ACADEMIC World Academic Union

ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 7, No. 1, 2012, *pp*. 003-010

A Visualization Tool for Teaching and Learning Database Decomposition System

Akinwale Adio Taofiki[+] and Arogundade Oluwasefunmi Tale

Department of Computer Science, University of Agriculture, Abeokuta, Nigeria

(*Received September 25, 2011, accepted December 2, 2011*)

Abstract. Database designers and students of computer science in tertiary institutions find it difficult to learn database theory in which database decomposition is a major aspect. This work presents a visualization tool to simplify the study of decomposition algorithms behaviours which would assist the students and database designers to decompose any relational schema with its functional dependencies in case it violates third normal form or Boyce Codd Normal Form. Students perceptions are captured via translating decomposition algorithms into visualization tool for entering relational attributes with their functional dependencies. Students are requested to move an arrow from one normal form to another and the system specifies a green light as an indication of normalized form or red light as an indication of violation of normal form for learning purposes. Students use the tool to understand the abstract knowledge of decomposition techniques and easily use it to decompose any relation if it violates either third normal form or Boyce Codd normal form. The effectiveness of the tool has been evaluated in surveys and the students generally viewed it more positively than conventional classroom teaching.

Keywords: functional dependencies, database keys, normal forms, dependencies preserving, algorithms, relational schemas

1. Introduction

The main purpose of schema normalization is avoidance of redundancies that can be problematic. Reducing the number of functional dependencies within one table is main idea to avoid redundancy, but we may loose efficiency and transparency. Normal forms guarantee that functional dependencies only involve key attributes. Transformation of a relation into normal forms as a result of anomalies results into decomposition of a relation schema. This is done by replacing relation R by two or more relation schemas R's that each contains a subset of the attributes of R and together include all attributes of R. The benefits of the decomposition into normal forms reduce redundancy and storage wastage as well as solving insert, update and delete anomalies. It appears often to be a better database design if the number of relations is not too big and makes it easier to maintain the information in the database in a consistent state A relation is in good form if the relation R_i preferably should be in either 3NF or BCNF. In the case that a relation R is not in good form, decompose it into a set of relations (R_1, R_2, R_3,…, R_n) such that the decomposition is a lossless join decomposition and each relation is dependency preservation (Hector G,. Ullman J.D., Widom J., 2009). BCNF and 3NF are in position to check lossless join decomposition and dependency preservation. To check if a relation schema R is in 3NF or BCNF, it suffices to check only the dependencies in the given set functional dependencies F for violation of 3NF or BCNF rather than checking all dependencies in F^+ closure. If none of the dependencies in F causes a violation of 3NF or BCNF then none of the dependencies in F^+ will cause a violation of 3NF or BCNF either. However, using F is incorrect when testing a relation in a decomposition of R. Assuming we have a schema R(ABCD) with functional dependencies (FD) A→B and B→C. The keys are AD, ABD and ACD. Based on these keys and FD, let decompose the relation into R_1(AB) and R_2(ACD). Neither of the dependencies in F contain only attributes from ACD so we might be mislead into thinking R2 satisfies 3NF or BCNF. In fact, dependency A→C in F^+ shows that R_2 is not in BCNF.

The poor decomposition of a relation when violation 3NF or BCNF indicated that teaching relational decomposition is a challenge to database designers and computer science students. The classical database

[+] *E-mail address*: aatakinwale@yahoo.com and shewflaw@yahoo.com

decomposition technique has often relied in the definition of decomposition algorithms. Some database textbooks and tutorials include decomposition algorithms to split a relation into various relations if it violates the principles of 3NF or BCNF using FDs and hope the students will be able to apply the definition of the decomposition algorithms. These approaches do not encourage many computer science students to efficiently capture the principle of relational schema decomposition. For example, for every set of attributes $\alpha \subseteq R_i$, we need to check that α^+, the attribute closure of α either includes no attribute of $R_i - \alpha$ or includes all attributes of R_i. If the above condition is violated by some $\alpha \rightarrow \beta$ in F, the dependency $\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i$ can be shown to hold in R_i and R_i violate BCNF. This aspects requires extensive data structure background that most computer science students lack. The objective of this work is to design an educationally efficient visualization of decomposition relational schema in database management system.

2. Literature review

E.F. Codd, the inventor of the relational model, introduced the concept of normalization and what we now know as the First Normal Form (1NF) in 1970. Codd went on to define the Second Normal Form (2NF) and Third Normal Form (3NF) in 1971, and Codd and Raymond F. Boyce defined the Boyce-Codd Normal Form in 1974. Higher normal forms were defined by other theorists in subsequent years, the most recent being the Sixth Normal Form (6NF) introduced by Chris Date, Hugh Darwen, and Nikos Lorentzos in 2002. Since the introduction of E.F Codd's work on normal forms in 1970, Bernstein (1976), Diederich and Milton (1988), Concepcion and Villafuerte (1990), and Reiner (1994), proposed tools and algorithms to synthesize a normalized database using functional dependencies. Maier (1998) indicated that normalization tends to be complex for average designers. Jarvenpaa and Machseky (1989) and Bock and Ryan (1993) showed that relational data model leads to poor designer performance. All these imply that teaching normalization is a challenge to information technology tutors (Kung and Tung, 2006).

The study of normalization provides the designer of schemas with a useful set of concepts which supports the integrity and consistency of data. Visualizing the process will aid better understanding as how to design a schema that provides a normal form for a relation such that relation will be free of all data anomalies (insertion, deletion, and update).

Mitrovic (2002) developed a self-explanatory tool called NORMIT, a data normalization tutor. NORMIT is a problem-solving environment, which complements conventional classroom teaching and deep understanding of the domain.

Jurgens (2004) developed Database Normalizer (DN) which works with functional dependencies 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 functional dependencies that are guaranteed to be in third normal form 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 Relational Schemas, sets of Attributes and sets of Functional Dependencies. The main functions supported are: test if a relation satisfies 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 dependencies, and later find canonical covers of sets of dependencies.

Kung and Tung (2006) developed a web-based tool to enhance teaching/learning of database normalization. The tool is being used for relational data modeling in systems analysis and design database management courses.

Nikolay Georgiev (2007) developed a web-based learning environment, called LDBN (Learn Database Normalization) 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 list of assignments, submitted by other users (lecturers). After an assignment has been loaded, the students try to solve it through the LDBN process. The LDBN cannot handle multi-valued dependencies (MVD) and thus cannot handle higher normal forms such as 4NF and above.

Murray & Guimaraes (2009) presented developed Animated Courseware Support for Teaching Database

Design. This paper presents a set of software animations designed to support the teaching of database design concepts. They are intended to provide supplemental instructional support and also provide students with additional learning opportunities.

In all the papers reviewed above, none referred to visualization as a more effective way of understanding database decomposition process. Hence, developing an effective visualization tool for teaching and learning database decomposition, with emphasis on a simple user friendly interface that will assist computer science students as well as database designers better understand the theoretical concepts underlying database decomposition.

3. Theoretical aspect of decomposition

The decomposition of R into relations with attribute sets $\alpha \text{ and } \beta$ is dependency preserving if

$$(F_\alpha \cup F_\beta)^+ = F^+$$

and attribute sets α, $\beta \text{ and } \chi$ are also dependency preserving if

$$\alpha \to \beta\chi \Rightarrow \alpha \to \beta \wedge \alpha \to \chi$$

F_α is the set of all functional dependencies in F^+ that involve only attribute in α. This means that taking the dependencies in $F_\alpha \text{ and } F_\beta$ and computing the closure of their union gets all dependencies in the closure of F back.

Assuming $\alpha \to \beta$ is the dependency that generated $R_i = (R_1, R_2, R_3, ..., R_n)$ in the algorithm. P must be in $\alpha \text{ or } \beta$. Since P is in R_i and $\alpha \to \beta$ generate R_i.

Let consider two possible cases

(1) P is in β but not α

(2) P is in α but not β

Case one: P is in β but not α

Let define the three conditions for 3NF

(1) $\alpha \to \beta$ is a trivial functional dependency

(2) α contains a key for R

(3) Every $P \in \beta$ is part of some super keys of R

The second condition of 3NF is satisfied since α must be a super key.

Case two: P is in α but not β

The third condition in the definition of 3NF is satisfied since α is a super key.

Assume α is not a super key and we have

$\alpha \to \beta$: *functional dependency in R*

$\alpha \to \beta$: *functional dependency that was used to generate* R_i

$\forall_\alpha$ must contain some attributes that are not in α

(i) Since $\alpha \to \beta$ is in F^+ it must be derivable from F_c by using attribute closure on α

(ii) Attribute closure cannot have use $\alpha \to \beta$ if it had been used. α must be contained in the attribute closure of α which is not possible since we assumed α is not a super key

(iii) Using $\alpha \to (\beta - \{P\}) \text{ and } \alpha \to \beta$, we can derive $\alpha \to \beta$. In this case $\alpha \to P$ is nontrivial functional dependency since $\alpha \subseteq \alpha\beta \text{ and } P \notin \alpha$

(iv) Then, P is extraneous in the right hand side of $\alpha \to \beta$, which is not possible because $\alpha \to \beta$ is in F_c

(v) Thus, if P is in β then α must be a super key

4. Decomposition design methodology

The architecture of the visualization database decomposition system is divided into two layers; namely, application and logic layer

4.1. Application layer

In this layer, the users interact with the application interface through operations of entering relational name, number of attributes and construction of functional dependencies (fds). All these operations generate events by moving from one step to another and each move sends request to the logic layer.

4.2. Logic layer

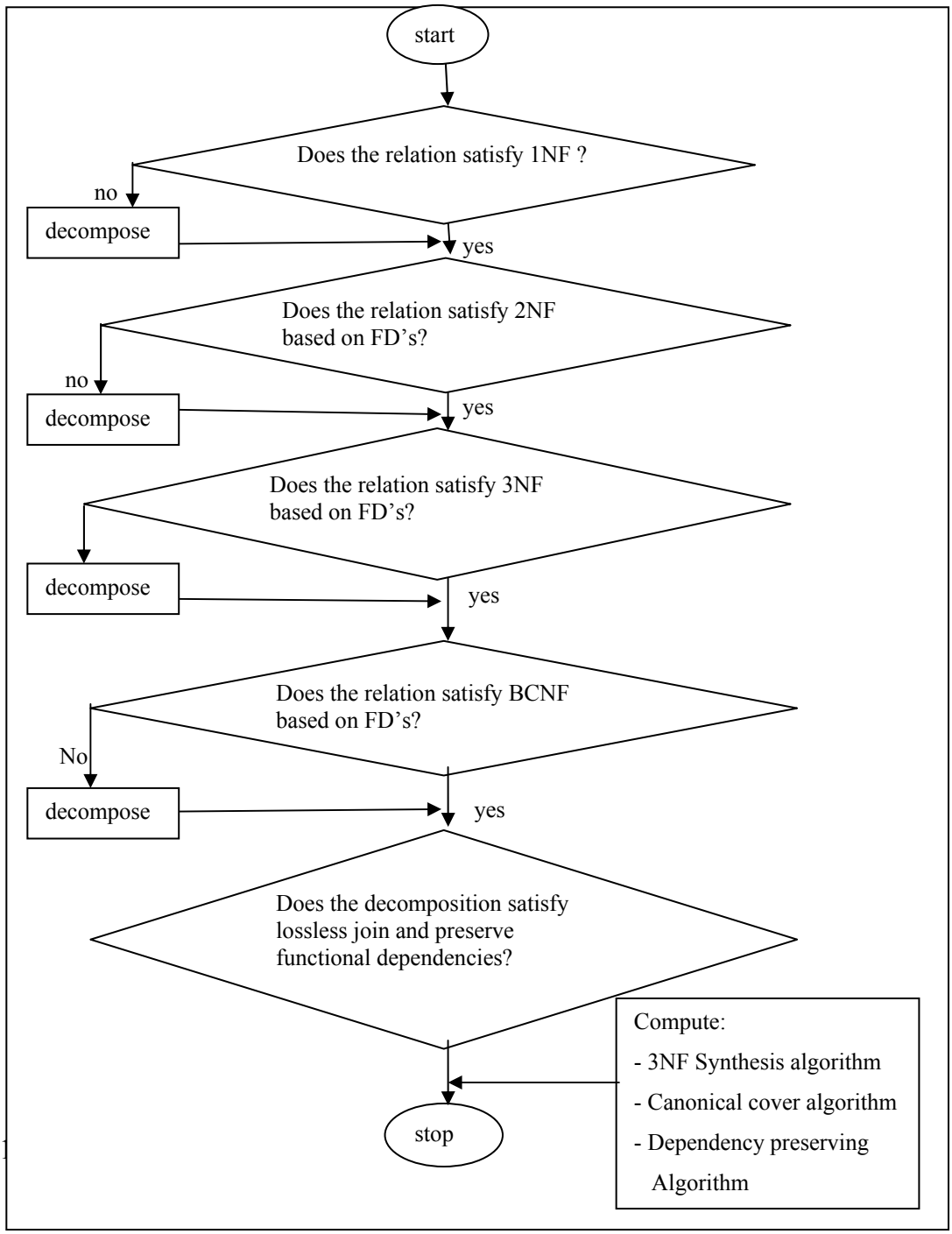

Figure 1

The logic layer consists of decomposition of 1NF to 2NF if a relation violates 1NF, 2NF to 3NF if it violates 2NF, 3NF to BCNF if it violates 3NF and BCNF if it violates BCNF. This step is illustrated in figure 1. Decomposition of a relation from 1NF to 2NF or 3NF or BCNF relies in splitting the relational schema into two or more relations taking into consideration the principle of normalization of atomic attributes, prime and non-prime attributes, partial and transitive dependencies and consideration of the super key impact. For example, if a relation violates 3NF, its decomposition algorithm is illustrated in figure 2. Figure 3 also shows Boyce Codd Normal Form decomposition algorithm in case any relation violates BCNF. If a relation fails to satisfy the principles of lossless join and preservation dependencies, the work employ 3NF synthesis algorithm, canonical cover and dependency preservation algorithms as described in figure 4a and 4b together with 3NF decomposition algorithm in figure 2.

Let F_c be the canonical cover of F
n = 0;
for each dependency $\alpha \to \beta$ in F_c
if none of schemes in $R_{i\ (i=1,2,3,\ldots n-1)}$ contains $\alpha\beta$ then
n = n + 1;
$R_n = \alpha\beta$
end if
If none of the schemes in $R_{i\ (i=1,2,3,\ldots n-1)}$ is contained in R_n
remove R_i
end if
end for
If none of the schemes $R_{i\ (i=1,2,3,\ldots n-1)}$ contains a candidate key for R then
n = n + 1;
R_n = any candidate key for R
end if
return $(R_1, R_2, R_3, \ldots, R_n)$

Figure 2: Decomposition algorithm into 3NF

result = $\{R\}$
execute = false
compute F^+
while (not execute) do
if (there is a schema R_i in result that is not in BCNF) then
begin
let $\alpha \to \beta$ be a nontrivial functional dependency that holds on R_i such that
$\alpha \to R_i$ is not in F^+ and $\alpha \cap \beta = \phi$
result = (result - R_i) $\cup (R_i - \beta) \cup (\alpha, \beta)$;
end;
execute = true

Figure 3: BCNF decomposition algorithm

The theory in section 3 and decomposition design in figure 1 show that a relation that is in BCNF which satisfies lossless join operation and preservation dependencies is also equal to a relation in 3NF after computing the algorithms of 3NF synthesis, canonical cover and preservation dependency.

For example, a canonical cover F_c for F is a set of dependencies F_c for which $F_c \Leftrightarrow F$, and there is no functional dependency in F_c that are superfluous or containing extraneous attribute. Each left side of

functional dependency in F_c is unique. extraneous attribute A in $\alpha \to \beta$ in R is defined as

$$1:\ A \in \alpha;\ F \Rightarrow F - \{\alpha - \beta\} \cup \{(\alpha - A) \to \beta\}$$
$$2:\ A \subset \beta;\ F - \{\alpha \to \beta\} \cup \{\alpha \to (\beta - A)\} \Rightarrow F$$

The result of the theory is that there is not always a lossless join decomposition in BCNF which is also dependency preserving. Moreso, there is always a lossless join decomposition into 3NF which is also dependency preserving with respect to F

$F_c = F$;
repeat
  apply union role (right side of functional dependency)
  find functional dependency with extraneous attributes
  (left or right side)
  and delete them
until F_c does not change

Figure 4a: Canonical cover algorithm

Result = α ;
  While (changes to result) do
    For each R_i in the decomposition
      T = (result $\cap$ R_i)$^+$ $\cap$ R_i
      Result = result $\cup$ t

Figure 4b: Proposed dependency preservation algorithm

5. System implementation

The algorithms of the first, second, third and boyce-codd normal forms and decompositions in figure 2, 3, 4a. and 4b were coded in Java programming language. The work also employed closure and database keys algorithms to derive keys which were used to determine normal forms violation. The interface was designed to accept the relation name, number of attributes, attribute names and functional dependencies. The system accepts these parameters and decompose the relation schema if it violates any normal form. It is possible for the system to use the derived keys to check if a relation violates 3NF or BCNF. For demonstration purpose, a relation R and three attributes A, B and C are entered into figure 5 together with functional dependencies of A→B, B→C. Based on the number of attributes and functional dependencies, the figure displays the derived keys and unordered normal forms of 1NF, 2NF, 3NF and BCNF. The user can move from one normal form to another in other to check if a relation violates the normal forms. The effect change is displayed by showing green or red light. Green light indicates that the normal form is okay while red light indicates violation and the system is automatically decomposed into appropriate relations. By moving the arrow into 3NF as shown in figure 5, the system displays red which indicates violation and the system was decomposed into R1(A, B) and R2(B, C). The R3() in the figure 5 shows that the relation can only be decomposed into R1 and R2.

The same process was performed by changing the functional dependencies into {A→B, AB→C} using the same attribute of A, B and C as illustrated in figure 6. By moving the arrow into 3NF, the result shows that the relation does not violate by indicating a green light.

6. Results and findings

Students in the Department of Computer Science, University of Agriculture, Abeokuta, Nigeria, normally offer database design (CSC422) at 400 level. Out of fifteen weeks course lecture of 60 hours, Lecturers and Tutors always use three weeks on normalization / decomposition topic. In their examination question, one question usually comes up. For the academic session between 2005 and 2008, eighty five out of four hundred and twenty three did normalization / decomposition questions. The detailed comments on student's performance on question by question by the External Examiner indicated that students found it difficult to decompose a relation when it violates either 3NF or BCNF using decomposition algorithm.

Due to the poor performance of the students on normalization / decomposition topic, a visualization tool

explained above was developed as part of teaching technique. In the academic session of 2008/09 and 2009/10, two hundred and nine out of two hundred and nineteen students answered the normalization / decomposition question using the developed tool. The performance on this question was impressive according to the External Examiner report on question by question. The means and standard deviation were computed based on the step-wise grade of the range of 70-100 as A, 60-69 as B and 50-59 as C. The other ranges are 45-59 as D, 40-44 as E and 0-39 as F. The performance computation for the academic session between 2005 and 2008 using conventional teaching and academic session from 2008 to 2010 using the visualization tool for teaching decomposition techniques was compared as shown in table 1.

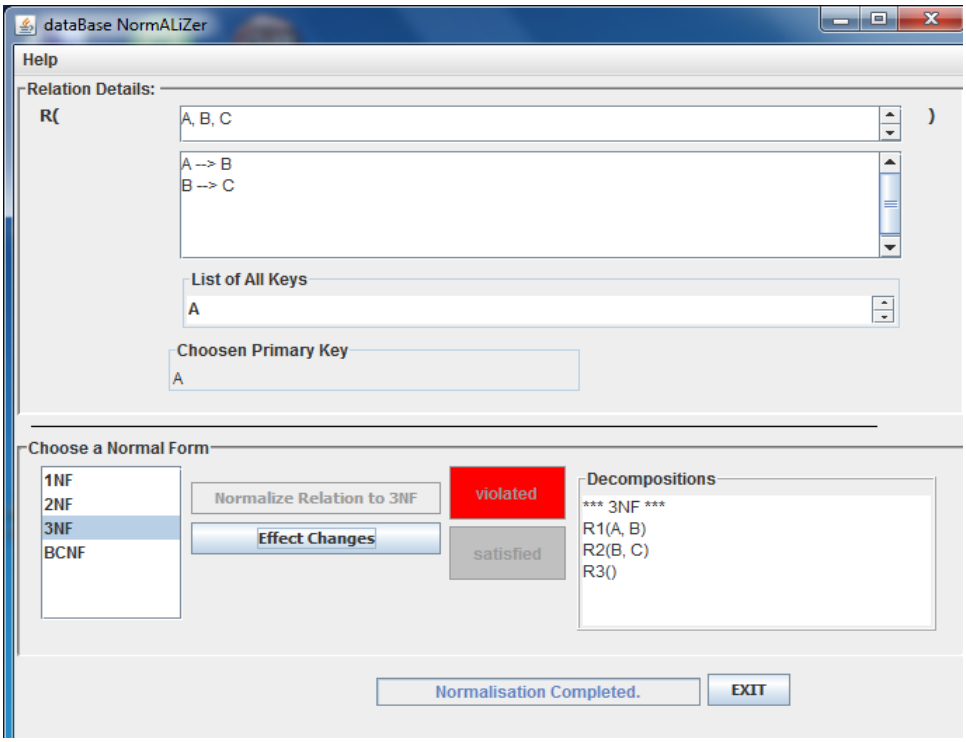

Figure 5: Sample of decomposition process by selecting 3NF

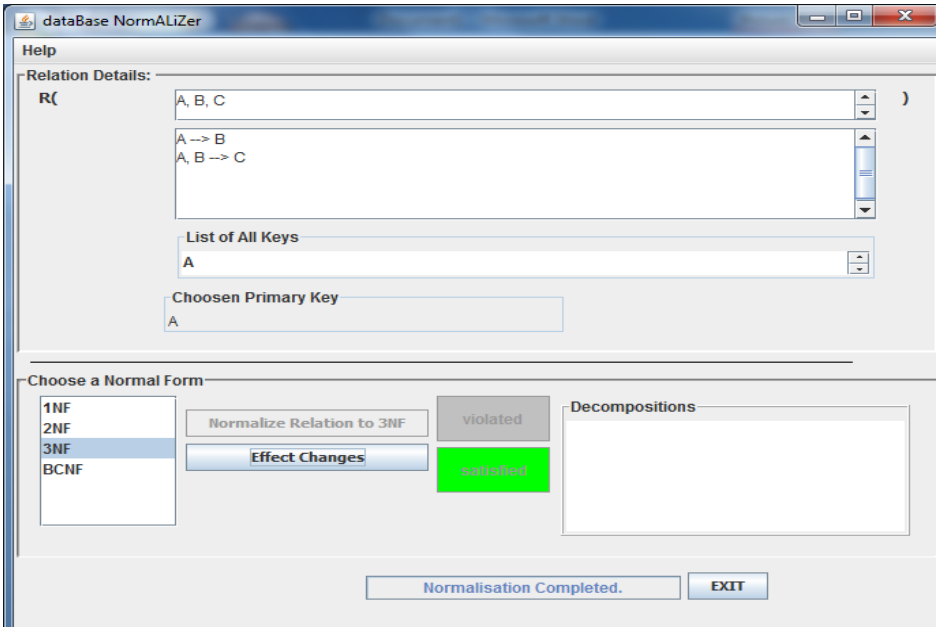

Figure 6: Sample of decomposition process by replacing FDs.

Table 1: Means and standard deviation of student performance

	Conventional		Visualization	
	Mean	STD. DEV.	Mean	STD. DEV.
Difficulty	3.73	1.17	4.33	1.07
Helpfulness	2.97	1.09	3.70	0.92
Perceived Grade	2.83	0.95	3.97	0.67

From the table 1, it can be seen that the conventional technique of teaching database decomposition is more difficult to understand with a mean score of 3.73 compared to 4.33 for the visualization tool. The visualization tool is more helpful than conventional with standard deviation score of 0.92 compared to 1.09 for conventional. The perceived grade after using the visualization tool is higher than conventional with mean of 3.97 as against 2.83 for conventional/ Hence; visualization tool helps students validate/check their learning of database decomposition.

7. Conclusions and future research direction

A visualization tool for teaching and learning decomposition of relational database schemas has been developed. All the necessary decomposition algorithms were used to generate a user friendly interface for accepting input parameters of relation and its functional dependencies. The tool decomposes relational schema if it violates 3NF or BCNF. In survey, the result shows that the tool has a positive impact on students' perception to learn database normalization / decomposition topic. Effort is on to make sure that the decomposed relations are reversed back to the normal schema without loss or adding extraneous values.

8. References

[1] P. A. Benstern. Synthesizing third normal form relations from functional dependencies. *ACM Transactions in database systems.* 1996, **1**(4): 277-298.

[2] D.B Bock and T. Ryyan. Accuracy in modeling with extended entity relationship and object oriented data models. *Journal of database management*. 1993, **4**(4): 30-39.

[3] E. F. Codd. A relational model of data for large relational databases. *Communication of the ACM*. Vol. 12, pp. 377-387

[4] A. I Concepcion and R. M. Villafuerte. As Assistant Database Design System. *Proceedings of the Third International Conference on Industrial Engineering Application of Artificial Intelligence and Expert Systems*. 1990.

[5] C. J. Date. *An Introduction to Database System*(7th Edition). Addison-Wesley, Reading MA, 2000.

[6] J. Diederic and, J. Milton. New Methods and Fast Algorithms for Database Normalization. *ACM Transaction on Database Systems*. 1988, **13**(3): 339-365.

[7] D. R. Giacomo. Dynamic Visualization. *Proceeding CHI94, Human Factors in Computing Systems*. New York: ACM Press, 2004.

[8] G. Hector, J.D. Ullman and J. Widom. *Database System*. Addison-Wesley Prentice Hell, 2009.

[9] S. I. Jarvenpaa and J. J. Data Analysis and Learning, An Experimental Study of Data Modeling Tools. *International Journal of Man-Machine Studies*. 1989, **31**: 367-391.

[10] E. Jurgen. *Database Normalizor***.** Technical University, Munchen Germany, 2004.

[11] H. Kung and H. Tung. A web-Based Tool to enhance Teaching/Learning Database Normalization. *Proceedings of the 2006 Southern Association for Information Systems Conference*. Jacksonville, 2006.

[12] D. Maier. *The Theory of Relational Database*. Rockville MD: Computer Science Press, 1998.

[13] A. Mitrovic. A Web-Enabled Tutor for Database Normalization. *Proceedings of the International Conference on Computers in Education (ICCE).* New Zealand: Auckland. 2002, pp. 1276-1280.

[14] E. Ramez and N. Shamkant. *Fundamental of Database Systems*. Benjamin Cummings Publishing Company, 1994.

[15] D. Reiner and A. Rosenthall. A Tools and Transformation Rigorous and Otherwise – for Practical Database Design. *ACM Transactions on Database Systems.* 1994, **19**(2): 167-211.

[16] G. Nikolay. Learnt Database Normalization. http://www.cs.umu.se.education.examina.reporter/n

[17] M. Murray and M. Guimaraes. Animated Courseware Support for Teaching Database Design. *Journal of Computing Science*. 2008, **24**(2): 144-150.