*, held on May 27-30, 2007 at Beijing, China (pp. 186-193). Beijing: Springer-Verlag Berlin Heidelberg.

Yi, T., Wu, F., & Gan, C. (2005). A comparison of metrics for UML class diagrams. *ACM SIGSOFT Software Engineering Notes*, *29*(5). 1–6.

Yu, C. H. (2000). *An introduction to computing and interpreting Cronbach Coefficient Alpha in SAS* (Arizona State University Technical Report No. 246-26). Retrieved May 27, 2011, from http://www2.sas.com/proceedings/sugi26/p246-26.pdf.

Appendix A: The Interview with the SerindIT UUM Company

The interview carried out as part of the initial stage of the research for purpose of gathering more information on the database normalization and exploring how do IT company's database designers achieve normalization process and what is the most used types of normalization process along with its importance to projects work.

The interview held on SerindIT UUM Company in Sintok city. Three of the company IT staff who work as database designer, database Administrator and software programming and developer were interviewed. They were asked the following questions:

1. What methodologies do you use in designing a relational database?

2. How far do you take the normalization process and to which level? Why? How?

3. Is normalization important?

Summary of Responses

<u>**Question 1:** What methodologies do you use in designing a relational database?</u>

Database Designer

- Through identifying the input sources and output requirements.
- Through building the Entity relationship diagram (ERD).

Database Administrator

- Through identifying the requirements.
- By using composite keys in other tables with caution.

Developer

- Through building the Entity relationship diagram (ERD).

- By resolving the many to many relationships.

Question 2: How far do you take the normalization process? Why?

Database Designer

- Each entity has its own table – usually forces till the 3NF.
- 4NF doesn’t occur.

Database Administrator

- I only go to 3NF.
- The 4NF is not done often.

Developer

- Never go beyond the 3NF.

Question 3: Is normalization important?

Database Designer

- Normalization comes to the picture clearly especially once you solve issues of someone else work.

Database Administrator

- It is good to have till the 3NF to avoid the changes

Developer

- Normalization is important once we look at other people’s tables.

Appendix B:

COLLEGE OF ARTS & SCIENCES
UNIVERSITY UTARA MALAYSIA

Questionnaire for Database Normalizer Prototype

Dear Sir/Madam

In recent time, relational databases have been used widely in almost all commercial applications to store, manipulate and use huge data for a specific enterprises and decision making. One of the essential steps in designing relational databases is Normalization which used to produce a set of relations with desirable properties for enterprises data requirements.

This questionnaire designed to seek your assistance on the study entitled "**DATABASE NORMALIZER PROTOTYPE**". This work attempts to evaluate the prototype in terms of easiness of use and satisfaction. You have been selected to participate in this research being undertaken as part of my final project for MSc-IT degree in University Utara Malaysia.

Please be assured that the information you have provided is strictly confidential and for academic purpose only. Your feedback in making this study successful is highly appreciated.

Thanking you for precious time and cooperation. If you have any inquiry please do not hesitate to contact us.

Yours sincerely,	*Supervisor*
Ahmed Absi	**Assoc. Prof. Abd Ghani B. Golamdin**
MSc-IT Student	
College of Arts & Sciences	
University Utara Malaysia	
Email: ahmed_absi2005@yahoo.com	

The following questions pertain to DATABASE NORMALIZER PROTOTYPE. For each question, kindly tick the options that come closest to your view and please try to be as accurate as possible when choosing your answers as your answers will be very important for making critical decisions.

Section A: (Demographic questions)
Please tick (/) or fill up the box at the appropriate blank

1. **User**:
   - ☐ Student
   - ☐ DB Developer/Designer
   - ☐ Other (please specify) …………………………………………………

2. **Gender**: ☐ Male ☐ Female

3. **Age**:
   - ☐ 18-29 Years
   - ☐ 30-39 Years
   - ☐ 40-50 Years
   - ☐ 51 and above

4. **Educational Background**:
   - ☐ Bachelors' Degree
   - ☐ Masters Degree
   - ☐ Doctoral
   - ☐ Other (please specify)…………………………………………………...

Section B

Please tick Strongly Disagree (SD), Disagree (D), Neutral (N), Agree (A), and Strongly Agree (SA) in the column that best represent your opinion in each of the statements in the table below.

Easiness Of Use		**SD**	**D**	**N**	**A**	**SA**
Q.1	Normalizing relations presented in straight-forward manner					
Q.2	The prototype is easy to use and requires fewest steps to accomplish desired tasks.					
Q.3	The prototype visual layout is clear.					
Q.4	User friendly and easier to get the result.					
Q.5	Are you interesting in using this system?					
Q.6	All users can use it without any difficulty.					

Satisfaction		**SD**	**D**	**N**	**A**	**SA**
Q.7	Terminologies related to task are appropriate.					
Q.8	All functionality is clearly labeled					
Q.9	All necessary functionality is available					
Q.10	The prototype display appropriate messages in case of any error					
Q.11	Are you satisfied that the prototype information is useful for your purpose?					

Thanks for Your Participation

Appendix C: The Algorithm Implementation

Appendix C:

1- First Normal Form Algorithm Source Code Implementation

```
' Filling out the rest of fileds
 Dim db As Database, rs As Recordset
 Dim AA_Array(100) As String
 If SKF = False Then
MsgBox "Please Select Primary Keys", , "DB Normalizer"
Exit Sub
End If
 Copy_Table FileName, MainTableName '-- here create 1NF_MainTable
 If SQL_Flag = True Then GoTo SQL_Lab '''------------------- JUMP -------- Applying 1NF Command ---
 Set db = OpenDatabase(FileName)
 Set rs = db.OpenRecordset("1NF_" + MainTableName)
 AA_Counter = 0
 While Not rs.EOF
  For i = 0 To rs.Fields.Count - 1
   If rs.Fields(i).Type = dbText Then
    If rs.Fields(i).Value <> "" Then
     AA_Array(i) = rs.Fields(i).Value
    Else
     rs.Edit
     rs.Fields(i).Value = AA_Array(i)
     rs.Update
    End If
   Else
    If rs.Fields(i).Value <> "" Then
     AA_Array(i) = Str(rs.Fields(i).Value)
    Else
     rs.Edit
     rs.Fields(i).Value = Val(AA_Array(i))
     rs.Update
    End If
   End If
  Next i
  rs.MoveNext
 Wend
 rs.Close
 Data1.RecordSource = "1NF_" & MainTableName
 Data1.Refresh
 MsgBox "First Normal Form Applied Successfully", , "DB Normalizer"
 NF1 = True
 Command4.Enabled = True
Exit Sub
'''----------------------------------------------------- END ACCESS-- Applying 1NF Command -------------
```

```
SQL_Lab:
Dim dbconn As New ADODB.Connection
Dim rs1 As New ADODB.Recordset
 Set dbconn = SQL_OpenDatabase(FileName)
 rs1.LockType = 3   '" for update
 rs1.Open "Select * from F1NF_" + MainTableName, dbconn
 AA_Counter = 0
 While Not rs1.EOF
  For i = 0 To rs1.Fields.Count - 1
   If rs1.Fields(i).Type = adVarChar Then   '"Varchar(50)
    If rs1.Fields(i).Value <> "" Then
     AA_Array(i) = rs1.Fields(i).Value
    Else
     'rs1.Edit
     rs1.Fields(i).Value = AA_Array(i)
     rs1.Update
    End If
   Else
    If rs1.Fields(i).Value <> "" Then
     AA_Array(i) = Str(rs1.Fields(i).Value)
    Else
     'rs1.Edit
      rs1.Fields(i).Value = Val(AA_Array(i))
     rs1.Update
    End If
   End If
  Next i
  rs1.MoveNext
 Wend
 rs1.Close
 dbconn.Close
 display_datagrid "Select * from " & "F1NF_" & MainTableName
 'Data1.RecordSource = "F1NF_" & MainTableName
 'Data1.Refresh
  MsgBox "First Normal Form Applied Successfully", , "DB Normalizer"
 NF1 = True
 Command4.Enabled = True
End Sub
```

2- Second Normal Form Algorithm Source Code Implementation

```
  Dim db As Database
  Dim TableName As String, td As TableDef, f As Field
  Dim A_Tables(30, 30) As Integer ' 30 subtables max, each 30 fileds max
  Dim PK As String, rs As Recordset, MyIndex  As Integer


If NF1 = False Then
MsgBox "Please Perform 1NF First", , "DB Normalizer"
Exit Sub
End If


If SQL_Flag = True Then GoTo SQL_Lab '''---------------- JUMP to SQL PART  2NF-------------Applying
2NF----

  Set db = OpenDatabase(CommonDialog1.FileName)
  Set rs = db.OpenRecordset("1NF_" & MainTableName)
  Set td = db.TableDefs("1NF_" & MainTableName)

  If (rs.BOF = True) And (rs.EOF = True) Then
   MsgBox "There is no data in the table"
   Exit Sub
  End If




  Dim keyInfo As String, otherInfo As String, CanSeparate As Boolean '''' Keyinfo changed to string
  Dim otherInfoInt As Integer, otherInfoDate As Date, otherInfoCurrency As Single
  Dim DoneValues(100) As String, otherInfoText As String ''------          DoneValue changed to string
  Dim DoneCounter As Integer, NextTableIndex As Integer
  DoneCounter = 0
  Dim A_Row As Integer, A_Col As Integer, cancelSearch As Boolean
  A_Row = 1: A_Col = 1 ' the 0 contins the number of fields in this table
  For i = 0 To Form4.List2.ListCount - 1 ' for ech primary key
   PK = Form4.List2.List(i)
   rs.MoveFirst
   ' this for for finding the index of the primary key in the table
   For j = 0 To rs.Fields.Count - 1
    If rs.Fields(j).Name = PK Then MyIndex = j
   Next j
   ' finding the partial dependencies
   keyInfo = "": otherInfoText = "": otherInfoInt = 0 ''------changed keyinfo=-1
   otherInfoCurrency = 0
   For j = 0 To rs.Fields.Count - 1
    CanSeparate = True
    DoneCounter = 0
    Set f = td.Fields(j)
    CanSeparate = True
```

```
' find non  searched key
2:      rs.MoveFirst
     cancelSearch = True
     While (cancelSearch = True) And (Not rs.EOF)
     keyInfo = rs.Fields(MyIndex).Value
     cancelSearch = False
     For w = 1 To DoneCounter
      If keyInfo = DoneValues(w) Then cancelSearch = True
     Next w
     If cancelSearch = True Then rs.MoveNext
     Wend
     If rs.EOF Then GoTo 1
    keyInfo = ""                                    '''----changed keyinfo=-1
    While (Not rs.EOF) And (CanSeparate = True)
    If f.Type = dbLong Then
     If keyInfo = "" Then                           '--- changed  keyinfo=-1
      keyInfo = rs.Fields(MyIndex).Value
      otherInfoInt = rs.Fields(j).Value
      DoneCounter = DoneCounter + 1
      DoneValues(DoneCounter) = keyInfo
     Else
      If keyInfo = rs.Fields(MyIndex).Value Then
       If otherInfoInt <> rs.Fields(j).Value Then CanSeparate = False
      End If
     End If

    ElseIf f.Type = dbText Then
     If keyInfo = "" Then                           '--- changed  keyinfo=-1
      keyInfo = rs.Fields(MyIndex).Value
      otherInfoText = rs.Fields(j).Value
      DoneCounter = DoneCounter + 1
      DoneValues(DoneCounter) = keyInfo
     Else
      If keyInfo = rs.Fields(MyIndex).Value Then
       If otherInfoText <> rs.Fields(j).Value Then CanSeparate = False
      End If
     End If

    ElseIf f.Type = dbCurrency Then
     If keyInfo = "" Then                           '--- changed  keyinfo=-1
      keyInfo = rs.Fields(MyIndex).Value
      otherInfoCurrency = rs.Fields(j).Value
      DoneCounter = DoneCounter + 1
      DoneValues(DoneCounter) = keyInfo
     Else
      If keyInfo = rs.Fields(MyIndex).Value Then
       If otherInfoCurrency <> rs.Fields(j).Value Then CanSeparate = False
      End If
     End If

    ElseIf f.Type = dbDate Then
```

```
     If keyInfo = "" Then                              '--- changed  keyinfo=-1
      keyInfo = rs.Fields(MyIndex).Value
      otherInfoDate = rs.Fields(j).Value
      DoneCounter = DoneCounter + 1
      DoneValues(DoneCounter) = keyInfo
     Else
      If keyInfo = rs.Fields(MyIndex).Value Then
       If otherInfoDate <> rs.Fields(j).Value Then CanSeparate = False
      End If
     End If
    End If
    rs.MoveNext
    Wend
    If CanSeparate = True Then GoTo 2
1:    If CanSeparate Then
     A_Col = A_Col + 1: A_Tables(A_Row, A_Col) = j
    End If
   Next j
   A_Tables(A_Row, 1) = A_Col
   A_Row = A_Row + 1: A_Col = 1
  Next i
  ' the exact number of trables is row-1
  A_Row = A_Row - 1


 ' now creating the new tables

 NextTableIndex = 1 ' for creation purposes like Table1 and  Table2 ...
 Dim newTD As TableDef
 Dim newF As Field, newDB As Database
 Set newDB = OpenDatabase(FileName)
 For i = 1 To A_Row
  Set newTD = New TableDef
  For j = 2 To A_Tables(i, 1)
   Set f = td.Fields(A_Tables(i, j))
   Select Case f.Type
    Case dbLong:   Set newF = newTD.CreateField(rs.Fields(A_Tables(i, j)).Name, dbLong)
    Case dbText:   Set newF = newTD.CreateField(rs.Fields(A_Tables(i, j)).Name, dbText, 50)
    Case dbCurrency:   Set newF = newTD.CreateField(rs.Fields(A_Tables(i, j)).Name, dbCurrency)
    Case dbDate:   Set newF = newTD.CreateField(rs.Fields(A_Tables(i, j)).Name, dbDate)
   End Select


'On Error GoTo lab1

  newTD.Fields.Append newF
  Next j
  newTD.Name = "Table" & NextTableIndex & "_" & MainTableName
  NextTableIndex = NextTableIndex + 1
  newDB.TableDefs.Append newTD
 Next i
```

```
' now creating the other tables that contain the composite primary key
 Set newTD = New TableDef
' first of all, creating the table that contains the promary keys
 For i = 0 To Form4.List2.ListCount - 1 ' for ech primary key
   PK = Form4.List2.List(i)
   rs.MoveFirst
   ' this for for finding the index of the primary key in the table
   For j = 0 To rs.Fields.Count - 1
    If rs.Fields(j).Name = PK Then MyIndex = j
   Next j
   Set f = td.Fields(MyIndex)
   Select Case f.Type
    Case dbLong:   Set newF = newTD.CreateField(rs.Fields(MyIndex).Name, dbLong)
    Case dbText:   Set newF = newTD.CreateField(rs.Fields(MyIndex).Name, dbText, 50)
    Case dbCurrency:   Set newF = newTD.CreateField(rs.Fields(MyIndex).Name, dbCurrency)
    Case dbDate:   Set newF = newTD.CreateField(rs.Fields(MyIndex).Name, dbDate)
   End Select

   newTD.Fields.Append newF
 Next i

 Dim NextFieldIndex As Integer, found As Boolean, AllFields As Integer
 ' creating the fields that are not PK
 For AllFields = 0 To rs.Fields.Count - 1
  found = False
  For i = 1 To A_Row
   For j = 2 To A_Tables(i, 1)
    NextFieldIndex = A_Tables(i, j)
    If NextFieldIndex = AllFields Then found = True
   Next j
  Next i
  If found = False Then
   Set f = td.Fields(NextFieldIndex)
   Select Case f.Type
    Case dbLong:   Set newF = newTD.CreateField(rs.Fields(AllFields).Name, dbLong)
    Case dbText:   Set newF = newTD.CreateField(rs.Fields(AllFields).Name, dbText, 50)
    Case dbCurrency:   Set newF = newTD.CreateField(rs.Fields(AllFields).Name, dbCurrency)
    Case dbDate:   Set newF = newTD.CreateField(rs.Fields(AllFields).Name, dbDate)
   End Select
    newTD.Fields.Append newF
  End If
 Next AllFields


newTD.Name = "Table" & NextTableIndex & "_" & MainTableName
NextTableIndex = NextTableIndex + 1
newDB.TableDefs.Append newTD
Dim newRS As Recordset
 ' removing the redandancies from the created tables
 Dim coll As New Collection, stPos As String
 On Error Resume Next
 Err.Clear
' filling the values
 For i = 1 To A_Row
  Set newRS = newDB.OpenRecordset("Table" & i & "_" & MainTableName)
  rs.MoveFirst
  While Not rs.EOF
   ' check for redandant
   stPos = "("
   For j = 0 To newRS.Fields.Count - 1
    If rs.Fields(newRS.Fields(j).Name).Type = dbText Then
     stPos = stPos + rs.Fields(newRS.Fields(j).Name).Value
    Else
     stPos = stPos + Str(rs.Fields(newRS.Fields(j).Name).Value)
    End If
    If j <> newRS.Fields.Count - 1 Then
     stPos = stPos + ","
    Else
     stPos = stPos + ")"
    End If
   Next j
   coll.Add stPos, stPos
   If Err.Number = 0 Then
    newRS.AddNew
    For j = 0 To newRS.Fields.Count - 1
     newRS.Fields(j).Value = rs.Fields(newRS.Fields(j).Name).Value
    Next j
    newRS.Update
   Else
    Err.Clear
   End If
   rs.MoveNext
  Wend
  newRS.Close
 Next i

 ' filling the last table values that contains the composite key
 i = NextTableIndex - 1
 Set newRS = newDB.OpenRecordset("Table" & i & "_" & MainTableName)
 rs.MoveFirst
  While Not rs.EOF
   ' check for redandant
   stPos = "("
   For j = 0 To newRS.Fields.Count - 1
    If rs.Fields(newRS.Fields(j).Name).Type = dbText Then
     stPos = stPos + rs.Fields(newRS.Fields(j).Name).Value
    Else
     stPos = stPos + Str(rs.Fields(newRS.Fields(j).Name).Value)
    End If
    If j <> newRS.Fields.Count - 1 Then
     stPos = stPos + ","
    Else
     stPos = stPos + ")"
    End If
   Next j
   coll.Add stPos, stPos
   If Err.Number = 0 Then
    newRS.AddNew
    For j = 0 To newRS.Fields.Count - 1
     newRS.Fields(j).Value = rs.Fields(newRS.Fields(j).Name).Value
    Next j
    newRS.Update
   Else
    Err.Clear
   End If
   rs.MoveNext
  Wend
 newRS.Close
 rs.Close


 MsgBox "Second Normal Form Applied Successfully", , "DB Normalizer"

 For i = 1 To A_Row + 1
 Display_Table i, "Table" & i & "_" & MainTableName
 Next i



 NF2 = True
 Command5.Enabled = True
 Exit Sub
 lab1:
 If Err.Number = 3010 Then
 MsgBox "DataBase alread Normalized"
 newDB.TableDefs.Delete Table1
 End If

 Exit Sub
 '''-------------------------------------- END OF ACCESS 2NF -------------------Applying 2NF---------------------
 -
 SQL_Lab:

 Dim dbconn As ADODB.Connection
 Dim rs1 As ADODB.Recordset
 Dim tdrs As ADODB.Recordset

 Set dbconn = SQL_OpenDatabase(FileName)
 Set rs1 = New ADODB.Recordset
 Set tdrs = New ADODB.Recordset
```

```
If dbconn.State = 1 Then
If rs1.State = 1 Then
       rs1.Close
    End If
Else
MsgBox "No Database Connection"
Exit Sub
End If
rs1.Open "select * from " & "F1NF_" & MainTableName, dbconn
'MsgBox rs1.Fields(0).Value


"--Set td = db.TableDefs("F1NF_" & MainTableName)
sql_query = "SELECT c.name ColumnName, t.name ColumnType " & _
         " FROM sys.columns AS c " & _
         " JOIN sys.types as t ON c.user_type_id=t.user_type_id " & _
         " where OBJECT_NAME(c.OBJECT_ID)='" & "F1NF_" & MainTableName & "'"

tdrs.CursorLocation = adUseServer
tdrs.Open sql_query, dbconn

'MsgBox tdrs.Fields(0).Value

Dim tdA() As String
ReDim tdA(0 To 1, 0 To 20)
t = -1
 While Not tdrs.EOF
 t = t + 1
 tdA(0, t) = tdrs.Fields(0).Value
 tdA(1, t) = tdrs.Fields(1).Value
 'MsgBox Str(t) + tdA(0, t) + tdA(1, t)
 tdrs.MoveNext
 Wend

 Dim ColNo As Integer
 ColNo = t + 1

  If (rs1.BOF = True) And (rs1.EOF = True) Then
   MsgBox "There is no data in the table"
   Exit Sub
  End If



  'Dim keyInfo As String, otherInfo As String, CanSeparate As Boolean "'' Keyinfo changed to string
  'Dim otherInfoInt As Integer, otherInfoDate As Date, otherInfoCurrency As Single
  'Dim DoneValues(100) As String, otherInfoText As String "------            DoneValue changed to string
  'Dim DoneCounter As Integer, NextTableIndex As Integer
  'Dim A_Row As Integer, A_Col As Integer, cancelSearch As Boolean
  DoneCounter = 0
  A_Row = 1: A_Col = 1 ' the 0 contins the number of fields in this table
  For i = 0 To Form4.List2.ListCount - 1 ' for ech primary key
   PK = Form4.List2.List(i)
   rs1.MoveFirst
   ' this for for finding the index of the primary key in the table


   For j = 0 To rs1.Fields.Count - 1
  ''' While Not tdd.EOF

    If rs1.Fields(j).Name = PK Then MyIndex = j
   Next j
 '''  Wend


   ' finding the partial dependencies
   keyInfo = "": otherInfoText = "": otherInfoInt = 0 ''------changed keyinfo=-1
   otherInfoCurrency = 0


   For j = 0 To rs1.Fields.Count - 1
    CanSeparate = True
    DoneCounter = 0

   ' Set f = td.Fields(j)


    CanSeparate = True
    ' find non  searched key
22:      rs1.MoveFirst
     cancelSearch = True
     While (cancelSearch = True) And (Not rs1.EOF)
     keyInfo = rs1.Fields(MyIndex).Value
     cancelSearch = False
     For w = 1 To DoneCounter
       If keyInfo = DoneValues(w) Then cancelSearch = True
     Next w
     If cancelSearch = True Then rs1.MoveNext
     Wend
     If rs1.EOF Then GoTo 11

    'adInteger  adVarChar adDate adSingle adDouble

    keyInfo = ""                                    '''----changed keyinfo=-1
    While (Not rs1.EOF) And (CanSeparate = True)

    'If f.Type = adInteger Then
    If tdA(1, j) = "int" Then
    If keyInfo = "" Then                           '--- changed  keyinfo=-1
     keyInfo = rs1.Fields(MyIndex).Value
     otherInfoInt = rs1.Fields(j).Value
     DoneCounter = DoneCounter + 1
     DoneValues(DoneCounter) = keyInfo
    Else
     If keyInfo = rs1.Fields(MyIndex).Value Then
       If otherInfoInt <> rs1.Fields(j).Value Then CanSeparate = False
     End If
    End If

   ' ElseIf f.Type = adVarChar Then
    ElseIf tdA(1, j) = "varchar" Then
    If keyInfo = "" Then                          '--- changed  keyinfo=-1
     keyInfo = rs1.Fields(MyIndex).Value
     otherInfoText = rs1.Fields(j).Value
     DoneCounter = DoneCounter + 1
     DoneValues(DoneCounter) = keyInfo
    Else
     If keyInfo = rs1.Fields(MyIndex).Value Then
       If otherInfoText <> rs1.Fields(j).Value Then CanSeparate = False
     End If
    End If

    'ElseIf f.Type = adCurrency Then
    ElseIf tdA(1, j) = "money" Then
    If keyInfo = "" Then                          '--- changed  keyinfo=-1
     keyInfo = rs1.Fields(MyIndex).Value
     otherInfoCurrency = rs1.Fields(j).Value
     DoneCounter = DoneCounter + 1
     DoneValues(DoneCounter) = keyInfo
    Else
     If keyInfo = rs1.Fields(MyIndex).Value Then
       If otherInfoCurrency <> rs1.Fields(j).Value Then CanSeparate = False
     End If
    End If

    'ElseIf f.Type = adDate Then
    ElseIf tdA(1, j) = "datetime" Then
    If keyInfo = "" Then                          '--- changed  keyinfo=-1
     keyInfo = rs1.Fields(MyIndex).Value
     otherInfoDate = rs1.Fields(j).Value
     DoneCounter = DoneCounter + 1
     DoneValues(DoneCounter) = keyInfo
    Else
     If keyInfo = rs1.Fields(MyIndex).Value Then
       If otherInfoDate <> rs1.Fields(j).Value Then CanSeparate = False
     End If
    End If
    End If
    rs1.MoveNext
    Wend
    If CanSeparate = True Then GoTo 22
```

```
11:    If CanSeparate Then
     A_Col = A_Col + 1: A_Tables(A_Row, A_Col) = j
    End If

   Next j
   ''tdd.MoveNext
   'Wend

   A_Tables(A_Row, 1) = A_Col
   A_Row = A_Row + 1: A_Col = 1

  Next i
  ' the exact number of trables is row-1
  A_Row = A_Row - 1



  ''-Dim newTD As TableDef
 ''-Dim newF As Field, newDB As Database
 '''Set newDB = OpenDatabase(FileName)   '--- no need

  ' Now creating the new tables
  Dim cn As ADODB.Connection
  Dim cmd As New ADODB.Command
  NextTableIndex = 1 ' for creation purposes like Table1 and  Table2 ...

 For i = 1 To A_Row
'  Set newTD = New TableDef
 NewTabName = "Table" & NextTableIndex & "_" & MainTableName

    For j = 2 To A_Tables(i, 1)
  ' Set f = td.Fields(A_Tables(i, j))

   'Select Case tdA(1, A_Tables(i, j))
    'Case adInteger:   Set newF = newTD.CreateField(rs1.Fields(A_Tables(i, j)).Name, dbLong)
    'Case adVarChar:   Set newF = newTD.CreateField(rs1.Fields(A_Tables(i, j)).Name, dbText, 50)
    'Case adCurrency:   Set newF = newTD.CreateField(rs1.Fields(A_Tables(i, j)).Name, dbCurrency)
    'Case adDate:   Set newF = newTD.CreateField(rs1.Fields(A_Tables(i, j)).Name, dbDate)

   'End Select

   'newTD.Fields.Append newF

  Set cn = SQL_OpenDatabase(FileName)

  cmd.ActiveConnection = cn
  cmd.CommandType = adCmdText

  If tdA(1, A_Tables(i, j)) = "varchar" Then
  dtype = "varchar(50)"
  Else
  dtype = tdA(1, A_Tables(i, j))
  End If

  If j = 2 Then
    cmd.CommandText = "CREATE TABLE " + NewTabName + " ( " + tdA(0, A_Tables(i, j)) + " " +
dtype + ")"
  Else
    cmd.CommandText = "ALTER TABLE " + NewTabName + " ADD " + tdA(0, A_Tables(i, j)) + " " +
dtype
  End If

  cmd.Execute
  cn.Close

  Next j

 ' newTD.Name = "Table" & NextTableIndex & "_" & MainTableName
  NextTableIndex = NextTableIndex + 1
  'newDB.TableDefs.Append newTD
 Next i

' Now creating the other tables that contain the composite primary key

 '--Set newTD = New TableDef
' first of all, creating the table that contains the promary keys


NewTabName = "Table" & NextTableIndex & "_" & MainTableName
 For i = 0 To Form4.List2.ListCount - 1 ' for ech primary key
   PK = Form4.List2.List(i)
   rs1.MoveFirst

   ' this for for finding the index of the primary key in the table
   For j = 0 To rs1.Fields.Count - 1
    If rs1.Fields(j).Name = PK Then MyIndex = j
   Next j



   'Set f = td.Fields(MyIndex)
   'Select Case f.Type
   ' Case dbLong:   Set newF = newTD.CreateField(rs1.Fields(MyIndex).Name, dbLong)
   ' Case dbText:   Set newF = newTD.CreateField(rs1.Fields(MyIndex).Name, dbText, 50)
   ' Case dbCurrency:   Set newF = newTD.CreateField(rs1.Fields(MyIndex).Name, dbCurrency)
   ' Case dbDate:   Set newF = newTD.CreateField(rs1.Fields(MyIndex).Name, dbDate)
   'End Select

   'newTD.Fields.Append newF
    Set cn = SQL_OpenDatabase(FileName)

  cmd.ActiveConnection = cn
  cmd.CommandType = adCmdText
  If tdA(1, MyIndex) = "varchar" Then
  dtype = "varchar(50)"
  Else
  dtype = tdA(1, MyIndex)
  End If

  If i = 0 Then
    cmd.CommandText = "CREATE TABLE " + NewTabName + " ( " + tdA(0, MyIndex) + " " + dtype +
")"
  Else
    cmd.CommandText = "ALTER TABLE " + NewTabName + " ADD " + tdA(0, MyIndex) + " " +
dtype
  End If

  cmd.Execute
  cn.Close

 Next i

 "--Dim NextFieldIndex As Integer, found As Boolean, AllFields As Integer
 ' creating the fields that are not PK
 'Dim found As Boolean
 For AllFields = 0 To rs1.Fields.Count - 1
  found = False
  For i = 1 To A_Row
   For j = 2 To A_Tables(i, 1)
    NextFieldIndex = A_Tables(i, j)
    If NextFieldIndex = AllFields Then found = True
   Next j
  Next i
  If found = False Then
    Set cn = SQL_OpenDatabase(FileName)

  cmd.ActiveConnection = cn
  cmd.CommandType = adCmdText

  If tdA(1, AllFields) = "varchar" Then
  dtype = "varchar(50)"
  Else
  dtype = tdA(1, AllFields)
  End If

  cmd.CommandText = "ALTER TABLE " + NewTabName + " ADD " + tdA(0, AllFields) + " " + dtype
  cmd.Execute
  cn.Close
  End If

 Next AllFields

'newTD.Name = "Table" & NextTableIndex & "_" & MainTableName
NextTableIndex = NextTableIndex + 1
```

```
'newDB.TableDefs.Append newTD

Dim newRS1 As New ADODB.Recordset
 ' removing the redandancies from the created tables
 ''--Dim coll As New Collection, stPos As String
 On Error Resume Next
 Err.Clear
' filling the values
 For i = 1 To A_Row

  'Set newrs1 = newDB.OpenRecordset("Table" & i & "_" & MainTableName)
  newRS1.LockType = 3
  newRS1.Open "Select * from " & "Table" & i & "_" & MainTableName, dbconn

  rs1.MoveFirst
  'MsgBox rs1.Fields(0).Value
  While Not rs1.EOF
   ' check for redandant
   stPos = "("
   For j = 0 To newRS1.Fields.Count - 1
    If rs1.Fields(newRS1.Fields(j).Name).Type = adVarChar Then
     stPos = stPos + rs1.Fields(newRS1.Fields(j).Name).Value
    Else
     stPos = stPos + Str(rs1.Fields(newRS1.Fields(j).Name).Value)
    End If
    If j <> newRS1.Fields.Count - 1 Then
     stPos = stPos + ","
    Else
     stPos = stPos + ")"
    End If
   Next j
   coll.Add stPos, stPos
   If Err.Number = 0 Then
    newRS1.AddNew

     For j = 0 To newRS1.Fields.Count - 1
     newRS1.Fields(j).Value = rs1.Fields(newRS1.Fields(j).Name).Value
    Next j
    newRS1.Update
    'MsgBox newRS1.Fields(0).Value
   Else
    Err.Clear
   End If
   rs1.MoveNext
  Wend
  newRS1.Close
 Next i
 ' Filling the last table values that contains the composite key
 i = NextTableIndex - 1

 'Set newrs1 = newDB.OpenRecordset("Table" & i & "_" & MainTableName)
```

```
  newRS1.LockType = 3
  newRS1.Open "Select * from " & "Table" & i & "_" & MainTableName, dbconn
 rs1.MoveFirst
  While Not rs1.EOF
   ' check for redandant
   stPos = "("
   For j = 0 To newRS1.Fields.Count - 1
    If rs1.Fields(newRS1.Fields(j).Name).Type = adVarChar Then
     stPos = stPos + rs1.Fields(newRS1.Fields(j).Name).Value
    Else
     stPos = stPos + Str(rs1.Fields(newRS1.Fields(j).Name).Value)
    End If
    If j <> newRS1.Fields.Count - 1 Then
     stPos = stPos + ","
    Else
     stPos = stPos + ")"
    End If
   Next j
   coll.Add stPos, stPos
   If Err.Number = 0 Then
    newRS1.AddNew
    For j = 0 To newRS1.Fields.Count - 1
     newRS1.Fields(j).Value = rs1.Fields(newRS1.Fields(j).Name).Value
    Next j
    newRS1.Update
   Else
    Err.Clear
   End If
   rs1.MoveNext
  Wend
 newRS1.Close
 rs1.Close
 MsgBox "Second Normal Form Applied Successfully", , "DB Normalizer"
 For i = 1 To A_Row + 1
 'Display_Table i, "Table" & i & "_" & MainTableName
 display_datagrid "Select * from " & "Table" & i & "_" & MainTableName
 MsgBox "This is " & "Table" & i & "_" & MainTableName, , "DB Normalizer"
 Next i
 Command5.Enabled = True
 NF2 = True
 Exit Sub
lab11:
If Err.Number = 3010 Then
MsgBox "DataBase alread Normalized"
newDB.TableDefs.Delete Table1
End If
End Sub
```

3- Third Normal Form Algorithm Source Code Implementation

```
' ------------------------------------------ 3NF ------------------------------
  Dim db As Database
  Dim td As TableDef, TableName As String, rs As Recordset

If NF2 = False Then
MsgBox "Please Perform 2NF First", , "DB Normalizer"
Exit Sub
End If


 If SQL_Flag = True Then GoTo SQL_Lab  '''------------------- JUMP INTo SQL ---- 3NF -------------

  Set db = OpenDatabase(FileName)
  Form5.Combo1.Clear
  For Each td In db.TableDefs
   TableName = td.Name
   If Left$(TableName, 4) <> "MSys" And Left$(TableName, 5) = "Table" Then
         Form5.Combo1.AddItem TableName
  Next
  Form1.Tag = FileName
  Form5.Show vbModal
  If Form5.Tag = 0 Then Exit Sub
  If Form5.Tag = 1 Then
   ' the table in form5.combo1.text and the fields is in form5.list2
   Dim TheTableName As String
   TheTableName = Form5.Combo1.Text
   Set rs = db.OpenRecordset(TheTableName)
   Set td = db.TableDefs(TheTableName)

   ' now creating the new tables
   Dim NextTableIndex As Integer
   NextTableIndex = 1 ' for creation purposes like Table21 and  Table22 ...
                ' if we select the table2 for separation
   Dim newTD As TableDef
   Dim newF As Field, newDB As Database
   Set newDB = OpenDatabase(FileName)
   Set newTD = New TableDef
   For j = 0 To Form5.List1.ListCount - 1
    Set f = td.Fields(Form5.List1.List(j))
    Select Case f.Type
     Case dbLong:   Set newF = newTD.CreateField(Form5.List1.List(j), dbLong)
     Case dbText:   Set newF = newTD.CreateField(Form5.List1.List(j), dbText, 50)
     Case dbCurrency:   Set newF = newTD.CreateField(Form5.List1.List(j), dbCurrency)
     Case dbDate:   Set newF = newTD.CreateField(Form5.List1.List(j), dbDate)
    End Select

    newTD.Fields.Append newF
   Next j
   ' add the first field of the list2 as connection field
```

```
Set f = td.Fields(Form5.List2.List(0))
Select Case f.Type
 Case dbLong:   Set newF = newTD.CreateField(Form5.List2.List(0), dbLong)
 Case dbText:   Set newF = newTD.CreateField(Form5.List2.List(0), dbText, 50)
 Case dbCurrency:   Set newF = newTD.CreateField(Form5.List2.List(0), dbCurrency)
 Case dbDate:   Set newF = newTD.CreateField(Form5.List2.List(0), dbDate)
End Select

newTD.Fields.Append newF
newTD.Name = TheTableName & NextTableIndex
NextTableIndex = NextTableIndex + 1
newDB.TableDefs.Append newTD

Set newTD = New TableDef
For j = 0 To Form5.List2.ListCount - 1
 Set f = td.Fields(Form5.List2.List(j))
 Select Case f.Type
  Case dbLong:   Set newF = newTD.CreateField(Form5.List2.List(j), dbLong)
  Case dbText:   Set newF = newTD.CreateField(Form5.List2.List(j), dbText, 50)
  Case dbCurrency:   Set newF = newTD.CreateField(Form5.List2.List(j), dbCurrency)
  Case dbDate:   Set newF = newTD.CreateField(Form5.List2.List(j), dbDate)
 End Select

newTD.Fields.Append newF
Next j

newTD.Name = TheTableName & NextTableIndex
NextTableIndex = NextTableIndex + 1
newDB.TableDefs.Append newTD

' filling the values
Dim newRS As Recordset
' removing the redandancies from the created tables
Dim coll As New Collection, stPos As String
On Error Resume Next
Err.Clear
For i = 1 To NextTableIndex - 1
 Set newRS = newDB.OpenRecordset(TheTableName & i)
 rs.MoveFirst
 While Not rs.EOF
  ' check for redandant
  stPos = "("
  For j = 0 To newRS.Fields.Count - 1
   If rs.Fields(newRS.Fields(j).Name).Type = dbText Then
    stPos = stPos + rs.Fields(newRS.Fields(j).Name).Value
   Else
    stPos = stPos + Str(rs.Fields(newRS.Fields(j).Name).Value)
   End If
   If j <> newRS.Fields.Count - 1 Then
    stPos = stPos + ","
   Else
```

```
      stPos = stPos + ")"
     End If
    Next j
    coll.Add stPos, stPos
    If Err.Number = 0 Then
     newRS.AddNew
     For j = 0 To newRS.Fields.Count - 1
      newRS.Fields(j).Value = rs.Fields(newRS.Fields(j).Name).Value
     Next j
     newRS.Update
    Else
     Err.Clear
    End If
    rs.MoveNext
   Wend
   newRS.Close
  Next i
  rs.Close
 End If


 For i = 1 To 2
Display_Table i, TheTableName & i
Next i

 NF3 = True

 Exit Sub
"-------------------------------------------------------- END ACCESS ---------- 3NF -----------

SQL_Lab:


 Dim dbconn As ADODB.Connection
 Dim rs1 As New ADODB.Recordset


  Set dbconn = SQL_OpenDatabase(FileName)

 Form5.Combo1.Clear

 rs1.Open "SELECT name FROM sys.Tables", dbconn
 While Not rs1.EOF
 TableName = rs1.Fields(0).Value
 If Left$(TableName, 5) = "Table" Then Form5.Combo1.AddItem TableName
 rs1.MoveNext
 Wend
 rs1.Close


 Form1.Tag = FileName
```

```
   Form5.Show vbModal
   If Form5.Tag = 0 Then Exit Sub
   If Form5.Tag = 1 Then
    ' the table in form5.combo1.text and the fields is in form5.list2
    'Dim TheTableName As String
    TheTableName = Form5.Combo1.Text

    'Set rs = db.OpenRecordset(TheTableName)
    rs1.Open "SELECT * FROM " & TheTableName, dbconn

    'Set td = db.TableDefs(TheTableName)
    Dim tdrs As New ADODB.Recordset
    sql_query = "SELECT c.name ColumnName, t.name ColumnType " & _
         " FROM sys.columns AS c " & _
         " JOIN sys.types as t ON c.user_type_id=t.user_type_id " & _
         " where OBJECT_NAME(c.OBJECT_ID)='" & TheTableName & "'"

tdrs.CursorLocation = adUseServer
tdrs.Open sql_query, dbconn

Dim tdA() As String
ReDim tdA(0 To 1, 0 To 20)
t = -1
 While Not tdrs.EOF
 t = t + 1
 tdA(0, t) = tdrs.Fields(0).Value
 tdA(1, t) = tdrs.Fields(1).Value
 'MsgBox Str(t) + tdA(0, t) + tdA(1, t)
 tdrs.MoveNext
 Wend

 Dim ColNo As Integer
 ColNo = t + 1


    ' now creating the new tables
    'Dim NextTableIndex As Integer
    NextTableIndex = 1 ' for creation purposes like Table21 and  Table22 ...
                 ' if we select the table2 for separation


    Dim cn As ADODB.Connection
    Dim cmd As New ADODB.Command


    For j = 0 To Form5.List1.ListCount - 1
     'Set f = td.Fields(Form5.List1.List(j))
     For k = 0 To ColNo
     If Form5.List1.List(j) = tdA(0, k) Then
     fldno = k
     Exit For
   End If
   Next k

Set cn = SQL_OpenDatabase(FileName)

cmd.ActiveConnection = cn
cmd.CommandType = adCmdText

If tdA(1, fldno) = "varchar" Then
dtype = "varchar(50)"
Else
dtype = tdA(1, fldno)
End If

If j = 0 Then
  cmd.CommandText = "CREATE TABLE " + TheTableName + Chr(NextTableIndex + 48) + " ( " +
          tdA(0, fldno) + " " + dtype + ")"
Else
  cmd.CommandText = "ALTER TABLE " + TheTableName + Chr(NextTableIndex + 48) + " ADD " +
          tdA(0, fldno) + " " + dtype
End If

cmd.Execute
cn.Close

  Next j


  ' add the first field of the list2 as connection field

  'Set f = td.Fields(Form5.List2.List(0))
   For k = 0 To ColNo
   If Form5.List2.List(0) = tdA(0, k) Then
   fldno = k
   Exit For
   End If
   Next k

Set cn = SQL_OpenDatabase(FileName)
cmd.ActiveConnection = cn
cmd.CommandType = adCmdText

If tdA(1, fldno) = "varchar" Then
dtype = "varchar(50)"
Else
dtype = tdA(1, fldno)
End If

   cmd.CommandText = "ALTER TABLE " + TheTableName + Chr(NextTableIndex + 48) + " ADD " +
          tdA(0, fldno) + " " + dtype
  cmd.Execute
cn.Close

  NextTableIndex = NextTableIndex + 1

  'newDB.TableDefs.Append newTD
  'Set newTD = New TableDef

  For j = 0 To Form5.List2.ListCount - 1
   'Set f = td.Fields(Form5.List2.List(j))
   For k = 0 To ColNo
   If Form5.List2.List(j) = tdA(0, k) Then
   fldno = k
   Exit For
   End If
   Next k

Set cn = SQL_OpenDatabase(FileName)

cmd.ActiveConnection = cn
cmd.CommandType = adCmdText

If tdA(1, fldno) = "varchar" Then
dtype = "varchar(50)"
Else
dtype = tdA(1, fldno)
End If

If j = 0 Then
  cmd.CommandText = "CREATE TABLE " + TheTableName + Chr(NextTableIndex + 48) + " ( " +
         tdA(0, fldno) + " " + dtype + ")"
Else
  cmd.CommandText = "ALTER TABLE " + TheTableName + Chr(NextTableIndex + 48) + " ADD " +
         tdA(0, fldno) + " " + dtype
End If

cmd.Execute
cn.Close

  Next j

  'newTD.Name = TheTableName & NextTableIndex
  NextTableIndex = NextTableIndex + 1
  'newDB.TableDefs.Append newTD

  ' filling the values
  Dim newRS1 As New ADODB.Recordset
  ' removing the redandancies from the created tables
  'Dim coll As New Collection, stPos As String
  On Error Resume Next
  Err.Clear
```

```
    For i = 1 To NextTableIndex - 1

     'Set newRS = newDB.OpenRecordset(TheTableName & i)
     newRS1.LockType = 3
     newRS1.Open "Select * from " & TheTableName & i, dbconn

     rs1.MoveFirst
     While Not rs1.EOF
     'MsgBox rs1.Fields.Count
      ' check for redandant
      stPos = "("
      For j = 0 To newRS1.Fields.Count - 1
       If rs1.Fields(newRS1.Fields(j).Name).Type = adVarChar Then
        stPos = stPos + rs1.Fields(newRS1.Fields(j).Name).Value
       Else
        stPos = stPos + Str(rs1.Fields(newRS1.Fields(j).Name).Value)
       End If
       If j <> newRS1.Fields.Count - 1 Then
        stPos = stPos + ","
       Else
        stPos = stPos + ")"
       End If
      Next j
      coll.Add stPos, stPos
      If Err.Number = 0 Then
       newRS1.AddNew
       For j = 0 To newRS1.Fields.Count - 1
        newRS1.Fields(j).Value = rs1.Fields(newRS1.Fields(j).Name).Value
       Next j
       newRS1.Update
      Else
       Err.Clear
      End If
      rs1.MoveNext
     Wend
     newRS1.Close
    Next i
    rs1.Close
   End If

  For i = 1 To 2
 'Display_Table i, TheTableName & i
 display_datagrid "Select * from " & TheTableName & i
 MsgBox "This is " & TheTableName & i, , "DB Normalizer"
 Next i

   NF3 = True

End Sub
```