



Jaypee University of Information Technology
Solan (H.P.)
LEARNING RESOURCE CENTER

Acc. Num. **SP03013** Call Num:

General Guidelines:

- ◆ Library books should be used with great care.
- ◆ Tearing, folding, cutting of library books or making any marks on them is not permitted and shall lead to disciplinary action.
- ◆ Any defect noticed at the time of borrowing books must be brought to the library staff immediately. Otherwise the borrower may be required to replace the book by a new copy.
- ◆ The loss of LRC book(s) must be immediately brought to the notice of the Librarian in writing.

Learning Resource Centre-JUIT



SP03013

**SYNTHESIZING NORMALIZED RELATIONS
FROM
A UNIVERSAL RELATION**

A PROJECT REPORT

Submitted by

**Aayush Singal (031401)
Saurabh Srivastava (031402)**

in partial fulfillment for the award of the degree

of

**BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY**



**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT**

MAY 2007

CERTIFICATE

This is to certify that the work entitled, "Synthesizing Normalized Relations from a Universal Set of Relation" submitted by Aayush Singal and Saurabh Srivastava in partial fulfillment for the award of degree of Bachelor of Technology in Information Technology of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.



Mr. Vipin Arora

PROJECT SUPERVISOR

**LECTURER
(CSE & IT)**

Jaypee University of
Information Technology,
Waknaghat, Solan (H.P)

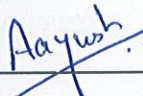


Saurabh Srivastava (IT-1402)

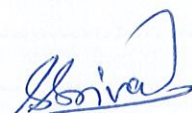
ACKNOWLEDGEMENT

No task is a single man's effort. Various factors, situations and persons integrate to provide the background for accomplishment of a task. We would wish to convey our sincere gratitude to our project supervisor and guide **Mr. Vipin Arora**, Lecturer, Jaypee University of Information Technology for his valuable and inspiring guidance, constant encouragement and constructive criticism throughout the duration of the project which helped us work upon the project as well as the report and refine wherever possible.

This opportunity is also availed by the group to thank each other for their constant support and encouragement.



Aayush Singal (031401)



Saurabh Srivastava (031402)

TABLE OF CONTENTS

ACKNOWLEDGEMENT -----	2
LIST OF FIGURES -----	6
LIST OF ABBREVIATIONS -----	7
ABSTRACT -----	8
CHAPTER 1: INTRODUCTION -----	9
1.1 About the Normalization Tool-----	9
1.2 Purpose of Normalization-----	10
1.3 Terms Related to Normalization-----	10
1.3.1 Universal Set-----	10
1.3.2 Relation-----	11
1.3.3 Attribute-----	11
1.3.4 Relational Database-----	11
1.3.5 Relational Schema-----	11
1.3.6 Functional Dependencies-----	11
CHAPTER 2: REQUIREMENTS ANALYSIS -----	13
2.1 Introduction-----	13
2.2 System Requirements-----	13
2.2.1 Minimum System Requirements-----	13
2.2.2 Recommended System Requirements-----	13
2.3 Hardware Requirements-----	14
2.3.1 Minimum Hardware Requirements-----	14
2.3.2 Recommended Hardware Requirements-----	14
2.4 Software Requirements-----	15
2.4.1 Platform-----	15
2.4.2 Interface-----	15
2.4.3 Databases-----	15
2.4.4 .NET Framework-----	17

CHAPTER 3: ANOMALIES -----	19
CHAPTER 4: NORMAL FORMS -----	21
4.1 First Normal Form-----	21
4.1.1 Objectives of 1NF-----	21
4.1.2 Problems-----	21
4.2 Full Functional Dependency-----	21
4.2.1 Definition-----	21
4.2.2 Example-----	22
4.3 Second Normal Form-----	22
4.3.1 Definition-----	22
4.3.2 Transforming a 1NF relation to 2NF relation-----	23
4.3.2.1 Example-----	23
4.3.3 Objectives of 2NF-----	24
4.3.4 Review-----	24
4.4 Third Normal Form-----	25
4.4.1 Definition-----	25
4.4.2 Example-----	25
4.4.3 Transforming a 2NF relation to 3NF relation-----	25
4.4.3.1 Example-----	26
4.4.4 Objectives of 3NF-----	26
4.5 Boyce Codd Normal Form-----	26
4.5.1 Definition-----	26
4.5.2 Example-----	27
4.5.3 Objectives of BCNF-----	27
4.6 Fourth Normal Form-----	28
4.6.1 Definition of MVD-----	28
4.6.2 Definition of 4NF-----	29
4.6.3 Objectives of 4NF-----	30
CHAPTER 5: THE NORMALIZATION PROCESS -----	31
5.1 Decomposition-----	31

5.2 Synthesis-----	31
5.2.1 Basic Steps-----	31
5.2.2 Problems-----	32
5.2.3 Examples-----	32
CHAPTER 6: SYNTHESIS-BOTTOM UP PROCESS-----	34
6.1 Synthesizing Third Normal Form Relations-----	34
6.2 Algorithm-----	35
6.2.1 Example-----	36
CHAPTER 7: NORMALIZATION TOOL DESIGN-----	38
7.1 Main Page or Welcome Screen-----	38
7.2 Synthesizing Normalized Relations-----	39
7.3 Normalizing 3NF Relations-----	40
7.4 Normalizing BCNF Relations-----	41
7.5 Normalizing 4NF Relations-----	42
7.6 Relations in 3NF-----	44
7.7 Relations in BCNF-----	45
7.8 Relations in 4NF-----	46
7.9 Insert Values in Database-----	47
7.10 View Database-----	48
7.11 Access Database Tables-----	49
CONCLUSION-----	50
REFERENCE-----	51

LIST OF FIGURES

Fig. 1 Main Page or Welcome Screen-----	38
Fig. 2 Synthesizing Normalized Relations-----	39
Fig. 3 Normalizing 3NF Relations-----	40
Fig. 4 Normalizing BCNF Relations-----	41
Fig. 5 Normalizing 4NF Relations-----	42
Fig. 6 Relations in 3NF-----	44
Fig. 7 Relations in BCNF-----	45
Fig. 8 Relations in 4NF-----	46
Fig. 9 Insert Values in Database-----	47
Fig. 10 View Database-----	48
Fig. 11 Access Database Tables-----	49

12. STU - Relation to student

13. FACID - Relation to faculty identity

14. SCHED - Relation to schedule

15. Nat - Nat Dept

16. FACNAME - Faculty Name

LIST OF ABBREVIATIONS

1. NF – Normal Form
2. MVD – Multi-valued Dependency
3. BCNF – Boyce Codd Normal Form
4. DBMS – Database Management Systems
5. RDBMS – Relational Database Management Systems
6. FD – Functional Dependency
7. ROM – Read Only Memory
8. RAM – Random Access Memory
9. GHz – Giga hertz
10. Mb – Megabytes
11. Gb – Gigabytes
12. STU – Refers to student
13. FACID – Refers to faculty identity
14. SCHED – Refers to schedule
15. .Net – Dot Net
16. FACNAME – Faculty Name

ABSTRACT

Our project *Synthesizing Normalized Relations* takes a practical approach towards implementing *normalization*. “**Normalization Tool**” is our software that implements normalization.

Normalization Tool is meant to synthesize normal form relations. It converts a universal set of attributes given with their respective functional dependencies into any of the given normal form:

1. Third Normal Form (3NF)
2. Boyce Codd Normal Form (BCNF)
3. Fourth Normal Form (4NF)

Normalization Tool can also create tables for the normalized set of relations and we can insert the values directly in the tables. We can, not only create tables but can update, insert, and delete also. We can view the database and entries in a given table also.

It has the reusability feature in it. The universal set, functional dependencies and normalization set of relations are stored in a text file. Whenever we want to change values and get new set of relations we can directly do it through the *Normalization Tool* and we can save it into a new file.

Through the text file we can directly load all the required information into the *Normalization Tool* and can convert to higher normal form by adding some extra information. For example: we can convert third normal form relations to fourth normal relations by adding information about multi-valued dependencies to it.

CHAPTER 1

INTRODUCTION

Databases and database technology are having a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, engineering, medicine, law, education and library science, to name a few.

A *database* is a collection of related data and the *Database Management System* (DBMS) is the software that manages and controls access to the database. A *database application* is simply a program that interacts with the database at some point in its execution.

When we design a database for a relational system, the main objective is to create an accurate representation of the data, its relationships, and constraints. To achieve this objective, we must identify a suitable set of relations. A technique that we can use to identify such relations is called **normalization**.

1.1 About the Normalization Tool

Normalization is a bottom-up approach to database design that begins by examining the relationships between attributes. It's a technique for producing a set of relations with desirable properties, given the data requirements of an enterprise. Our *Normalization Tool* normalizes the given universal set of attributes with the help of given functional dependencies between the attributes.

It makes the process of normalization easy for the user, as the user can get desired normal form relations by simply entering only the basic information about the data i.e. attributes and FDs. It can also insert, delete, and update the relations into database tables.

1.2 Purpose of Normalization

Normalization is often performed as a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form.

We describe a relation as consisting of a number of attributes, and a relational schema as consisting of a number of relations. Attributes may be grouped together to form a relational schema based largely on the common sense of the database designer, or by mapping the relational schema from an ER model. Whatever the approach taken, a formal method is often required to help the database designer identify the optimal grouping of attributes for each relation in the schema.

The process of normalization is a formal method that identifies relations based on their primary or candidate keys and the functional dependencies among their attributes. Normalization supports database designers by presenting a series of tests, which can be applied to individual relations so that a relational schema can be normalized to a specific form to prevent the possible occurrence of update anomalies.

1.3 Terms Related to Normalization

There are certain terms related to normalization in *Relational Database Management System* (RDBMS), so are they related to our project. Before going into the details of our project we must know about these terms in relation to database.

1.3.1 Universal Set

A relational schema has attributes (A, B, C, \dots, Z) and the whole database is described by a single universal relation called $R = (A, B, C, \dots, Z)$. This assumption means that every attribute in the database has a unique name. R is also called as the universal set.

1.3.2 Relation

A relation is a set of values. The relation has a name that is distinct from all other relation names in the relational schema. In terms of RDBMS a relation is a table with columns and rows. For example: Relation Class = (STUID, COURSE#, STUNAME, FACID, SCHED, ROOM, GRADE).

1.3.3 Attribute

An attribute is a named column of a relation. In the relational model, relations are used to hold information about the objects to be represented in the database. A relation is represented as a two-dimensional table in which the rows of the table correspond to individual records and the table columns correspond to attributes. For example: in the above given relation STUID, COURSE#, STUNAME, FACID, SCHED, ROOM, GRADE are the attributes.

1.3.4 Relational Database

A collection of normalized relations with distinct relation names is called as a relational database.

1.3.5 Relational Schema

It's a named relation defined by a set of attribute and domain name pairs whereas a domain is the set of allowable values for one or more attributes.

1.3.6 Functional Dependencies (FDs)

It describes the relationship between attributes in a relation. For example:

Let's consider the following relation

CLASS (STUID, COURSE#, STUNAME, FACID, SCHED, ROOM, GRADE)

Then the FDs are:

STUID, COURSE# → STUNAME, FACID, SCHED, ROOM, GRADE

COURSE# → FACEID, SCHED, ROOM

STUID → STUNAME

REQUIREMENTS ANALYSIS

1.1 Introduction

Requirements analysis of a system is a process that goes into determining the requirements of a new or altered system. Requirements analysis is also known as requirements engineering. It is sometimes referred to loosely by names such as requirements gathering, requirements capture, or requirements specification. Requirements must be measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

2.2 System Requirements

To design and implement our *normalization Tool* we had some system requirements. To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer system. These pre-requisites are known as system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

2.2.1 Minimum System Requirements

This set of requirements must be satisfied for the software to be usable at all. Computers with lower specifications than the minimum requirements may sometimes also run the software. It is suggested, however, that the user will not have a representative experience.

CHAPTER 2

REQUIREMENTS ANALYSIS

2.1 Introduction

Requirements analysis encompasses those tasks that go into determining the requirements of a new or altered system. Systematic requirements analysis is also known as *requirements engineering*. It is sometimes referred to loosely by names such as *requirements gathering*, *requirements capture*, or *requirements specification*. Requirements must be measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

2.2 System Requirements

To design and implement our *Normalization Tool* we had some system requirements. To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer system. These pre-requisites are known as **system requirements** and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

2.2.1 Minimum System Requirements

This set of requirements must be satisfied for the software to be usable at all. Computers with lower specifications than the minimum requirements may sometimes also run the software. It is suggested, however, that the user will not have a representative experience

of the software this way. Generally this set is regarded more of a rule than a guideline. A system meeting this requirement will provide basic performance of a software application.

2.2.2 Recommended System Requirements

This set of requirements is often suggested by software vendors for optimal performance of a software. Although not a necessity, this set of requirements is often sought after by power users who expect to gain a better experience of software usability. Recommended System Requirements do not promise best possible performance of a software and are treated as more of a guideline than a rule. Almost always a better system is available, or will be in future, to provide better performance.

2.3 Hardware Requirements

Hardware requirements for implementation of our project are as follows:

2.3.1 Minimum Hardware Requirements

Minimum hardware requirements for our project are:

1. Intel system architecture
2. Intel Pentium 4 Processor-1.6 GHz
3. 256 MB RAM (Random Access Memory)
4. Hard Disk Space- 4GB
5. CD-ROM Drive-24x

2.3.2 Recommended Hardware Requirements

Recommended hardware requirements for our project are:

1. Intel system architecture
2. Intel Pentium 4 Processor-2.0 GHz

3. 512 MB RAM (Random Access Memory)
4. Hard Disk Space- 6 GB
5. CD-ROM Drive-52x

2.4 Software Requirements

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of the application. These requirements or pre-requisites are, almost always, not included in the software installation package and need to be installed separately before the software is installed.

2.4.1 Platform

In computing, a **platform** describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming language and their runtime libraries.

Operating system is one of the first requirements mentioned while defining System requirements (software). Much software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. Most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although converse is not always true.

As our software *Normalization Tool* is a **windows application**, it runs only on **Windows Platform**. Microsoft Windows'XP is the recommended platform.

2.4.2 Interface

In order to give a presentable interface to our software, we analyzed various packages and finally decided upon *VISUAL BASIC.NET*.

2.4.2.1 Visual Basic.Net

Visual Basic .NET (VB.NET) is an object-oriented computer language that can be viewed as an evolution of Microsoft's Visual Basic (VB) implemented on the Microsoft .NET framework. Like all .NET languages, programs written in VB.NET require the .NET framework to execute. We have used **Visual Basic.Net 2003** version for our project.

Visual Basic .NET 2003 was released with version 1.1 of the .NET Framework. New features included support for the .NET Compact Framework and a better VB upgrade wizard. Improvements were also made to the performance and reliability of the .NET IDE (particularly the background compiler) and runtime. In addition, Visual Basic .NET 2003 was also available in the *Visual Studio .NET 2003 Academic Edition* (VS03AE).

2.4.3 Databases

Many options were available to us regarding the databases or backend that we could use in the development of the project like Microsoft SQL Server, Oracle, MS access, DB2 etc. We short listed a few and decided to use Microsoft SQL Server.

2.4.3.1 SQL Server

Microsoft SQL Server is a relational database management system (RDBMS may be a DBMS in which data is stored in the form of tables and the relationship among the data is also stored in the form of tables). The primary query language is Transact-SQL, an implementation of the ANSI/ISO standard Structured Query Language (SQL) used by both Microsoft and Sybase. Microsoft SQL Server and Sybase/ASE both communicate over networks using an application-level protocol called Tabular Data Stream (TDS). The TDS protocol has also been implemented by the Free TDS project in order to allow more kinds of client applications to communicate with Microsoft SQL Server and Sybase

databases. Microsoft SQL Server also supports Open Database Connectivity (ODBC). We have used **SQL server 2003** as the back-end for our project.

SQL Server 2003 also supports the ability to deliver client connectivity via the Web Services SOAP protocol. This allows non-Windows Clients to communicate cross platform with SQL Server. Microsoft SQL Server 2003 also features automated database mirroring, failover clustering, and database snapshots.

Microsoft and other vendors provide a number of software development tools designed to allow business applications to be developed using the data stored by Microsoft SQL Server. Microsoft SQL Server 2003 now includes the common language runtime (CLR) component for Microsoft .NET. Applications developed with .NET languages such as Visual Basic can implement stored procedures and other functions. Older versions of Microsoft development tools typically use APIs to access Microsoft SQL Server functionality. Rapid application development tools incorporate native database gateways for high speed database access and automatic table drill-down for the creation of quick prototype applications for viewing, editing and adding data to any table in the database.

2.4.4 .NET Framework

The **Microsoft .NET Framework** is a software component that can be added to or is included with the Microsoft Windows operating system. It provides a large body of pre-coded solutions to common program requirements, and manages the execution of programs written specifically for the framework. The .NET Framework is a key Microsoft offering, and is intended to be used by most new applications created for the Windows platform.

The pre-coded solutions that form the framework's class library cover a large range of programming needs in areas including: user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The functions of the class library are used by programmers who combine them with their own code to produce applications.

Programs written for the .NET Framework execute in a software environment that manages the program's runtime requirements. This runtime environment, which is also a part of the .NET Framework, is known as the Common Language Runtime (CLR). The CLR provides the appearance of an application virtual machine, so that programmers need not consider the capabilities of the specific CPU that will execute the program. The CLR also provides other important services such as security mechanisms, memory management, and exception handling. The class library and the CLR together compose the .NET Framework. The framework is intended to make it easier to develop computer applications and to reduce the vulnerability of applications and computers to security threats.

.Net Framework is required for our software, as it allows our software to run on any system having a SQL Server 2003 or higher. No other software is required once the .exe file is created of the *Normalization Tool* from VB.Net Package except the SQL Server. For the software, recommended hardware requirements are same as for the designing phase of the software.

CHAPTER 3

ANOMALIES

Database normalization is a design technique by which relational database tables are structured in such a way as to make them less vulnerable to certain types of logical inconsistencies and *anomalies*. Tables can be normalized to varying degrees: relational database theory defines "normal forms" of successively higher degrees of stringency, so, for example, a table in third normal form is less open to logical inconsistencies and anomalies than a table that is only in second normal form. Although the normal forms are often defined (informally) in terms of the characteristics of tables, rigorous definitions of the normal forms are concerned with the characteristics of mathematical constructs known as relations.

A table that is not sufficiently normalized can suffer from logical inconsistencies of various types, and from anomalies involving data operations. In such a table:

- The same fact can be expressed on multiple records; therefore updates to the table may result in logical inconsistencies. For example, each record in an unnormalized "CLASS" table might contain a STUID, COURSE#, STUNAME, ROOM, and GRADE; thus a change of a ROOM for a particular student will potentially need to be applied to multiple records. If the update is not carried through successfully—if, that is, the student's ROOM is updated on some records but not others—then the table is left in an inconsistent state. Specifically, the table provides conflicting answers to the question of what this particular student's ROOM is. This phenomenon is known as an **update anomaly**.
- There are circumstances in which certain facts cannot be recorded at all. In the above example, if it is the case that student ROOM is held only in the "CLASS" table, then we cannot record the ROOM of a student who has not yet registered for any of the course. This phenomenon is known as an **insertion anomaly**.

- There are circumstances in which the deletion of data representing certain facts necessitates the deletion of data representing completely different facts. For example, suppose a table has the attributes Student ID, Course ID, and Lecturer ID (a given student is enrolled in a given course, which is taught by a given lecturer). If the number of students enrolled in the course temporarily drops to zero, the last of the records referencing that course must be deleted—meaning, as a side-effect, that the table no longer tells us which lecturer has been assigned to teach the course. This phenomenon is known as a **deletion anomaly**.

Ideally, a relational database table should be designed in such a way as to exclude the possibility of update, insertion, and deletion anomalies. The normal forms of relational database theory provide guidelines for deciding whether a particular design will be vulnerable to such anomalies. It is possible to correct an unnormalized design so as to make it adhere to the demands of the normal forms: this is normalization.

CHAPTER 4

NORMAL FORMS

4.1 First Normal Form (1NF)

4.1.1 Objectives of 1NF

- The schema of an unorganized relation gives no clues to which attributes can have multiple values.
- Semantics of a 1NF are more explicit.
- The relational operators are applicable only on flat that is 1NF relations.

4.1.2 Problems

- Update Anomalies
- Deletion Anomalies
- Insertion Anomalies



4.2 Full Functional Dependency

4.2.1 Definition

In a relation R attribute B of R is “*fully functional dependent*” on an attribute or set of attribute A of R if B is functional dependent on A but not functional dependent on any proper subset of A.

4.2.2 Example

Let's consider the following relation:

CLASS (STUID, COURSE#, STUNAME, FACID, SCHED, ROOM, GRADE)

FDs:

STUID, COURSE# \rightarrow STUNAME, FACID, SCHED, ROOM, GRADE

COURSE# \rightarrow FACEID, SCHED, ROOM

STUID \rightarrow STUNAME

4.3 Second Normal Form (2NF)

4.3.1 Definition

A relation is in second normal form (2NF) if and only if it is in 1NF and all nonkey attributes are fully dependent on the key.

Clearly if the relation is in 1NF and the key consists of a single attribute the relation is automatically 2NF.

For example the previous relation is not in 2NF.

CLASS (STUID, COURSE#, STUNAME, FACID, SCHED, ROOM, GRADE)

STUID, COURSE# \rightarrow STUNAME, FACID, SCHED, ROOM, GRADE

COURSE# \rightarrow FACEID, SCHED, ROOM

STUID \rightarrow STUNAME

The CLASS relation is not in 2NF.

- The STUNAME is not fully dependent on {COURSEID#, STUID}.
- The {FACEID, SCHED, ROOM} are not fully dependent on {COURSE#, STUID}.

- Intuitively a relation might not be in 2NF if is trying to describe information for more than one entity i.e. through a many-many relationship.
- In the previous example: student and the course entity.

4.3.2 Transforming a 1NF relation to 2NF relation

1. Identify each nonfull functional dependency.
2. Form projections by removing the attributes that depend on each of the determinants so identified.
3. Place these determinants in separate relations along with their dependent attributes.
4. The original relation still contains the composite key and any attributes that are fully functional dependent on it.

These types of projections are called “lossless projection” because the original relation can be reconstructed by taking the natural join of the resulting projections.

4.3.2.1 Example

CLASS (STUID, COURSE#, STUNAME, FACID, SCHED, ROOM, GRADE)

FDs:

STUID, COURSE# → STUNAME, FACID, SCHED, ROOM, GRADE

COURSE# → FACEID, SCHED, ROOM

STUID → STUNAME

Changing the FDs to 2NF

CLASS2 (STUID, COURSEID#, GRADE)

COURSE (COURSE#, FACEID, SCHED, ROOM)

STU (STUID, STUNAME)

4.3.3 Objectives of 2NF

1. The semantics of a 2NF are more explicit: all the attributes are dependent on the entire primary key.

2. Database designed with 2NF relations avoid undesirable update anomalies present in 1NF relations.

3. The schema of a 1NF relation gives no clue to which attributes are dependent on which other attributes.

Knowing that a relation in 2NF means that no attribute is dependent on only part of the key.

4.3.4 Review

- Transitive Dependency (3rd Armstrong's axiom)

Let's consider the following relation

STUDENT (STUID, STUNAME, MAJOR, CREDITS, STATUS)

FDs:

STUID \rightarrow All the other attributes

So,

STUID \rightarrow CREDITS

But also,

CREDITS \rightarrow STATUS

- The STUID functionally determines STATUS in two ways: Directly and Transitively through CREDITS.

- So the attribute STATUS is said to be transitively dependent on the attribute STUID.

4.4 Third Normal Form (3NF)

4.4.1 Definition

A relation is in third normal form (3NF) if and only if

- It is in 2NF and
- No nonkey attribute is “transitively dependent” on the key.

4.4.2 Example

The following relation is in 2NF but not in 3NF.

STUDENT (STUID, STUNAME, MAJOR, CREDITS, STATUS)

- Because the nonkey attribute STATUS is transitively dependent on the key, STUID.

Clearly a 2NF relation with one nonkey attribute must always be a 3NF relation.

4.4.3 Transforming a 2NF relation to 3NF relation

1. We look to see if any nonkey attribute is functionally dependent on another nonkey attribute.
2. Remove the functionally dependent attribute from the relation placing it in a new relation with its determinant.
3. The determinant can remain in the original relation.

4.4.3.1 Example

STUDENT (STUID, STUNAME, MAJOR, CREDITS, STATUS)

Changing the FDs to 3NF

STU2 (STUID, STUNAME, MAJOR, CREDITS)

STATS (CREDITS, STATUS)

4.4.4 Objectives of 3NF

1. The semantics of a 3NF are more explicit: all the attributes are dependent ONLY on the primary key.
2. Database designed with 3NF relations avoid undesirable update anomalies present in 2NF relations.
3. The schema of a 2NF relation gives no glue to which nonkey attributes are dependent on which other nonkey attributes.
4. Knowing that a relation in 3NF means that no nonkey attribute is dependent on only part of the key.

4.5 Boyce Codd Normal Form (BCNF)

4.5.1 Definition

A relation is in Boyce-Codd normal form (BCNF) if and only if every determinant is a candidate key.

4.5.2 Example

FACULTY (FACNAME, DEPT, OFFICE, RANK, DATEHIRED)

Constraints

1. No two faculty members within a single department have the same name.
2. Each faculty member has only one office.
3. A department may have several faculty offices.
4. Faculty members from the same department may share offices.

Resulting FDs

OFFICE \rightarrow DEPT

FACNAME, DEPT \rightarrow OFFICE, RANK, DATEHIRED

FACNAME, OFFICE \rightarrow DEPT, RANK, DATEHIRED

The relation is not in BCNF because OFFICE is not a candidate key.

Relation schema in BCNF

FAC1 (OFFICE, DEPT)

FAC2 (FACNAME, OFFICE, RANK, DATEHIRED)

4.5.3 Objectives of BCNF

1. The semantics of multiple candidate keys are more explicit: all the attributes are dependent ONLY on the candidate key.

2. Database designed with BCNF relations avoid undesirable update anomalies present in 3NF relations.

3. In previous example

We can not delete a faculty member from a department without losing information about an office (assuming he is the only occupant). That is because OFFICE is not a candidate key.

4.6 Fourth Normal Form (4NF)

4.6.1 Definition of Multivalued dependency (MVD)

Given a relation R with attributes A, B, and C, the *multivalued dependency (MVD)*

$$R.A \twoheadrightarrow R.B$$

holds in R if and only if the set of B-values matching a given (A-value, C-value) pair in R depends only on A-value.

i.e. if $(a, b_1, c_1) \in R, (a, b_2, c_2) \in R$

then $(a, b_1, c_2) \in R, (a, b_2, c_1) \in R$

Another way to view MVD

Definition

Let $R(X, Y, Z)$ be a relation and X, Y, Z be pairwise disjoint.

Let $Yxz = \{ y \mid (x, y, z) \in R \}$

The MVD $X \twoheadrightarrow Y$ is said to hold for $R(X, Y, Z)$ if and only if Yxz depends on X i.e. $Yxz = Yxz'$ for all x, z, z' values of attributes X and Z , such that Yxz and Yxz' are non-empty.

We sometime use

$$X \twoheadrightarrow Y \mid Z$$

The two definitions for MVD are equivalent.

For the relation CTX, we have

$$\text{Course} \twoheadrightarrow \text{Teacher}$$

$$\text{Course} \twoheadrightarrow \text{Text}$$

$$\text{i.e. Course} \twoheadrightarrow \text{Teacher} \mid \text{Text}$$

Note

$X \twoheadrightarrow \emptyset$ and $X \twoheadrightarrow Y$ hold for $R(X, Y)$.

$X \twoheadrightarrow Y$ whenever $Y \subseteq X \subseteq R$ for R , there we use R to represent all attributes of relation R also.

These are called *trivial multivalued dependencies*.

(Note: A functional dependency $X \rightarrow Y$ is said to be trivial if $Y \subseteq X$)

4.6.2 Definition of 4NF

A relation R is in **fourth normal form (4NF)** if and only if any nontrivial MVD $X \twoheadrightarrow Y$ holds in R implies X is a superkey of R . i.e. $X \rightarrow a$ for all attribute a of R .

(Note: A relation R is in BCNF if and only if any nontrivial FD $X \rightarrow Y$ holds in R implies $X \rightarrow a$ for all attribute a of R .)

4.6.3 Objectives of 4NF

1. The semantics are more explicit.
2. All dependencies are related.
3. Database designed with 4NF relations avoid undesirable update anomalies present in 3NF.

1. Start with a universal relation

2. Identify functional dependencies

3. Use decomposition techniques to split the universal relation into a set of ones.

The previous example was based on the decomposition approach.

5.2 Synthesis (Bottom-Up Process)

Begin with attributes and combine them into related groups using functional dependencies to develop a set of normalized relations.

A synthesis algorithm was developed by *Peter D. Bernstein*.

5.2.1 Basic Steps

1. Make a list of all FDs

2. Group together those with the same determinant

CHAPTER 5

THE NORMALIZATION PROCESS

The process of finding stable set of relations that is a faithful model of the enterprise.

5.1 Decomposition (Top-Down Process)

1. Start with a universal relation
2. Identify functional dependencies
3. Use decomposition techniques to split the universal relation into a set of ones.

The previous example was based on the decomposition approach.

5.2 Synthesis (Bottom-Up Process)

Begin with attributes and combine them into related group using functional dependencies to develop a set of normalized relations.

A synthesis algorithm was developed by *Philip A. Bernstein*.

5.2.1 Basic Steps

1. Make a list of all FDs.
2. Group together those with the same determinant

3. Construct a relation of each group.

5.2.2 Problems

1. Some FDs have more attributes in the determinant than needed.

We must eliminate extraneous attributes or 2NF relations might not result.

2. Eliminate redundant FDs before grouping 3NF will not result.

3. Two relations may appear to have different keys when in fact the keys are equivalent.

5.2.3 Examples

Case 1

FDs

f1: $ABF \rightarrow G$ R1 (A, B, F, G) R1 not in 2NF

f2: $A \rightarrow F$ R2 (A, F)

Extraneous attribute is f1.

Case 2

FDs

f1: $X \rightarrow Y$ R1 (X, Y, Z)

f2: $X \rightarrow Z$ R2 (Y, Z) R1 in 2NF but not in 3NF.

f3: $Y \rightarrow Z$

Redundant FD; f2 can be derived from f1 and f3.

Case 3

FDs

f1: $X \rightarrow A$ R1 (X, A, Y)
f2: $Y \rightarrow B$ R2 (Y, B, X) Too many relations.
f3: $X \leftrightarrow Y$

Case 3A

FDs

f1: $X \rightarrow A$ R1 (X, A)
f2: $A \rightarrow Y$ R2 (A, Y)
f3: $Y \rightarrow X$ R3 (Y, X) Too many relations.

CHAPTER 6

SYNTHESIS-BOTTOM UP PROCESS

6.1 Synthesizing Third Normal Form Relations (by Philip A. Bernstein)

A Description of the Algorithm:

The simple synthesis procedure led to problems because the rules for composing FDs were ignored. The main difficulty is that the redundant FDs that filter into the synthesized schema creates extra attributes and contributes to unnormalized connections among attributes. By first taking a nonredundant covering of the given set of FDs, the normalization problems can be alleviated. In the example given in the previous chapter F2 is redundant and therefore will not appear in a nonredundant covering of the given FDs, and the 3NF violation of R1 is thereby avoided.

Finding a nonredundant covering is not sufficient to avoid problem FDs such as F6. This further problem can be eliminated by excising extraneous attributes from the left sides of FDs. An attribute X_i is extraneous in an FD $g \in G$, $g: X_1, \dots, X_i \rightarrow Y$, if $X_1, \dots, X_{i-1}, X_{i+1}, \dots, X \rightarrow Y$ is in G^+ . Eliminating extraneous attributes helps to avoid partial dependencies and superkeys that are not keys, as in R4.

If two relations have keys that are functionally dependent upon each other (i.e. are equivalent), then the two relations can be merged together. This can be accomplished in the synthesis procedure by merging together two groups of FDs if their left sides are functionally equivalent, For example g_2 and g_3 can be merged into a single group.

6.2 Algorithm

I. Eliminate extraneous attributes

Let F be the given set of FD's where the right side of each FD is a single attribute.

Eliminate extraneous attributes from the left side of each FD in F , producing the set G .

II. Finding covering

Find a non-redundant covering H of G .

III. Partition

Partitions H into groups such that all of the FD's in each group have identical left sides.

IV. Merge equivalent keys

Let $J = \Phi$. For each pair of groups, say H_i and H_j with left sides X and Y respectively. If X and Y are properly equivalent, then

- Merge H_i and H_j together
- Add $X \rightarrow Y$ and $Y \rightarrow X$ to J
- If $X \rightarrow Z \in H$ and $Z \in Y$, then delete $X \rightarrow Z$ from H . Similarly, if $Y \rightarrow Z \in H$ and $Z \in X$, and then delete $Y \rightarrow Z$ from H .

V. Eliminate transitive dependencies

Find a minimal H' which is subset of H such that

$$(H' \vee J)^+ = (H \vee J)^+$$

Then add each FD of J into its corresponding group of H' .

VI. Construct relations

- Each group forms a relation.
- Each set of attributes that appears on the left side of any FD in the group is a key of relation. They are explicit keys.

Result1: The relations produced by step 6 are all in 3NF.

Result2: The number of relations produced is minimum.

6.2.1 Example

Given $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, B \rightarrow D, D \rightarrow B, ABE \rightarrow F\}$

Step I Eliminating extraneous attributes

$G = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, B \rightarrow D, D \rightarrow B, AE \rightarrow F\}$

(since $AE \rightarrow ABE \in F^+$)

Step II Find covering

$H = \{A \rightarrow B, B \rightarrow C, B \rightarrow D, D \rightarrow B, AE \rightarrow F\}$

(since $A \rightarrow C \in (G - \{A \rightarrow C\})^+$)

Step III Partition

$H_1 = \{A \rightarrow B\}$

$H_2 = \{B \rightarrow C, B \rightarrow D\}$

$H_3 = \{D \rightarrow B\}$

$H_4 = \{AE \rightarrow F\}$

Step IV Merge groups

B and D are properly equivalent

$$J = \{B \rightarrow D, D \rightarrow B\}$$

$$H1 = \{A \rightarrow B\}$$

$$H'2 = H2 \cup H3 - \{B \rightarrow D, D \rightarrow B\}$$

$$= \{B \rightarrow C\}$$

$$H4 = \{AE \rightarrow F\}$$

Step V Eliminate transitive dependencies

None!

Step VI Construct relations

$$R1 (\underline{A}, B)$$

$$R2 (\underline{B}, \underline{D}, C)$$

$$R3 (\underline{A}, \underline{E}, F)$$

CHAPTER 7

NORMALIZATION TOOL DESIGN

The Normalization tool was designed and working software was created. Front-end was designed with the help of VB.NET.

7.1 Main Page or Welcome Screen

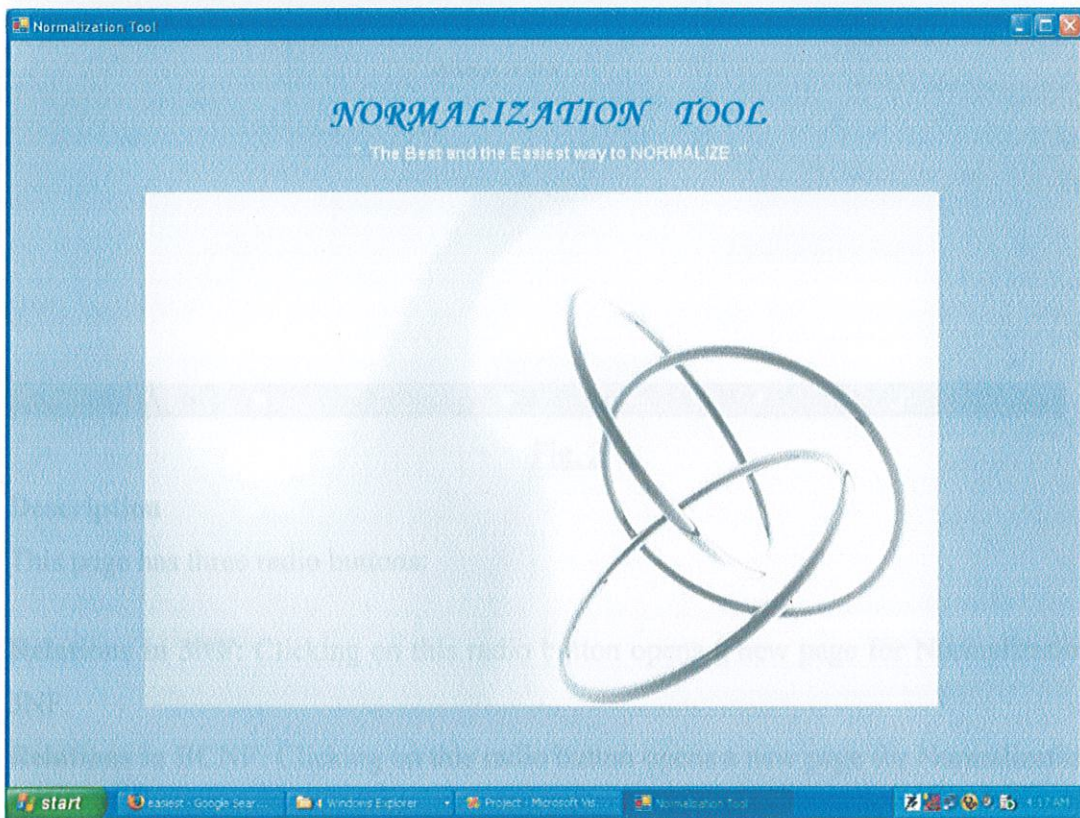


Fig. 1

Description

This screen is our welcome screen which is displayed while the loading is on. After the loading the next screen is the synthesizing normalized relations screen. It provides user with the option of normalizing into any of the given normal forms.

7.2 Synthesizing Normalized Relations

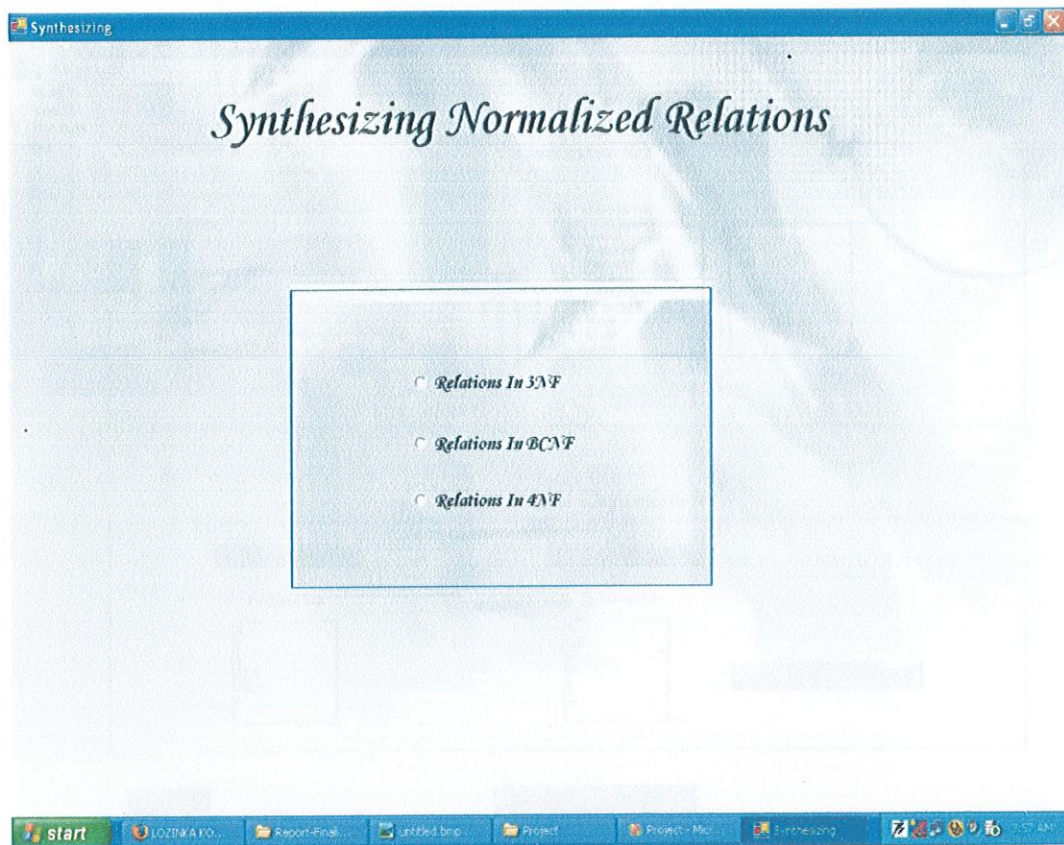


Fig. 2

Description

This page has three radio buttons:

Relations in 3NF: Clicking on this radio button opens a new page for Normalization in 3NF.

Relations in BCNF: Clicking on this radio button opens a new page for Normalization in BCNF.

Relations in 4NF: Clicking on this radio button opens a new page for Normalization in 4NF.

7.3 Normalizing 3NF Relations

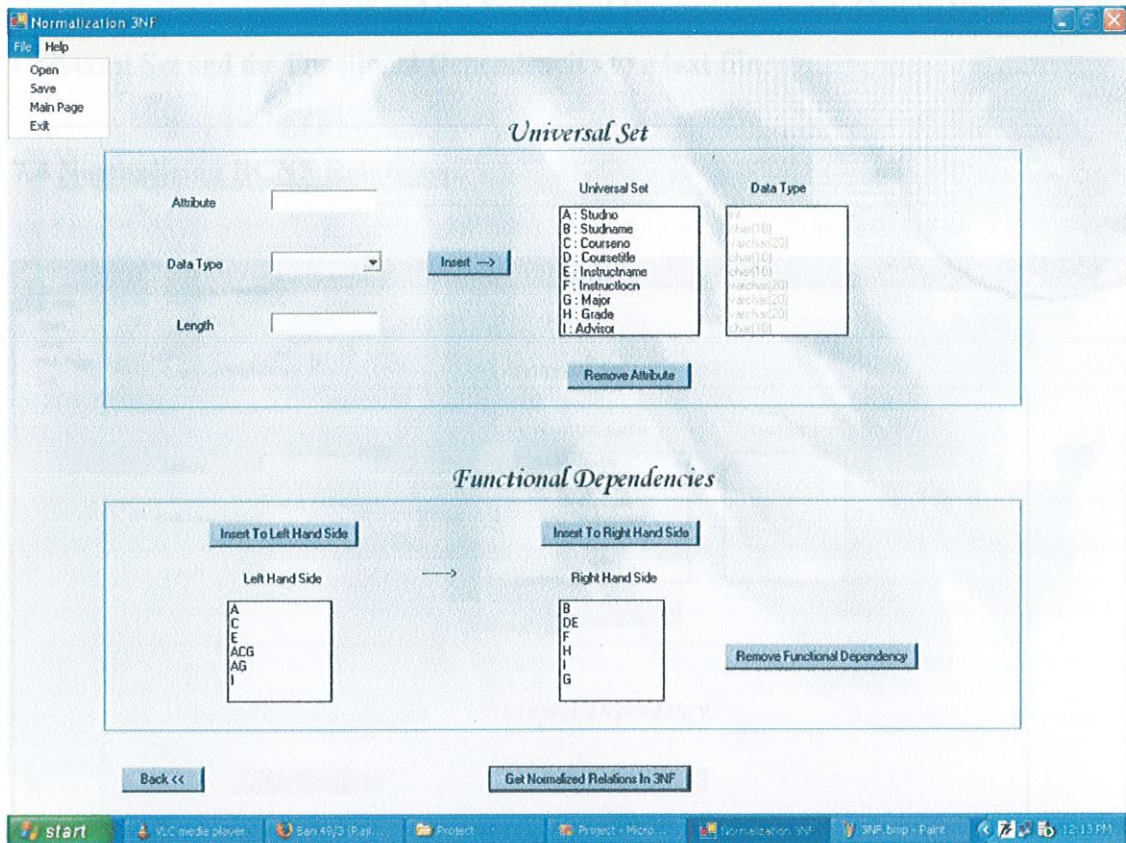


Fig. 3

Description

In this page attributes are inserted in the universal set listbox through the attribute textbox. The entries in the Universal Set are of this form - **A: Employee Name**. In addition to attribute their data type and length are also inserted as they would be required in case the user desires to create the normalized tables. The **Remove Attribute** button removes the selected attribute from the universal set and the functional dependency which has the selected attribute in it. The **Insert To Left Hand Side** button inserts the selected attribute (in universal set listbox) to the left hand side listbox of the Functional Dependencies. Same is for right hand side. **Remove Functional Dependency** button removes the selected functional dependency from the set of functional dependencies. Pressing the **Get Normalized Relations** opens a new page Relations In 3NF. The **Back** button is to go back to the Synthesizing Normalized Relations Page.

In the File Menu we have open and save options. Option **Open** opens a text file which already has the Universal Set and the Functional Dependencies set. Option **Save** saves the Universal Set and the Functional Dependencies to a text file.

7.4 Normalizing BCNF Relations

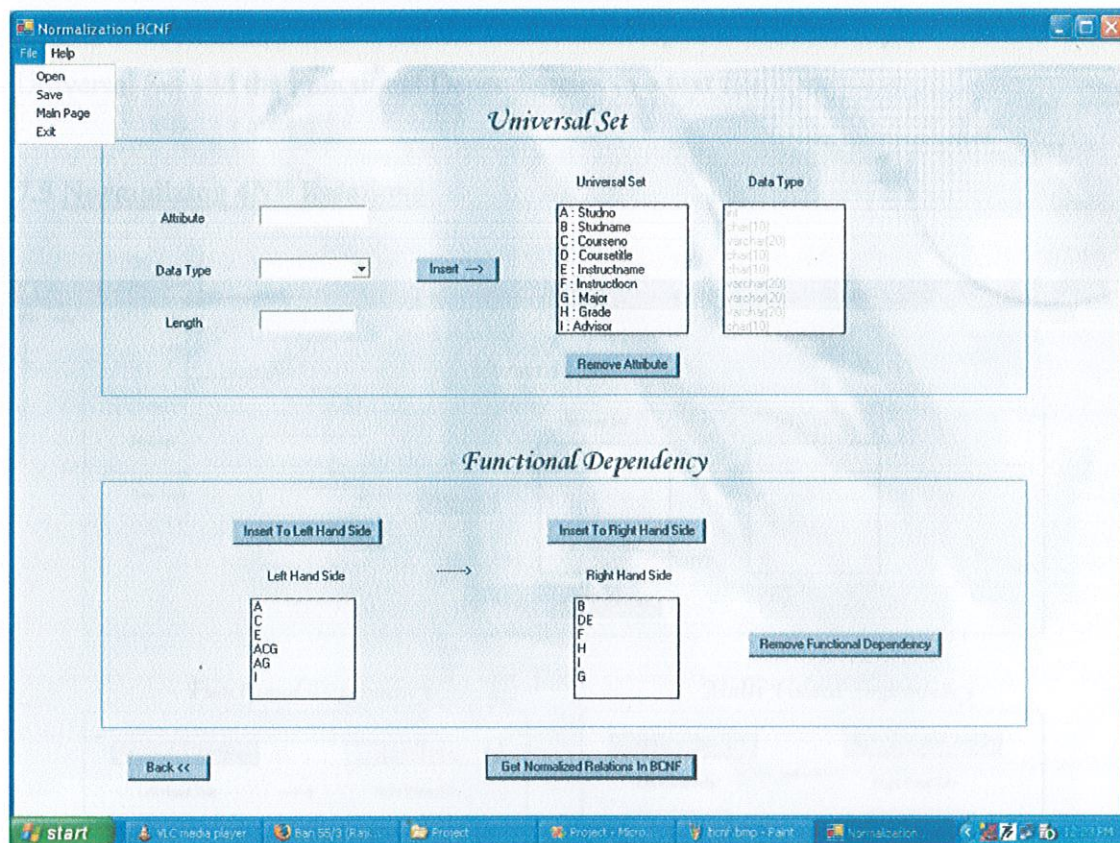


Fig. 4

Description

In this page attributes are inserted in the universal set listbox through the attribute textbox. The entries in the Universal Set are of this form - A : Employee Name. In addition to attribute their data type and length are also inserted as they would be required in case the user desires to create the normalized tables. The **Remove Attribute** button removes the selected attribute from the universal set and the functional dependency which has the selected attribute in it. The **Insert To Left Hand Side** button inserts the

selected attribute (in universal set listbox) to the left hand side listbox of the Functional Dependencies. Same is for right hand side. **Remove Functional Dependency** button removes the selected functional dependency from the set of functional dependencies. Pressing the **Get Normalized Relations** opens a new page Relations In BCNF. The **Back** button is to go back to the Synthesizing Normalized Relations Page.

In the File Menu we have open and save options. Option **Open** opens a text file which already has the Universal Set and the Functional Dependencies set. Option **Save** saves the Universal Set and the Functional Dependencies to a text file.

7.5 Normalizing 4NF Relations

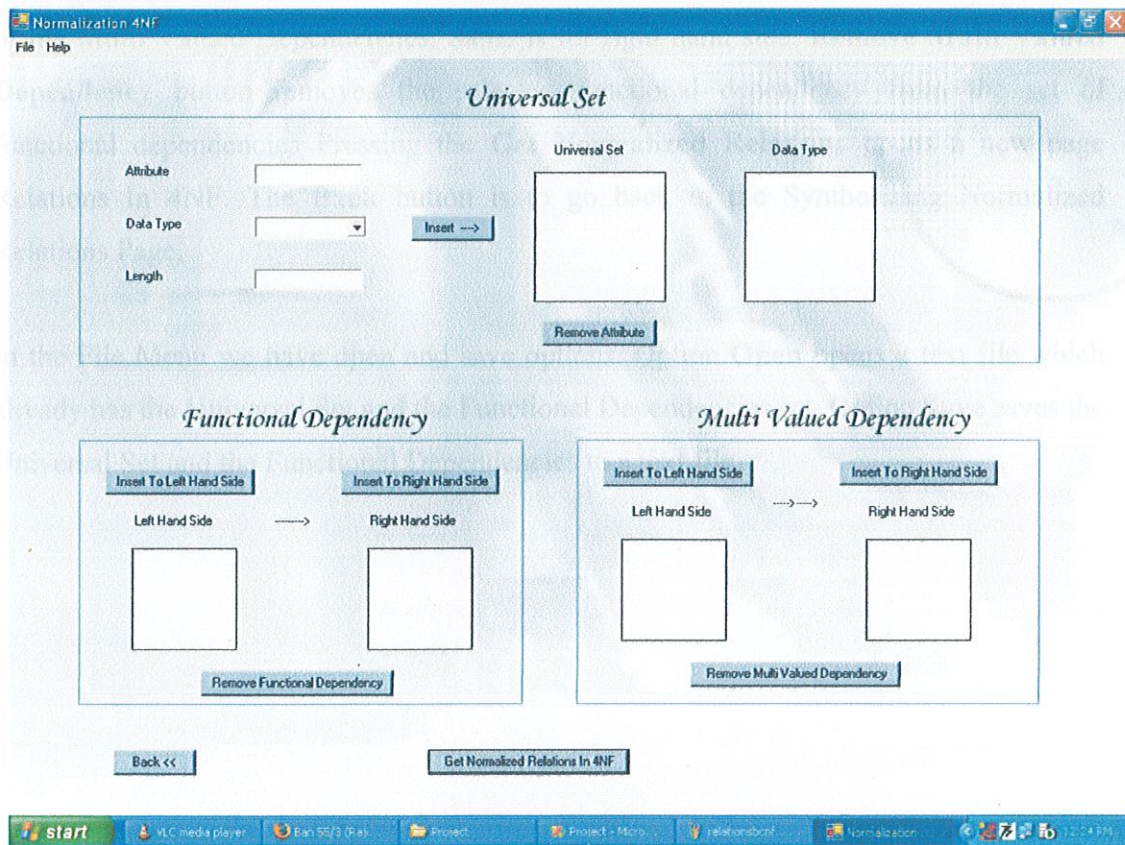


Fig. 5

Description

In this page attributes are inserted in the universal set listbox through the attribute textbox. The entries in the Universal Set are of this form - **A : Employee Name**. In addition to attribute their data type and length are also inserted as they would be required in case the user desires to create the normalized tables. The **Remove Attribute** button removes the selected attribute from the universal set and the functional dependency which has the selected attribute in it. In Functional Dependency the **Insert To Left Hand Side** button inserts the selected attribute (in universal set listbox) to the left hand side listbox of the Functional Dependencies. Same is for right hand side. **Remove Functional Dependency** button removes the selected functional dependency from the set of functional dependencies. . In Multi Valued Dependencies the **Insert To Left Hand Side** button inserts the selected attribute (in universal set listbox) to the left hand side listbox of the Multi Valued Dependencies. Same is for right hand side. **Remove Multi Valued Dependency** button removes the selected functional dependency from the set of functional dependencies Pressing the **Get Normalized Relations** opens a new page Relations In 4NF. The **Back** button is to go back to the Synthesizing Normalized Relations Page.

In the File Menu we have open and save options. Option **Open** opens a text file which already has the Universal Set and the Functional Dependencies set. Option **Save** saves the Universal Set and the Functional Dependencies to a text file.

7.6 Relations in 3NF

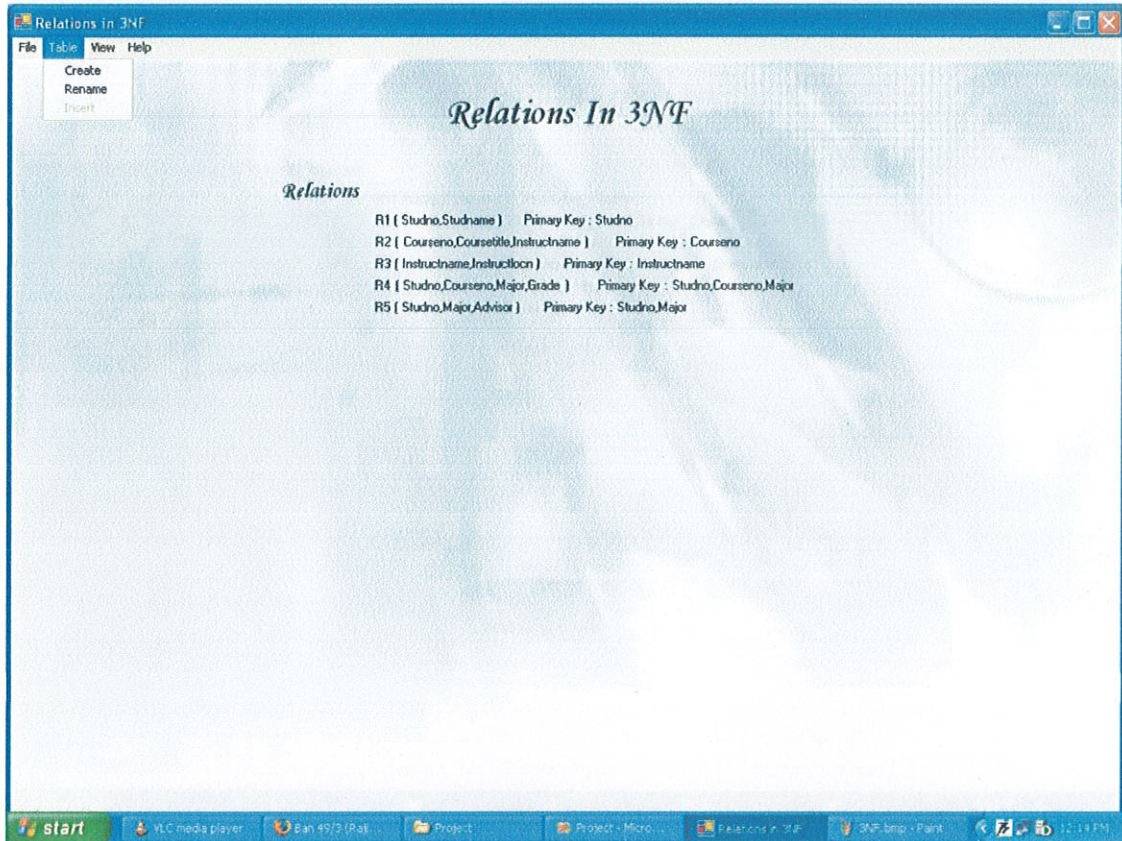


Fig. 6

Description

This page is the output page. On this Page we get the relations and the primary key corresponding to the related Relation. The Relations are in 3NF.

In the File Menu we have the option to **Save** the output (Relations and the Primary Keys) to a notepad file. We can also go back to the Main Page i.e. Synthesizing Normal Form Page or exit from the software.

In the Table Menu we have options of creating, renaming the tables and entering the data into the tables. Option **Create** creates the tables in the database. Option **Rename** renames the table in the database as well as on the output page. Option **Insert** opens the Insert Page from where we enter the data into the tables.

7.7 Relations In BCNF

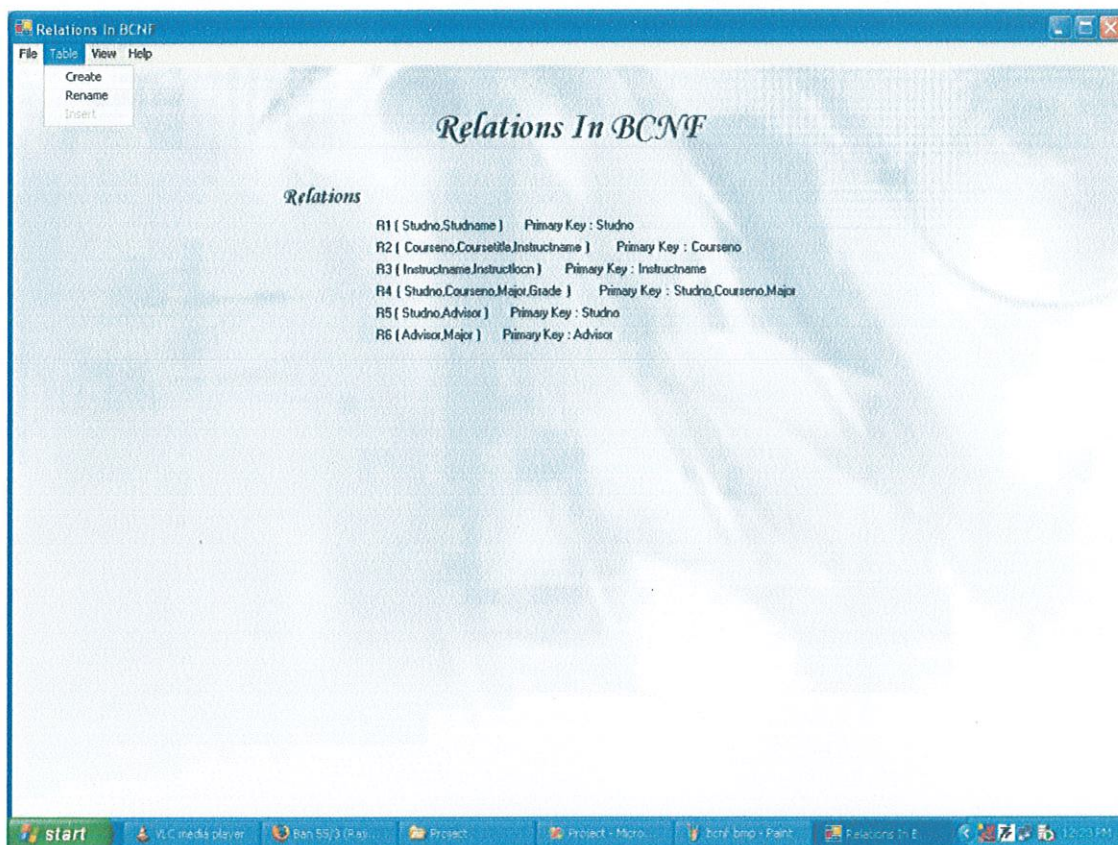


Fig. 7

Description

This page is the output page. On this Page we get the relations and the primary key corresponding to the related Relation. The Relations are in BCNF.

In the File Menu we have the option to **Save** the output (Relations and the Primary Keys) to a notepad file. We can also go back to the Main Page i.e. Synthesizing Normal Form Page or exit from the software.

In the Table Menu we have options of creating, renaming the tables and entering the data into the tables. Option **Create** creates the tables in the database. Option **Rename** renames the table in the database as well as on the output page. Option **Insert** opens the Insert Page from where we enter the data into the tables.

7.8 Relations In 4NF

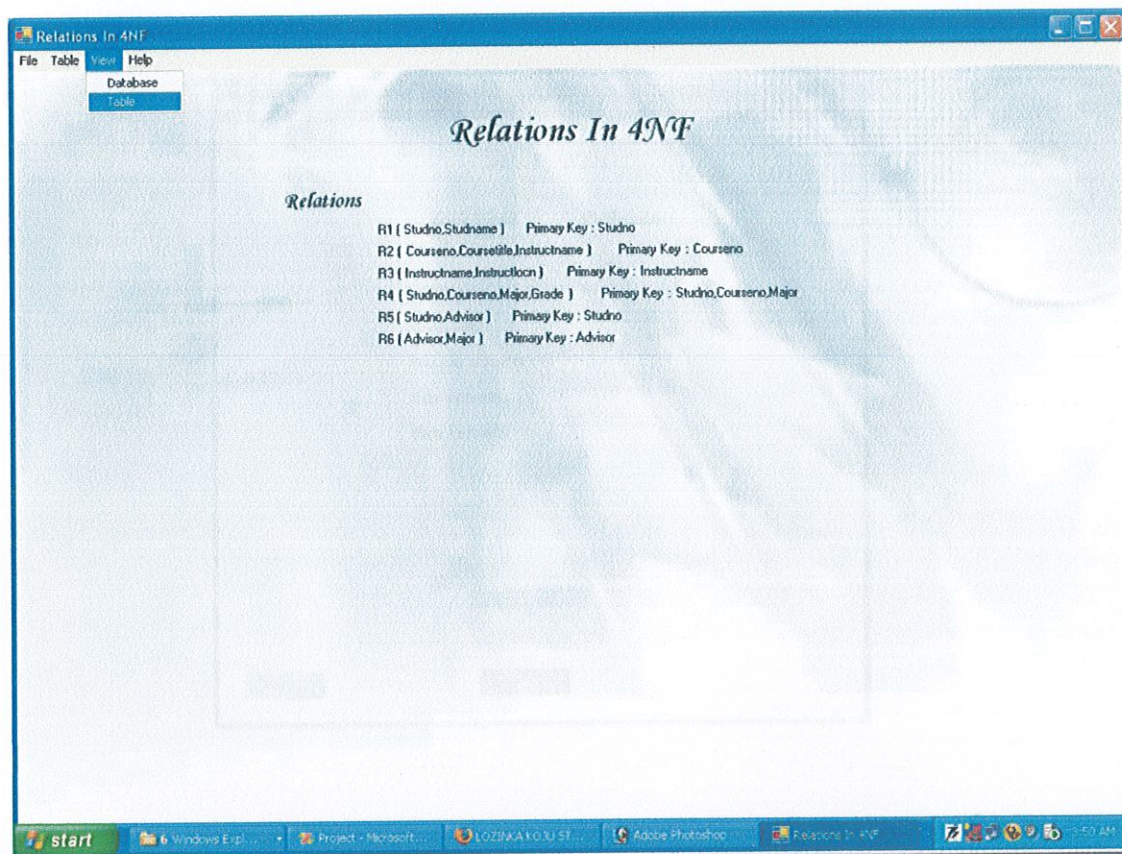


Fig. 8

Description

This page is the output page. On this Page we get the relations and the primary key corresponding to the related Relation. The Relations are in 4NF.

In the File Menu we have the option to Save the output (Relations and the Primary Keys) to a notepad file. We can also go back to the Main Page i.e. Synthesizing Normal Form Page or exit from the software.

In the Table Menu we have options of creating, renaming the tables and entering the data into the tables. Option **Create** creates the tables in the database. Option **Rename** renames the table in the database as well as on the output page. Option **Insert** opens the Insert Page from where we enter the data into the tables.

7.9 Insert Values In Database

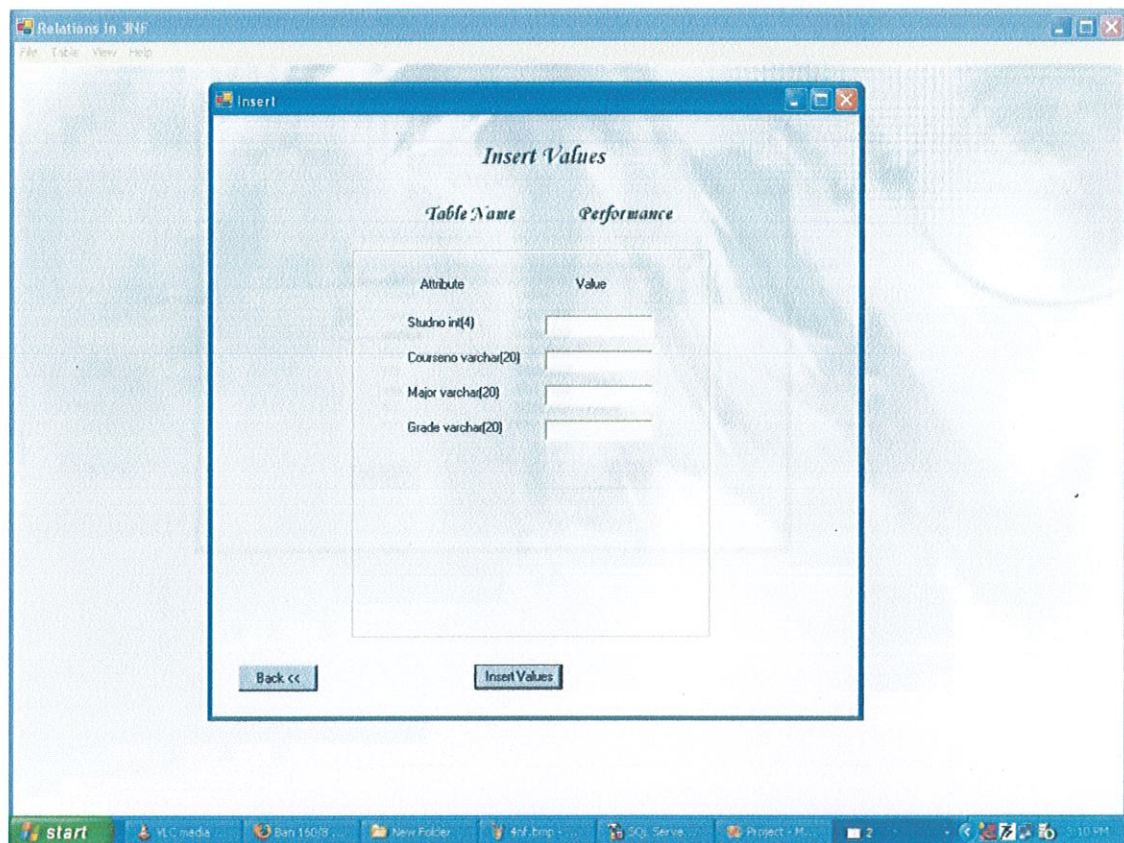


Fig. 9

Description

This page shows the table name and the attributes with their data types and length. The Insert Values button inserts the values into the database. We can insert as many entries we want to insert into the database. The **Back** button is to go back to the Relations Page.

7.10 View Database

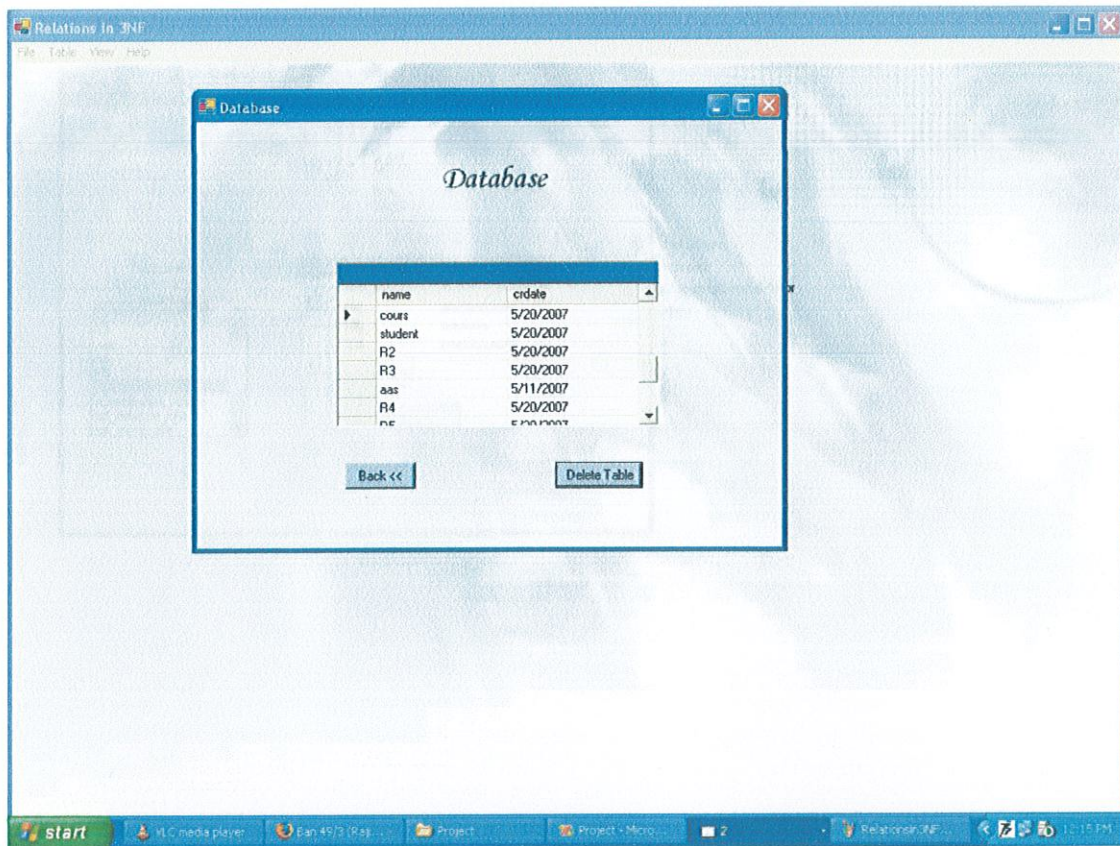


Fig. 10

Description

This Page shows all the tables in the database. We can delete any table from the database by clicking the Delete Table button. On clicking it, it opens a pop up asking for the table name to be deleted. On clicking OK the table gets deleted from the database.

7.11 Access Database Tables

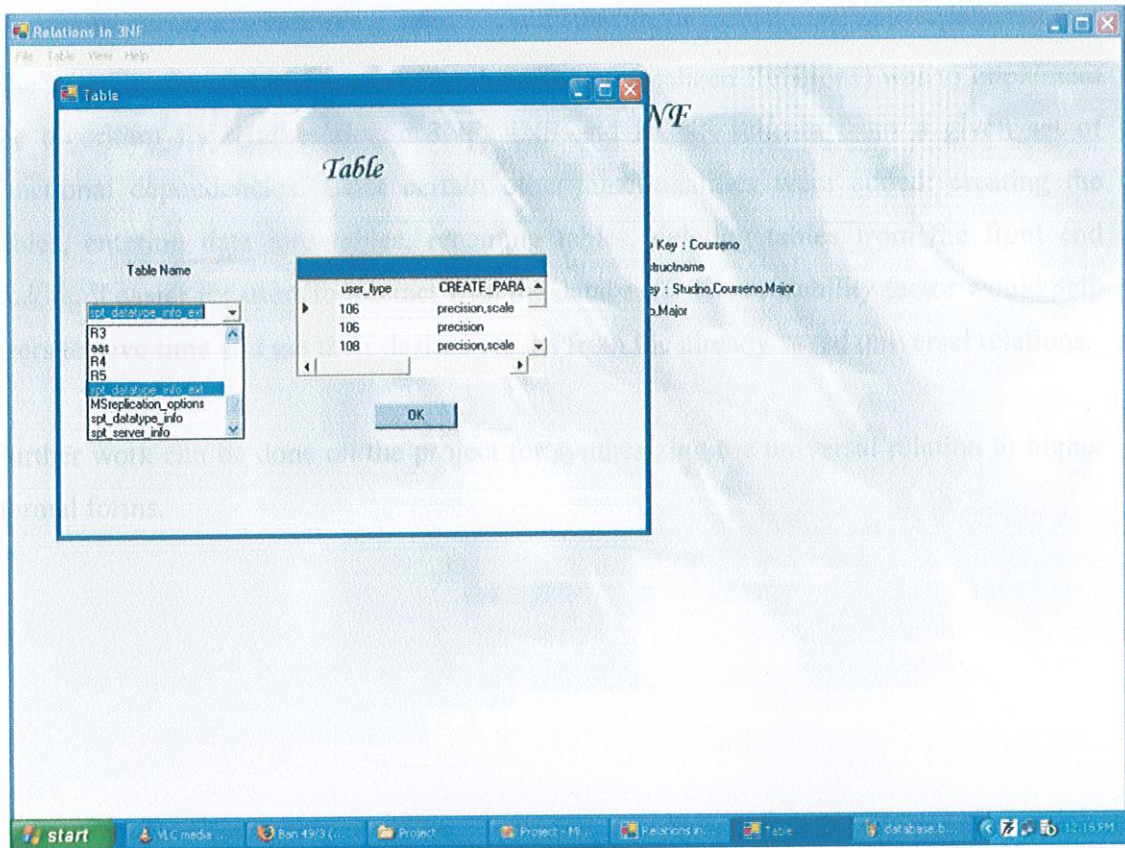


Fig. 11

Description

This Page has a Combobox which consists of all the tables in database. Selecting any table opens the table and shows all the entries in the table.

CONCLUSION

The initial purpose of this project (Synthesizing Normalized Relations) was to implement the algorithm for synthesizing a 3NF, 4NF and BCNF schema from a given set of functional dependencies. Later certain other functionalities were added: creating the tables, entering data into tables, renaming tables, deleting tables from the front end making it easier for users to interact with the databases. The reusability factor would help users to save time and get their desired results from the already saved universal relations.

Further work can be done on the project for synthesizing the universal relation to higher normal forms.

REFERENCE

1. *www.portal.acm.org*

- ACM is the world's first educational and scientific computing society.
- Association for Computer Machinery.
- The Best Portal To Access All Kind Of Research Work.

2. *www2.cs.ucy.ac.cy*

- University of Cyprus.
- Notes on Relational Model and Synthesis of Database Design.

3. *doi.ieeecomputersociety.org*

- Institute for Electrical and Electronics Engineers, Inc.
- A Unified Approach to Functional Dependencies and Normalization Function.

4. *www.cs.ucla.edu*

- University of California, Los Angeles.
- A Formal Approach to the Definition and the Design of Conceptual Schemata for Database Systems.

5. *IBM labs research papers, IBM Research Labs, San Jose, California*

- Research work done by E.F. Codd.
- On A Relational Model of Data for Large Shared Data Banks.

6. *ACM transaction of Database Systems, Vol. 7 (Published in March, 1982)*

- All Information Required for Database Related Research Work.

7. *International Journal of Information Technology Education, Vol. 1 (Seneate Hall Publishing 2004)*

- Traditional and Alternative Database Normalization Techniques.
- By Hisang-Jui Kung and Thomas Case(Georgia Southern University-
www.georgiasouthern.edu)

8. *www.utoronto.ca*

- Dep't. Of Computer Science, University of Toronto, Toronto, Canada
- Research papers on "Synthesizing Third Normal Form Relations from Functional Dependencies.

9. *www.ucdavis.edu*

- University of California, Davis
- New Methods and Fast Algorithms for Database Normalization given by Jim Diederich and Jack Milton

10. *Fundamentals of Database Systems*

- By Elmasri, Navathe, Somayajulu, Gupta

11. *Database Systems*

- By Thomas Connolly and Carolyn Begg