Automating the Lower and Higher Normal Form Process for the Database Systems

Manal Fadel Younis
Department of Computer Engineering, University of Baghdad
Baghdad-Iraq

ABSTRACT
Normalization is an important technique for the analysis of relational databases. It aims to create a set of relational tables with minimum data redundancy that preserve consistency and facilitate correct insertion, deletion, and modification. It is very much time consuming to do this data analysis manually. Thus in this paper, a system is proposed which aims to automate the most complex phase of the database design normalization. It will help to achieve a good database design and eliminate the drawbacks of manual normalization process.

This system is suitable to eliminate redundancy and inconsistent dependency automatically. It aims to handle the normalization process up to fifth normal. This includes creating tables and establishing relationships between those tables by using their general definitions in a step-by-step feature on the set of functional dependencies to remove redundant data. Then this system is tested on many examples with multiple candidate keys taken from different sources.

General Terms
Normalization, Relational Database System

Keywords
Functional Dependency, Keys, Redundancy, Normal Forms.

1. INTRODUCTION
Good relational database system is not enough to avoid the data redundancy. Normalization is a process used for evaluating and correcting table structures to minimize this redundancy, and reducing the data anomalies which is based on their functional dependencies and primary keys. The normalization process involves assigning attributes to tables based on the concept of determination [1].

It usually involves dividing a database into two or more tables and defining relationships between the tables. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships [2].

Key is one or more attributes which determine other attributes [1][3].

- Superkey: An attribute (or combination of attributes) that uniquely identifies each row in a table.
- Candidate key: A superkey that does not contain a subset of attributes that is itself a superkey.
- Non-prime attribute: A non-prime attribute is an attribute that does not occur in any candidate key.
- Primary key: A candidate key selected to uniquely identify all other attribute values in any given row cannot contain null entries.

2. A FUNCTIONAL DEPENDENCY (FD) [1][4][3]
The attribute B is functionally dependent on the attribute A if each value in column A determines one and only one value in column B. (written $A \rightarrow B$)

- **Partial dependency:** The determinant is only part of the primary key.
- **Full functional dependency:** If attribute B is functionally dependent on a composite key A but not on any subset of that composite key, the attribute B is fully functionally dependent on A.
- **Transitive dependency:** Is a functional dependence exists among nonprime attributes.
- **Multivalued dependency:** Two or more attributes are dependent on a determinant and each dependent attribute has a specific set of values. The values in these dependent attributes are independent of each other.
- **Join dependency:** A table T is subject to a join dependency if T can always be recreated by joining multiple tables each having a subset of the attributes of T.

Normalization works through a series of stages called normal form, these are [1][5]:

- First Normal Form (1NF).
- Second Normal Form (2NF).
- Third Normal Form (3NF).
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF).
- Fifth Normal Form (5NF).

The **first normal form** (**1NF**):

- There are no repeating groups in the table. In other words, each row/column intersection contains one and only one value, not a set of values.
- Define the primary key.
- Define all dependencies on the table.

The **second normal form** (**2NF**):

- It is in 1NF.
- Remove all partial dependencies.

The **third normal form** (**3NF**):

- It is in 2NF.
- It contains no transitive dependencies.

The **Boyce-Codd normal form (BCNF):**

- Every determinant in the table is a candidate key.

- Is special case of 3NF, when the table contains only one candidate key, then 3NF and the BCNF are equivalents.

The **fourth normal form (4NF):**

- It is in 3NF.
- Remove the multivalued dependencies.

The **fifth normal form:**

- It is in 4**NF.**
- The entity has no join dependencies. Also called project-join normal form.

This section 1 describes the introduction of the proposed work. Section 2 focuses on literature survey. Section 3 describes the proposed system for automatic higher normal form. Section 4 focuses on the result of the proposed system..

3. LITERATURE SURVEY

This section focus on literature survey of the paper:

Sherry Verma,"Comparing manual and automatic normalization techniques for relational database", [6] proposed the Comparing manual and automatic normalization techniques for relational database, based on the dependency matrix and approach primary key to generate automatically identified the final table.

Amir Hassan bahmani, Mahmoud Naghib zadeh, "Automatic database normalization and primary key generation", [7] the authors proposed an approach for automatic database normalization and primary key generation. In discussed an automatic distinguish one primary key for every final table which is generated. The problem is to normalize the database tables automatically. In the current normalization process, even first normal form, second normal form and third normal forms are difficult by doing automatically.

P.B. Alappanavar, Dhiraj Patil, Radhika Grover, Srishti Hunjan, Yuvraj Girnar ,"An Ameliorated Approach towards Automating the Relational Database Normalization Process", [8] aims to automate the most complex and elaborate phase of the database design process-Normalization, which will help to achieve the trademarks of an acceptable database design.

G.Sunitha, Dr.A.Jaya, "A KNOWLEDGE BASED APPROACH FOR AUTOMATIC DATABASE NORMALIZATION", [9] aims to provide automatic normalization of databases up to 3NF in order to reduce the time consuming in manually normalization.

Moussa Demba, "ALGORITHM FOR RELATIONAL DATABASE NORMALIZATION UP TO 3NF", [2] the author proposed an algorithmic approach for database normalization up to third normal form by taking into account all candidate keys, including the primary key.

4. PROPOSED SYSTEM FOR AUTOMATIC ALL NORMAL FORM

First Case study: Consider the following table with set of attributes to apply the proposed system:

Table 1: Employee

Name	Project	Task	Office	Floor	Phone
Bill	100X	T1	400	4	1400
		T2	400	4	1400
	200Y	T1	400	4	1400
		T2	400	4	1400
Sue	100X	T33	442	4	1442
	200Y	T33	442	4	1442
	300Z	T33	442	4	1442
Ed	100X	T2	588	5	1588

First Normal Form:

Step 1: Eliminate the Repeating Groups

Eliminate the nulls by making sure that each repeating group attribute contains an appropriate data value.

Step 2: Identify the Primary Key

The above table has more than one field that represent the primary key (Name, Project, Task) because the field Name is not uniquely identify all of the remaining entity (row) attributes. For example, the Name value Bill can identify any one of two projects. Then if the primary key composed from (Name and Project) so not uniquely identify any one of two tasks.

To maintain a proper primary key that will uniquely identify any attribute value, the new key must be composed of a combination of Name, Project and Task.

For example, if Name=Bill, Project=100X and Task=T1 the entries for the attributes Office, Floor and Phone must be 400, 4, 1400 and so on. This change converts the table Employee to table Employee2 which is in 1NF.

Table 2: Employee2

Name	**Project**	**Task**	**Office**	**Floor**	**Phone**
Bill	100X	T1	400	4	1400
Bill	100X	T2	400	4	1400
Bill	200Y	T1	400	4	1400
Bill	200Y	T2	400	4	1400
Sue	100X	T33	442	4	1442
Sue	200Y	T33	442	4	1442
Sue	300Z	T33	442	4	1442
Ed	100X	T2	588	5	1588

Step 3: Identify All Dependencies

The primary key in step 2 identified the following dependency:

Name, Project, Task → Office, Floor, Phone

a- Partial dependency:
Name→ Office, Floor, Phone

b- Fully dependency: None

c- Transitive dependency:

- **Office** is the office number for the employee. Bill works in office number 400.
- **Floor** is the floor on which the office is located.
- **Phone** is associated with the phone in the given office.

Office → Floor, Phone

d- Multivalued dependency: Name → Project
Name → Task

Second Normal Form:
Split the table which results from the 1NF according to partial dependency into two relations (tables):-

Table 3: EmpNPT (Name, Project, Task)

Name	Project	Task
Bill	100X	T1
Bill	100X	T2
Bill	200Y	T1
Bill	200Y	T2
Sue	100X	T33
Sue	200Y	T33
Sue	300Z	T33
Ed	100X	T2

Table 4:EmpNOFP (Name, Office, Floor, Phone)

Name	Office	Floor	Phone
Bill	400	4	1400
Sue	442	4	1442
Ed	588	5	1588

Third Normal Form:
Step 1: Identify the transitive dependency:-
There is transitive dependency in table EmpNOFP (Name, Office, Floor, Phone)

Step 2: Split EmpNOFP into two tables

Table 5 : EmpNPT (Name, Project, Task)

Name	Project	Task
Bill	100X	T1
Bill	100X	T2
Bill	200Y	T1
Bill	200Y	T2
Sue	100X	T33
Sue	200Y	T33
Sue	300Z	T33
Ed	100X	T2

Table 6: EmpNO (Name, Office)

Name	Office
Bill	400
Sue	442
Ed	588

Table 7: EmpOPF (Office, Phone,Floor)

Office	Phone	Floor
400	1400	4
442	1442	4
588	1588	5

Boyce-Codd Normal Form
Is special case of the 3NF, every determinant in the table is a candidate key. When the table contains only one candidate key, then the 3NF and the BCNF are equivalent. Then the above tables are in BCNF.

Fourth Normal Form:
Step 1: Determine the multivalued dependency:
Name → Project
Name → Task

Table 8: EmpNP (Name, Project)

Name	Project
Bill	100X
Bill	200Y
Sue	100X
Sue	200Y
Sue	300Z
Ed	100X

Table 9: EmpNT (Name, Task)

Name	Task
Bill	T1
Bill	T2
Sue	T33
Ed	T2

Table 10: EmpNO (Name, Office)

Name	Office	Floor
Bill	400	4
Sue	442	4
Ed	588	5

Table 11: EmpOPF (Office, Phone)

Office	Phone	Floor
400	1400	4
442	1442	4
588	1588	5

Second Case study of BCNF:

To check for BCNF, it must identify all the determinants and make sure that they are candidate keys. For example, STUDENT relation is given below with attributes StdID, Major and Advisor.

Table 12: Student

StdID	Major	Advisor
MS-100	Math	Prof. B
MS-100	Physics	Prof. S
MS-200	Chemistry	Prof. R
MS-300	Physics	Prof. A
MS-300	Math	Prof. B

The functional dependencies in this relation are:

StdID, major → Advisor

Advisor → Major

The following may be the candidate keys:

(StdID, Major) is one candidate key

(StdID, Advisor) is another candidate key

Suppose (StdID, Major) as a primary key for the STUDENT relation. It is represented as:

STUDENT (StdID, Major, Advisor)

The 'STUDENT' relation can easily be converted to BCNF by dividing it into two relations. The attribute that is determinant but not a candidate key (such as Advisor) must be placed in a

separate relation. It must be the key of that relation. Suppose, the two relations are STD_ADVISOR and ADV_MAJOR such as:

STD ADVISOR (StdID, Advisor)

ADV MAJOR (Advisor, Major)

Table 13: STDADV

StdID	Advisor
MS-100	Prof. B
MS-100	Prof. S
MS-200	Prof. R
MS-300	Prof. A
MS-300	Prof. B

Table 14: ADVMAJ

Advisor	Major
Prof. B	Math
Prof. S	Physics
Prof. R	Chemistry
Prof. A	Physics

Third Case study of 4NF:
Employee relation is shown below:

Table 15: Employee

Eid	Language	Skill
100	English	Teaching
100	Kurdish	Politics
100	English	Politics
100	Kurdish	Teaching
200	Arabic	Singing
200	English	Cooking
200	Arabic	Cooking
200	English	Singing

Step 1: Check the table must be in 3NF and BCNF.
Step 2: Determine the multivalued dependency:
Eid → Language
Eid → Skill
The table is projected to the following two non-loss projections which are in forth normal form:

Table 16: EL

Eid	Language
100	English
100	Kurdish
200	Arabic
200	English

Table 17: ES

Eid	Skill
100	Teaching
100	Politics
200	Singing
200	Cooking

Fourth Case study of 5NF is shown below:
Step 1: Check the table must be in 4NF.
Step 2: Check the join dependency.

Table 18: AgentCompanyProduct

Agent	Company	Product
Smith	Ford	Car
smith	Ford	Truck
smith	Gm	Car
smith	Gm	Truck
jones	Ford	Car

This relation has a join dependency (**Agent**, **Company**, **Product**) among the three projections:

R1(**Agent**,**Company**), R2(**Agent**,**Product**), and R3(**Company**,**Product**) of **AgentCompanyProduct**.

To remove the join dependency, it must create the following relations:

AC(**Agent**, **Company**)

AP(**Agent**, **Product**)

CP(**Company**, **Product**)

Table 19: AC

Agent	Company
Smith	Ford
Smith	Gm
Jones	Ford

Table 20: AP

Agent	Product
smith	Car
smith	Truck
jones	Car
smith	Truck

Table 21: CP

Company	Product
Ford	car
Ford	truck
Gm	car
Gm	truck

5. TESTING THE PROPOSED SYSTEM

This proposed approach aims to normalize the database automatically to reduce the time of the design. The following figure is showing this:

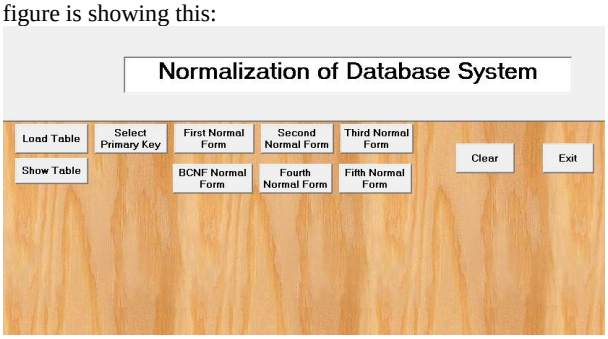

Fig 1: The main interface

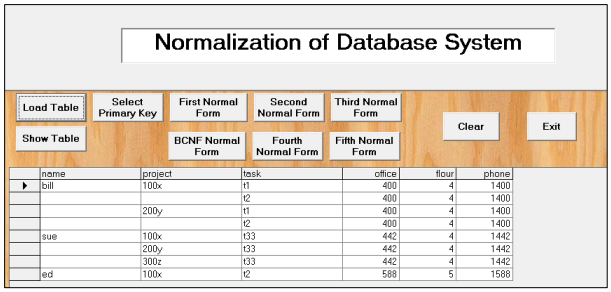

Fig 2: Load unnormalized table

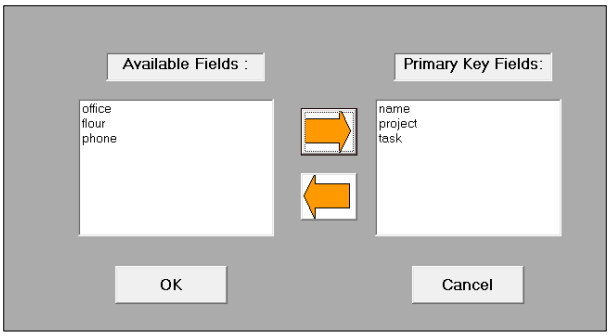

Fig 3: Select the primary key

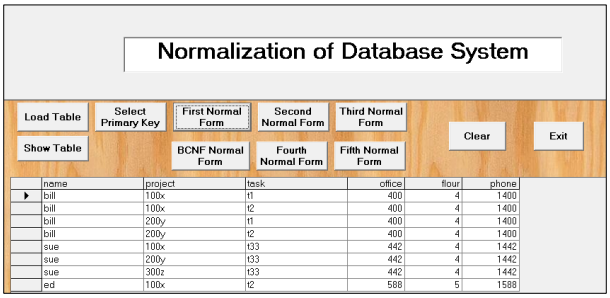

Fig 4: The table in 1NF

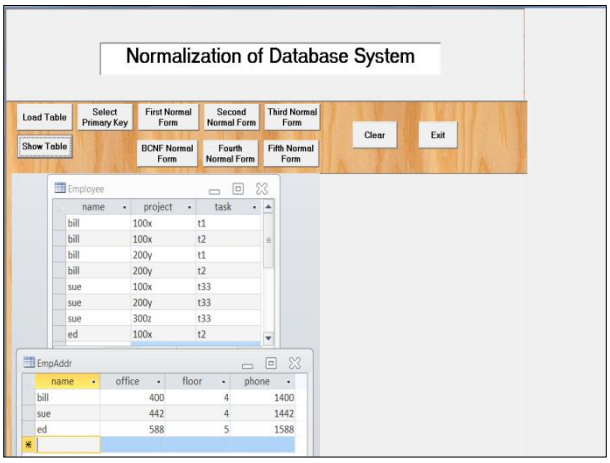

Fig 5: The table in second normal form

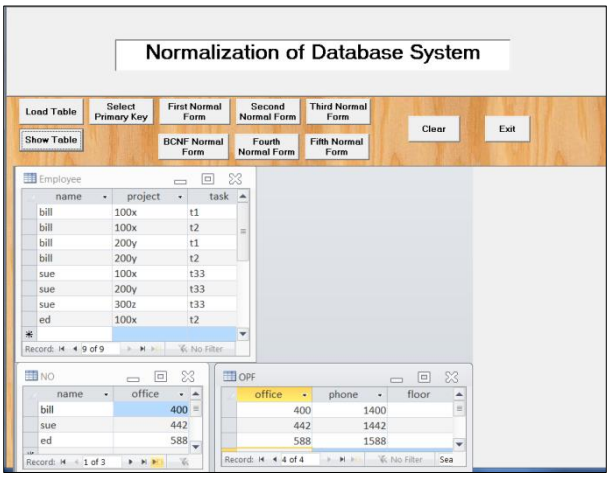

Fig 6: The table in 3NF

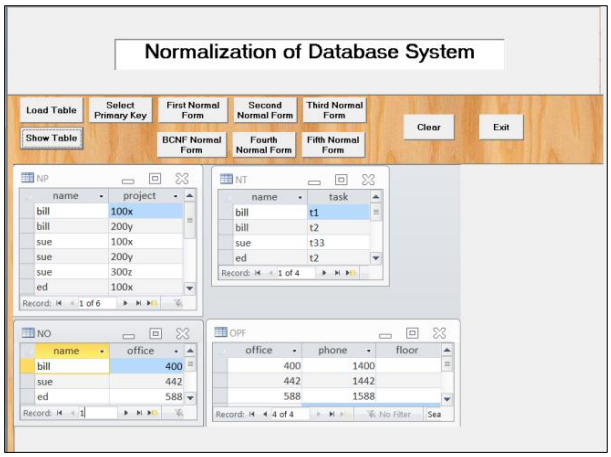

Fig 7: The tables in 4NF form

The result of the second case study about BCNF:

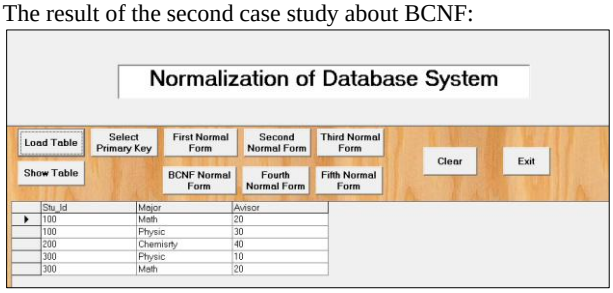

Fig 8: Load unnormalized table

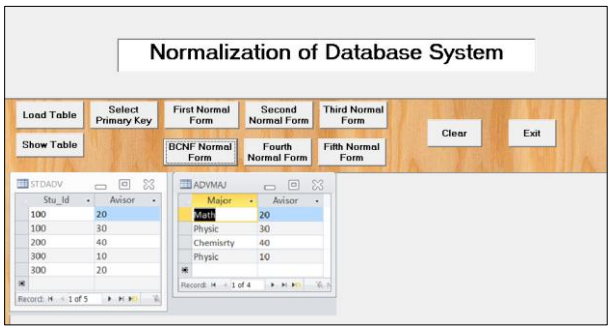

Fig 9: The tables in BCNF form

The third case study about 4th:

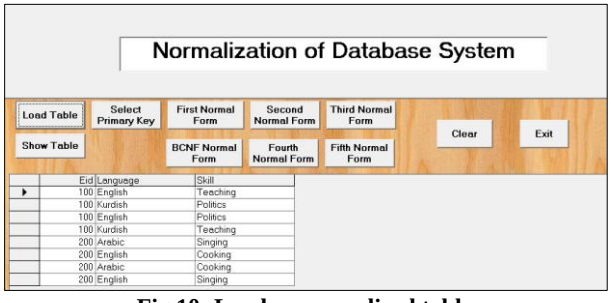

Fig 10: Load unnormalized table

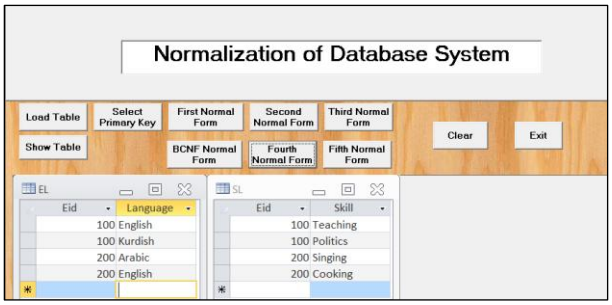

Fig 11: The tables in 4NF form

The third case study about 5th:

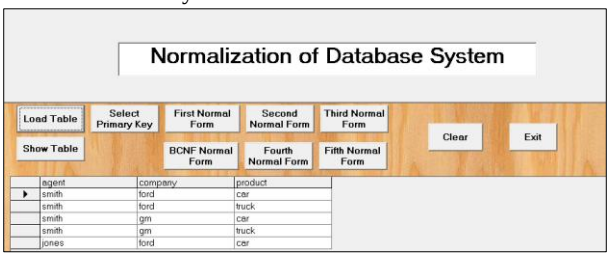

Fig 12: Load unnormalized table

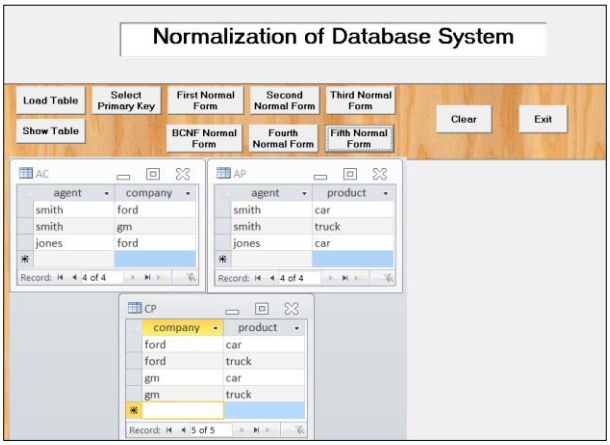

Fig 13: The table in 5th form

6. CONCLUSION

This work presented a system to perform database normalization up to fifth normal form automatically to design a database.

This work has the following advantages:

- Less time required for normalization.
- Easily remove the redundant dependencies.

7. REFERENCES

[1] Coronel C., Morris S., and Rob P. 2011. Database Systems: Design, Implementation, and Management, United States of America, 9th edition, 175-197.

[2] Demba M. 2013. Algorithm for Relational Database Normalization up to 3NF, International Journal of Database Management Systems (IJDMS) 5(3): 39-40, June.

[3] Sushant S. Sundikar Introduction to Database Management System.

[4] Ryan K., Ronald R., 2001. Database Design, United States of America, 208-216.

[5] Sushant. S. Sundikar Introduction to Database Management Systems, 5-8.

[6] Verma S. 2012. Comparing manual and automatic normalization techniques for relational database. IJREAS, 2(2): 59-67, Feb.

[7] Bahmani A, Naghibzadeh M., Bahmani B. 2008. An Automatic Database Normalization – Primary Key Generation, IEEE.

[8] Alappanavar B., Radhika P., Hunjan S., Girnar Y. 2013. An Ameliorated Approach towards Automating the Relational Database Normalization Process. International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, 2(4), April.

[9] Sunitha G., Jaya A. 2013. A Knowledge Based Approach for Automatic Database Normalization ISSN: 2278 – 1323 International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 2(5), May.