Developing visualization support system for teaching/learning database normalization

Olusegun Folorunso and AdioTaofeek Akinwale
Department of Computer Science, University of Agriculture, Abeokuta, Nigeria

Abstract

Purpose – In tertiary institution, some students find it hard to learn database design theory, in particular, database normalization. The purpose of this paper is to develop a visualization tool to give students an interactive hands-on experience in database normalization process.

Design/methodology/approach – The model-view-controller architecture is used to alleviate the black box syndrome associated with the study of algorithm behavior for database normalization process. The authors propose a visualization "exploratory" tool that assists the learners in understanding the actual behavior of the database normalization algorithms of choice and also in evaluating the validity/quality of the algorithm. This paper describes the visualization tool and its effectiveness in teaching and learning normalization forms and their functional dependencies.

Findings – The effectiveness of the tool has been evaluated in surveys. It shows that students generally viewed the tool more positively than the textbook technique. This difference is significant to $p < 0.05$ ($t = 1.645$). The mean interactions precision and calculated value using expert judge relevance ratings show a significant difference between visualization tool and textbook performance 3.74 against 2.61 for precision with calculated $t = 6.69$.

Originality/value – The visualization tool helped students validate/check their learning of normalization process. Consequently, the paper shows that the tool has a positive impact on students' perception.

Keywords Databases, Database theory, Students, Teaching, Learning, Interactive devices

Paper type Research paper

1. Introduction

In recent time, the volume of data in the databases increases geometrically with wide areas of applications in communications, health, education, finance, defense, and the development of internet. The issue of normalizing unwanted or redundant data has to be given deserved attention. Database normalization is a process of reducing redundancies of data in a database. It is a process or set of guidelines used to optimally design and redesigning a database. Since the introduction of Codd's (1970) seminal work on normal forms, Bernstein (1976), Diederich and Milton (1988), Concepcion and Villafuerte (1990) and Rosenthal and Reiner (1994) proposed algorithms and tools to synthesize a normalized database using functional dependencies (FDs). Maier (1988) indicated that relational data model theory (normalization) tends to be complex for the average designers. Jarvenpaa and Machesky (1989) and Bock and Ryan (1993) showed that relational data model leads to poor designer performance. Hence, the poor performance of normalization indicated that teaching normalization is a challenge to IS/IT educators (Kung and Tung, 2006).

The classical database normalization technique has often relied on the definition of normal forms. Some database textbooks include normalization algorithms to find the

Campus-Wide Information Systems
Vol. 27 No. 1, 2010
pp. 25-39

1065-0741
DOI 10.1108/10650741011011264

canonical cover by removing extraneous attributes of FDs and then to convert each FD in the canonical cover to a relation/table (Silberschatz *et al.*, 2002). The normalization algorithms often require extensive relational algebraic backgrounds that most computer science students lack. Most systems analysis and design textbooks rely on the definition of normal forms in their coverage of database normalization (Hoffer *et al.*, 2005; Avison and Fitzerald, 2002). They simply give the definition of first, second, and third normal forms (1NF, 2NF, and 3NF) and hope that students will be able to apply the definitions and normalize a set of tables. These approaches may not be the best way to help many computer science students effectively understand the database normalization process.

The main objective of this work is designing an educationally effective visualization of normalization processes in relational database management systems. Hundhausen *et al.* (2002) revealed that by graphically representing computer algorithms in action, algorithm visualization (AV) technology aims to help computer science students understand how algorithms work. Over three decades, evolution of software has endeavored to enhance computer science education in a variety of capacities. For example, AV software has been used to help:

- instructors illustrate operations in a lecture, e.g. Brown (1988);
- students as they study and learn about fundamental algorithms in computer science course, e.g. Gloor (1998); and
- students learn about the basic operations of an abstract data types in a computer science laboratory, e.g. Naps (1990).

With the listed points above, it shows that AV will support learning of database normalization.

The study of normalization provides the designer of schemas at this level with a useful set of concepts which, by their very structure, support integrity and consistency of the data. Visualizing the normalization processes will provide more insight as to how to design a schema than can pre-determine an ideal normal form for relation such that the relation will be free of all modification anomalies.

The focus of this work is on the various normal forms based on Codd's relational theory. The implementation (i.e. visualization of the processes) is limited to only 1NF, 2NF, and 3NF.

The remainder of this paper is organized as follows. Section 2 describes the related works, and Section 3 illustrates design methodology. The effectiveness of the visualization tool is evaluated and interpreted in Section 4, and Section 5 concludes the paper.

2. Related works

Self-explanation is one of the most effective learning strategies, resulting in deep knowledge. Mitrovic (2002) discussed how self-explanation is scaffolded in NORMIT, a data normalization tutor. NORMIT is a problem-solving environment, which complements traditional classroom instruction. The emphasis is therefore on problem solving, not on providing information. However, the system does provide help about the basic domain concepts, when there is evidence that the student does not understand them, or has difficulties applying knowledge. The system also insists on using the appropriate domain vocabulary; "talking science" has been shown to increase learning

and deep understanding of the domain. The student needs to log on to NORMIT first, and the first-time user gets a brief description of the system and data normalization in general. After logging in, the student needs to select the problem to work on. NORMIT lists all the pre-defined problems, so that the student may select one that looks interesting. In addition, the student may enter his/her own problem to work on. Data normalization is a procedural task; the student goes through a number of steps to analyze the quality of a database. NORMIT requires the student to complete the following steps while solving a problem: determine candidate keys, determine the closure of a set of attributes, determine prime attributes, simplify FDs, determine the normal form the table is in, if necessary, decompose the table so that all the final tables are in Boyce-Codd normal form (BCNF), this made the approach to be difficult for learners.

Jürgens (2004) developed database normalizer, an application that works with FDs to compute normalization properties of relational database schemas. It determines the normal form, a schema, and compute candidate keys and equivalent tuples. In addition to these analysis features it implements a synthesis algorithm that can create relational schemas from FDs that are guaranteed to be in 3NF and contain a minimal set of relations. This is meant to be used to support the process of learning about database normalization.

Giacomo (2004) also developed a Java-based software Normalizer that allows one to try out some relational database theory algorithms. It works on relations schemas, sets of attributes and sets of FDs. The main functions supported are: test if a relation satisfy BCNF or 3NF, decompose a relation in a set of relations all satisfying BCNF or 3NF, find all keys and prime attributes for a relation, compute closures of sets of attributes, compute projections of sets of dependences, and later find canonical covers of sets of dependences.

Kang and Tung (2006) developed a web-based tool to enhance teaching/learning of database normalization. They noticed that the decompositions happened when attributes on the right-hand side of FDs have more than one copy, and the number of the decomposed relations is exactly the same as the number of FDs. Hence, to decompose a relation into the 3NF, one simply eliminates extraneous attributes on the right-hand side of the FDs. The tool is being used for relational data modeling in systems analysis and design and data management courses. The paper describes the web-based tool and its effectiveness in teaching relational data model.

Georgiev (2008) developed a web-based learning environment, called learn database normalization (LDBN) with JavaScript and Ajax where students can test their knowledge on the subject of relational-database normalization online. Here, students meant to choose an assignment from a list with assignments, submitted by other users (lecturers). An assignment consists of a relational-database schema in universal-relation form, i.e. all the attributes in a single relation and a set of FDs on the attributes. After an assignment has been loaded, the students try to solve it through the LDBN process. The LDBN cannot handle multi-valued dependencies (MVDs) and thus cannot handle higher normal forms such as 4NF and above.

However, in all the papers reviewed, none referred to visualization as a better way of understanding database normalization processes, despite its various applications in data-intensive applications such as data mining, xml, ontology, etc. We are of the

opinion that developing a visualization tool for teaching database normalization will assist computer science students.

2.1 Theoretical view of database normalization technique

The following schema together and its FDs describe normalization technique using database theories and relational algebra

StudenProject (matric_number, student_name, Project_Number, Project_Name, Project_location, hours)

FD_1 = matric_Number → student_Name, project_Number, project_Name

FD_2 = project_Number → student_Name, project_Name

FD_3 = matric_Number, project_Number → student_Name, project_Name, project_Location, hours

(1) There is a need to retain all FDs on the left-hand side because they always have semantics of the relation attributes.

(2) There is a need to eliminate extraneous attributes on the right-hand side. The step is to eliminate both partial and transitive dependencies. For example, attributes student_Name, project_Number, and project_Name functionally depend on part of the whole key attribute matric_Number. Attribute student_Name, and project_Name in FD3 will be deleted as FD on attribute matric_Number and project_Number.

In FD_3 we have matric_Number, project_Number → hours, project_location

This has removed the partial FD.

When two FDs have the same number of attributes on the left-hand side, the attributes that have the fewer numbers of attribute on the right-hand side should be kept.

For example, attributes student_Name and project_Name appear in FDs FD_1 and FD_2. These attributes are transitively dependent on attribute matric_Number. These attributes (student_Name and project_Name) will be deleted from FD_1, A new FD is derived. New FDs after removing partial and transitive FD:

FD_1: matric_Number → project_Number

FD_2: project_Number → student_Name, project_Name

FD_3: matric_Number, project_Number → project_location, hours.

Assertion to what we have done:

> The schema does not contain attributes of tuples so it is in 1NF. The schema has eliminated partial functional dependency; hence it is in second normal form. Finally, it has eliminated transitive functionally dependency hence it is in third form.

The schema,

StudenProject (matric_number, student_name, Project_Number, Project_Name, Project_location, hours) and functional dependencies:

FD_1: matric_number → project_Number

FD_2: project_Number → student_Name, project_Name

FD_3: matric_Number, project_Number → Project_location, hours

violate both 2NF and 3NF. There is a need to decompose it by using decomposition algorithm as shown in Figure 1.

The result of decomposition is StudentProject FD into FD_4, FD_5, and FD_6 as follows:

FD_4: {matric_Number, project_Number}

FD_5: {project_Number, student_Name, project_Name}

FD_6: {project_Number, project_Location, hours} or {matric_Number, hours, project_Location}.

The decomposition as listed above is still the same as the result derived from normalization.

Based on database theories of Armstrong's inference rules, we can determine the keys of primary, secondary, candidate and super. Thus, for each set of attribute FD, we determine the set of FD^+ of attributes that are functionally determined by FD^+ of attributes that are functionally determined by FD. FD^+ is called the closure of FD under the schema and the closure algorithm is shown in Figure 2.

Consider the relation schema student_Project, it is possible to specify the FDs that would hold on the relation. The number of the closure is $2^N - 1$. In the case of our schema, we have $2^6 - 1 = 63$:

(1) { matric_Number}$^+$

(2) { student_name}$^+$

(3) { Project_Name}$^+$

Input: relation schema R and set of functional dependencies for R

1. Compute keys for R based on functional dependencies
2. Repeat until no more 3NF violations
3. Select any R' with AA→ BB that violates 3NF
4. Decompose R' into R_1 (AA,BB) and R_2 (AA,CC) where CC is all attributes in R' except (AAUBB)
5. Compute functional dependencies for R_1 and R_2
6. Compute key for R_1 and R_2 based on functional dependencies

Figure 1.
Algorithm for decomposition

```
Begin
   FD+ = FD
Repeat
   Old FD+ : = FD+
    For each functional dependency
       Y → Z  do
        If Y ≤ FD+ then
          FD+ : = FD+ UZ
      Until old FD+ : = FD+
    End
```

Figure 2.
Algorithm to determine FD^+, the closure of FD under the schema

(4) { Project_Number}$^{+}$

(5) { Project_location}$^{+}$

(6) { hours}$^{+}$

(7) { matric_number, student_name}$^{+}$

(8) {matric_number, Project_Name}$^{+}$

(9) {matric_number, Project_Number}$^{+}$

...

...

...

(63) {matric_number, matric_number, Project_Number, Project_Name, Project_location, hours}$^{+}$

Through the process matric_Number is the primary key while matric_Number and project_Number are candidate keys.

The determination of keys, elimination of partial and transitively functionally dependencies has been a difficult task to both database designers and students.

2.2 Model-view-controller

This is an architectural pattern used in software engineering. Successful use of the pattern isolates business logic (or other program logic) from the user interface, permitting one to be freely modified without affecting the other. The controller collects user input, the model manipulates application data, and the view presents results to the user. Typically, views and controllers come in pairs, each corresponding to a small part of the user interface; an application will likely be built with many pairs (Wikipedia, 2009).

Though model-view-controller comes in different flavors, control flow is generally as follows:

- The user (tutor/learner) interacts with the user interface in some way (for example, presses a mouse button).
- The controller handles the input event from the user interface, often via a registered handler or callback.
- The controller notifies the model of the user action, possibly resulting in a change in the model's state. (For example, the controller updates the normalization process.)
- A view uses the model (normalization algorithms) indirectly to generate an appropriate user interface (for example, the view displays the normalized results). The view gets its own data from the model. The model and controller have no direct knowledge of the view.
- The user interface waits for further user interactions, which restarts the cycle.

3. Design methodology

The architecture of the visualizing database normalization system as shown in Figure 3 is divided into two layers namely:

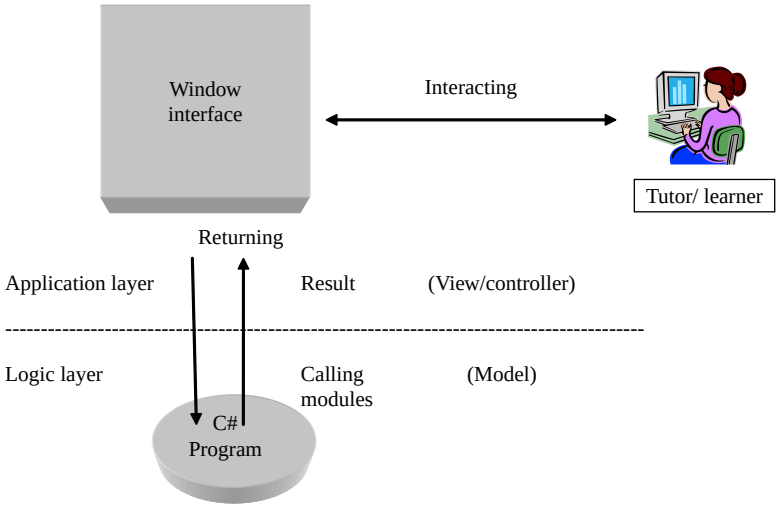

Figure 3.
Visualizing database normalization processes

(1) the application layer; and
(2) the logic layer.

3.1 Application layer (view/controller)
In this layer, the users such as students and tutors interact with the application/program interface through operations such as button clicks when moving from one step of the normalization process to another and when constructing FDs, typing text when entering relation name, the number of fields, and attributes names of the relational schema. All these operations generate events that send request to the logic layer.

3.2 Logic layer (model)
This layer can be referred to the model layer. The logic layer consists of 1NF, 2NF, and 3NF algorithms described in Figures 4-8. It sends the correct corresponding output of a particular request or module call. The logic behind the normalization process such as detecting if a particular normal form is satisfied or not as well as decomposition is coded here with C# pronounced as C-Sharp, an object oriented programming language. The normalization flowchart for coding the algorithms is shown in Figure 9.

A table is said to be 2NF if it is in 1NF and every attribute in the row is functionally dependent upon the whole key, and not just part of the key.

A relation is said to be in 3NF when it is in 2NF and every non-key attribute is functionally dependent only on the primary key.

4. Results and findings

The algorithms of 1NF, 2NF, and 3NF closure and decomposition were coded in C# programming language. The input interface was designed to submit parameters such as number of attributes, attribute names, FDs and buttons to accept parameters and normalize results. For demonstration purpose, four attributes were entered into the input interface in form of A, B, C, and D and with two FDs (AB → C, B → D).

```
Begin
 FOR Each Rows in Table T
    FOR  Each  Attribute in Row
        IF Attribute Contains More Than One Data value THEN
         OUTPUT" 1NF IS VIOLATED"
display red colour
GOTO 1:
        END IF
       END FOR
END FOR
OUTPUT "1NF IS NOT VIOLATED"
display green colour
1:
end
```

Figure 4.
First normal form (1NF) algorithm

```
Begin
IF 1NF IS SATISFIED THEN
BEGIN
   Get FunctionalDepencies  AS FunctDep
    GET PrimaryKey AS PriKey
    IF COUNT(PriKey) ==1 THEN
       OUTPUT "2NF IS SATISFIED"
       Display green colour
   ELSE
      IF (IsThereFunctionalDepencies(FunctDep,PriKey)==true) THEN
          OUTPUT "2NF IS VIOLATED" // display red colour
     ELSE
        OUTPUT "2NF IS SATISFIED"
      Display green colour
     END IF
ELSE
          OUTPUT"2NF IS VIOLATED" // display red colour
END IF
end
```

Figure 5.
Second normal form (2NF) algorithm

```
FUNCTION Bool  IsThereFunctionalDepencies(string FunctionalDependencies, string  PrimaryKey )
BEGIN
Bool State=false
SPLIT FunctionalDependencies into LeftHandSide and RightHandSide
FOR  EACH CHILD  LeftHandSide in FunctionalDependencies
   IF PrimaryKey=LeftHandSide THEN
      GET Current RightHandSide  AS  RightHandSide
IF (IsContainPrimaryKey(LeftHandSide,PrimaryKey)==true)  AND  LeftHandSide!=
PrimaryKey)
THEN
IF (IsContainPrimaryKey(LeftHandSide,PrimaryKey)==true)  AND  RightHandSide
IS NOT NULL) THEN
GET Current RightHandSide  AS  RightHandSide1
    IF (RightHandSide= RightHandSide1) THEN
     STATE=true
     GOTO 1:
END IF
END FUNCTION
```

Figure 6.
Computing the FDs of 2NF algorithms

```
FUNCTION Bool  IsContainPrimaryKey(String s,Array  myArray)
BEGIN
Bool State=false
IF COUNT(s)==1
 FOR EACH CHILD a IN myArray
  IF  s=a THEN
    State=true
   GOTO  1
  ELSE
 FOR EACH CHILD p IN PrimaryKey
     FOR EACH CHILD s1 IN s
        IF p=s1 THEN
        State=true
     END FOR
END FOR
END FOR
RETURN State
  1:
END FUNCTION
```

Figure 7.
Computing the primary keys of 2NF algorithms

```
begin
IF 2NF IS SATISFIED THEN
Bool State=true
Get FunctionalDepencies  AS FunctDep
Get PriamryKey As PriKey
SPLIT FunctionalDependencies into LeftHandSide and RightHandSide
FOR EACH CHILD p IN PriKey
FOR EACH CHILD r IN RightHandSide
    GET Current LeftHandSide
    IF LeftHandSide!=PriKey THEN
        State=false
  END IF
END FOR
END FOR
IF STATE=true THEN
 OUTPUT"3NF IS NOT VIOLATED"
display green colour
END IF
ELSE
 OUTPUT"3NF IS VIOLATED"// display red colour
END IF
end
```

Figure 8.
Third normal form (3NF) algorithm

4.1 Screen shots of the implementation

Figure 10 shows the input interface with accepted parameters. The visualization tool permits to normalize the schema based on the FDs the user submitted. The visualization tool normalizes any set of table to 1NF, 2NF, and 3NF if the set of FDs used are non-trivial. The user needs to press button next to see the step-by-step window which shows the step-by-step normalization process. Figures 11 and 12 show the normalization to 2NF and 3NF that eliminate both partial and transitive FDs. If the schema violates either 1NF or 2NF, there will be a red light button blinking as against the normal form, otherwise green light button blinks indicating that there is no violation. The tool permits the users to repeat the process many times in order to familiarize with normalization process.

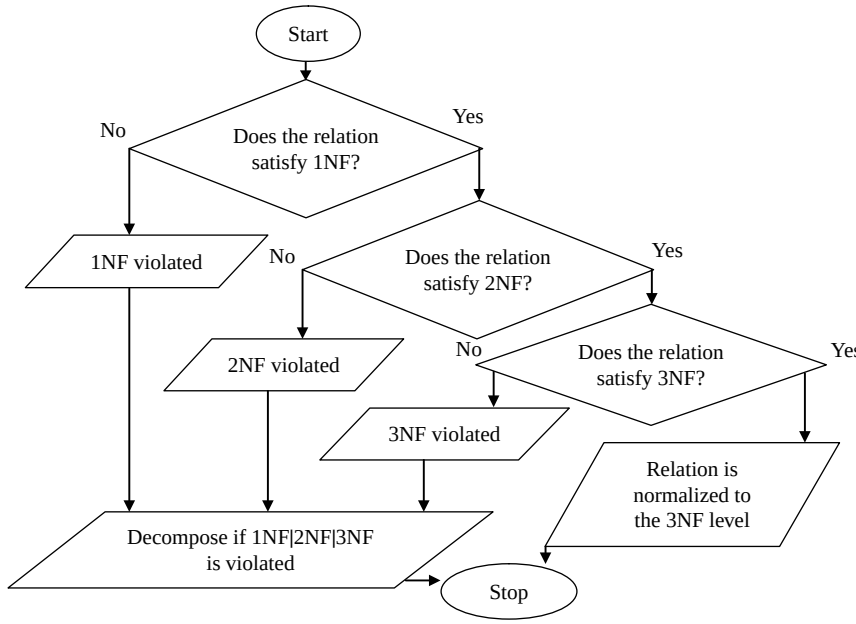

Figure 9.
Normalization process flowchart

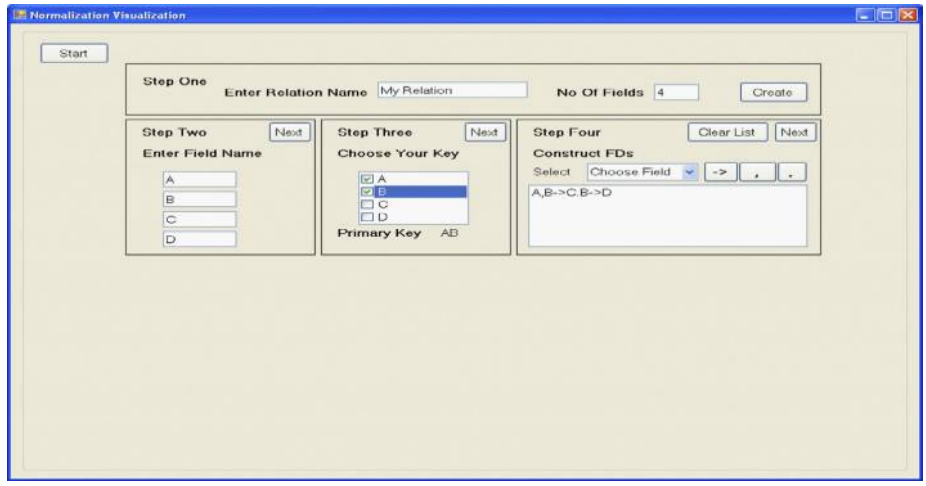

Figure 10.
Form showing the first step of the normalization process

4.2 Evaluation study

4.2.1 Design of the study. In total, 105 students in database design course participated in this study. The 15-week course met twice weekly. The Date (2000) textbook was used to cover feasibility study, data modeling, process modeling, and physical design. Students spent four periods of 75 minute on database normalization processes. The instructor spent one session explaining the importance of database normalization. The next session, the instructor demonstrated the textbook normalization technique

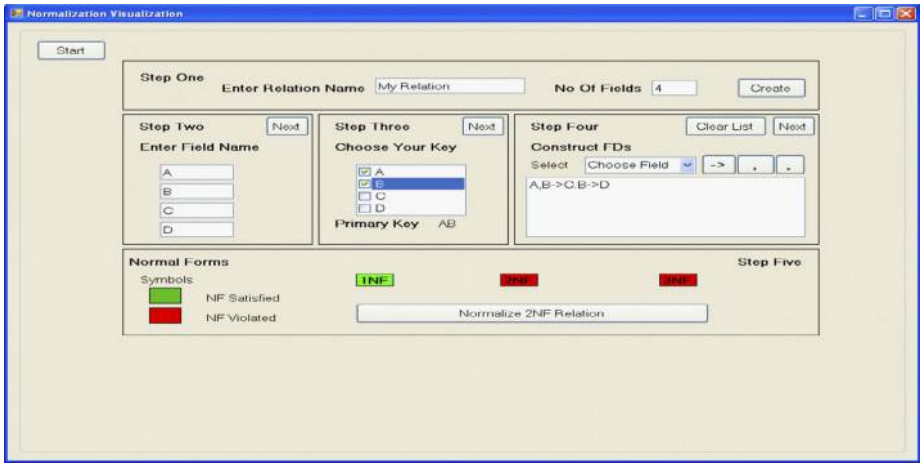

Figure 11.
Form showing a progress message of the normalization process

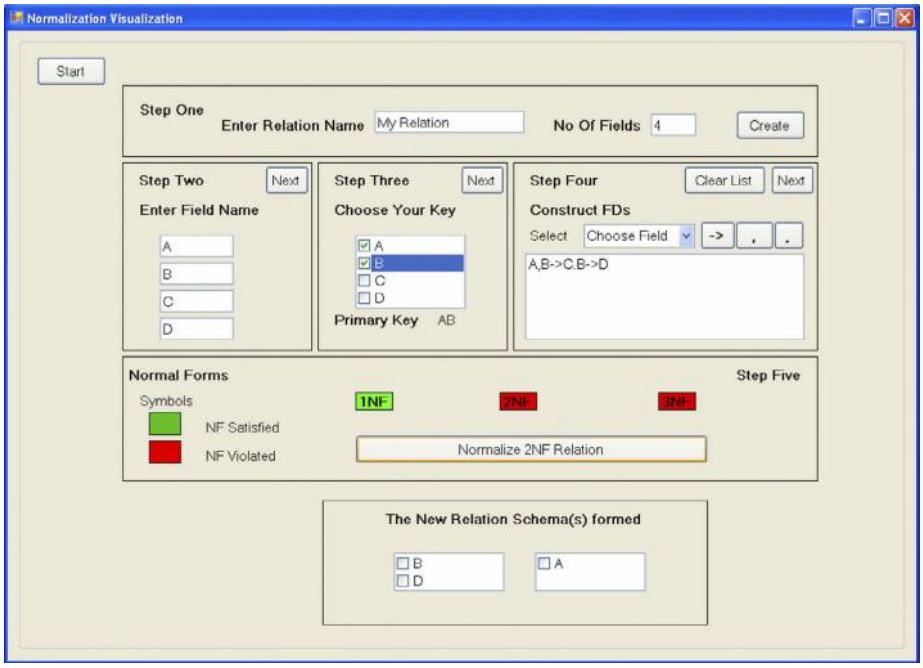

Figure 12.
Form showing the result of the normalization process based on the FDs constructed

using three normal forms with examples. The third session, students worked on two exercises using normalization techniques learned from previous session. The forth session, students exposed to the cookbook normalization technique and the visualization tool to solve the in-class exercises.

The research design is a single group pretest-posttest design. This design includes pretest measure followed by a treatment and a posttest for a single group. Surveys

were used to measure students' perceptions about the effectiveness of the textbook technique and the visualization tool in helping them to LDBN. The survey also asked students to provide their perceptions of their performance of the textbook technique and the tool.

4.2.2 Procedure. Each student was asked to complete a survey questionnaire after he/she applied the textbook technique and the visualization tool in order to assess the students' perceptions of the instructional effectiveness of the textbook technique and the tool.

The questionnaire, which was administered at the end of the third and forth sessions, consisted of five questions, with two asking for demographic information (gender and number of courses taken previously in discrete mathematics, data structure, algorithms, computer graphics, and databases) and three questions that asked students about the:

(1) difficulty of understanding the textbook technique/tool;

(2) helpfulness of the textbook technique/tool; and

(3) perceived grade/score of normalization after using the textbook technique/tool.

The questions were answered on a five-point Likert scale. An open-ended question was also included on the survey to enable the students to include their comments about the textbook normalization technique and the tool.

5. Results and analyses

Of the textbook technique, 100 students completed the survey and 105 completed the survey of the tool (completion was voluntary). The sample was predominantly male (58 male vs 47).

Table I summarizes the distribution of student demographics across treatment conditions.

The next set of analyses was performed to provide a general overview of student perceptions of the textbook technique and the visualization tool. The means and standard deviations for the three items on Part II of the survey questionnaire were calculated for both the textbook and tool (Table II). With three being the mid-point on

Table I.
Demographic frequency counts

	Gender	
Technique	Female	Male
Textbook	58	47
Visualization tool	58	46

Table II.
Mean student survey responses for textbook and tool

			Visualization tool	
Textbook	Mean	SD	Mean	SD
Difficulty	2.61	1.32	3.74	1.13
Helpfulness	2.54	1.42	3.86	1.33
Perceived grade	2.74	1.37	4.11	1.07

the scale, Table II illustrates that students generally viewed the tool more positively (with mean scores hovering around 4.11) than the textbook technique (whose mean scores hovered around 2.61-2.74). Table II also indicates that students generally perceived that they knew normalization better, grade-wise, using the tool.

Visualization tool over textbook normalization technique was also observed in responses to the open-ended question on the survey questionnaire. Most comments about the traditional technique were negatively toned. They suggest that students felt that the textbook technique took more time and effort to master (especially after exposure to the tool). In contrast, most comments about the tool were positively toned.

Furthermore, using visualization tool is better and less difficult than using textbook for learning database normalization. Consider the following *H0*: $\mu_1 = \mu_2$ and *H1*: $\mu_1 > \mu_2$.

This difference was significant to $p < 0.05$ ($t = 1.645$). The mean interactions precision and calculated value using expert judge relevance ratings show a significant difference between visualization tool and textbook performance 3.74 against 2.61 for precision at 95 percent confidence interval and $t = 6.69$. This suggests that visualization tool helped students validate/check their learning of normalization process by accepting the alternative hypothesis.

6. Conclusions and future research direction

The main design goal of the paper was to develop a system that is capable of teaching students database normalization processes effectively through the problem proposed by the students, and not just giving a sample one. Achieving the main goal was not an easy task. A lot of effort and time was invested in making this possible.

Another major goal was the development of a user friendly user interface (UI). Indeed, the UI is one of the key features considered in the course of implementation of the normalization system and is very critical for its success. Therefore, during the implementation process most of the efforts were concentrated on the development of the user interface. We believe that it will be well received by both students and tutor. At present, the system can handle relations up to third normal level, and so there is need for future extension to BCNF, 4NF, 5NF, and higher normal forms.

In achieving this, it would require the implementation of new data structures and algorithms which have to support MVDs.

Another possible direction for future development is a support of visualization of FDs constructed. Internationalization is also an important feature when it comes to educational software. This can be realized easily using the built-in tools of. Net Framework. It will only require small changes in the UI, once the first translations into other languages are ready.

References

Avison, D.E. and Fitzgerald, G. (2002), *Information Systems Development: Methodologies, Techniques and Tools*, 3rd ed., McGraw-Hill, London.

Bernstein, P.A. (1976), "Synthesizing third normal form relations from functional dependencies", *ACM Transactions on Database Systems*, Vol. 1 No. 4, pp. 277-98.

Bock, D.B. and Ryan, T. (1993), "Accuracy in modeling with extended entity relationship and object oriented data models", *Journal of Database Management*, Vol. 4 No. 4, pp. 30-9.

Brown, M.H. (1988), *Algorithm Animation*, MIT Press, Cambridge, MA.

Codd, E.F. (1970), "A relational model of data for large relational databases", *Communications of the ACM*, June, Vol. 13, pp. 377-87.

Concepcion, A.I. and Villafuerte, R.M. (1990), "Expert DB: an assistant database design system", *Proceedings of the Third International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Vol. 1.

Date, C.J. (2000), *An Introduction to Database Systems*, 7th ed., Addison-Wesley, Reading, MA.

Diederich, J. and Milton, J. (1988), "New methods and fast algorithms for database normalization", *ACM Transactions on Database Systems*, Vol. 13 No. 3, pp. 339-65.

Georgiev, N. (2008), "A web-based environment for learning normalization of relational database schemata", MSc thesis in Computing Science, Umeå University, Umeå.

Giacomo, D.R. (2004), "Dynamic visualization", *Proceedings CHI'94 Human Factors in Computing Systems*, ACM Press, New York, NY.

Gloor, P. (1998), "Animated algorithms", in Brown, M., Domingue, J., Price, B. and Stasko, J. (Eds), *Software Visualization: Programming as a Multimedia Experience*, MIT Press, Cambridge, MA, pp. 409-16.

Hoffer, J.A., George, J.F. and Valacich, J.S. (2005), *Modern Systems Analysis & Design*, 4th ed., Prentice-Hall, Englewood Cliffs, NJ.

Hundhausen, D.C., Douglas, S.A. and Stasko, J.T. (2002), "A meta-study of algorithm visualization effectiveness", *Journal of Visual Languages and Computing*, Vol. 13, pp. 259-90.

Jarvenpaa, S.L. and Machesky, J.J. (1989), "Data analysis and learning: an experimental study of data modelling tools", *International Journal of Man-Machine Studies*, Vol. 31, pp. 367-91, *Proceedings of the 2006 Southern Association for Information Systems Conference*, p. 258.

Jürgens, E. (2004), *Database Normalizer*, Technical University, Munchen.

Kung, H. and Tung, H. (2006), "A web-based tool to enhance teaching/learning database normalization", *Proceedings of the 2006 Southern Association for Information Systems Conference, Jacksonville, FL*, p. 8.

Maier, D. (1988), *The Theory of Relational Databases*, Computer Science Press, Rockville, MD.

Mitrovic, A. (2002), "NORMIT, a web-enabled tutor for database normalization", *Proceedings of the International Conference on Computers in Education ICCE, 2002, Auckland, New Zealand, December 3-6*, pp. 1276-80.

Naps, T. (1990), "Algorithm visualization in computer science laboratories", *Proceedings of the 21st SIGCSE Technical Symposium on Computer Science Education*, ACM Press, New York, NY, pp. 105-10.

Rosenthal, A. and Reiner, D. (1994), "Tools and transformations – rigorous and otherwise – for practical database design", *ACM Transactions on Database Systems*, Vol. 19 No. 2, pp. 167-211.

Silberschatz, A., Korth, H.F. and Sudarshan, S. (2002), *Database System Concepts*, 4th ed., McGraw-Hill, Boston, MA.

Wikipedia (2009), "Model-view-controller" (accessed on July 25, 2009).

Further reading

Felder, R. (1994), "Things I wish they had told me", *Journal of Information Systems Education*, Vol. 5 No. 1.

Appendix. Developing visualization support system for teaching database normalization

This work attempts to investigate how visualization tool supports the teaching of database normalization. The effectiveness of the tool needs to be evaluated. This calls for a survey to obtain the perception on the impact of visualization tool on students.

The following questions therefore pertain to visualizing database normalization. For each question, kindly tick the options that come closest to your view and please try to be as honest as possible when choosing your answers as your answers will be very important for making critical decisions.

Section A: (Demographic questions)

1. Gender

Male ☐ Female ☐

2. AGE

(A) 20-29 Years ☐

(B) 30-39 Years ☐

(E) 40-50 Years ☐

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.

Questions	SD	D	N	A	SA
A: (text book technique)					
1. The difficulty of understanding the textbook technique;					
2. The helpfulness of the textbook technique;					
3. The perceived grade/score of normalization after using the textbook technique					
B: (Visualization tool option)					
1. The difficulty of understanding the tool;					
2. The helpfulness of the tool;					
3. The perceived grade/score of normalization after using the tool					

C: Comment freely on your opinion about the difference(s) between Visualization tool and text book techniques _____

_____

_____

_____

_____

_____

_____

Corresponding author
Olusegun Folorunso can be contacted at: folorunsolusegun@yahoo.com