Facilitating Learning by Practice and Examples: a Tool for Learning Table Normalization

Junhu Wang
Griffith University
Gold Coast, Australia
j.wang@griffith.edu.au

Bela Stantic
Griffith University
Gold Coast, Australia
b.stantic@griffith.edu.au

ABSTRACT

We describe the design and evaluation of a web-based tool to help students learn database normalization, which is an important topic in relational database design. Compared with existing systems, our tool has the advantage of allowing a user to practice with as many examples as he/she likes, at all possible levels of difficulty, showing the detailed steps for each solution, and allowing users to upload existing examples and saving their own examples for later reference. Our user survey and observation indicate that the tool is well liked by those who used it, and the tool has impacted them positively in their learning.

CCS CONCEPTS

• **Social and professional topics** → *Information technology education.*

KEYWORDS

Relational database, Normalization, Learning by example, Learning outcome, Learning experience

ACM Reference Format:
Junhu Wang and Bela Stantic. 2019. Facilitating Learning by Practice and Examples: a Tool for Learning Table Normalization . In *Proceedings of 9th Balkan Conference on Informatics (BCI'19). September 26-28, Sofia, Bulgaria, 2019.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3351556.3351589

1 INTRODUCTION

Table normalization is an important technique for relational database design, which is taught in many Information Technology (IT) programs. Over the last 20 years or so, the authors taught the subject to both undergraduate and postgraduate students at several universities in Australia and overseas, and we observed that normalization is one of the topics many students were struggling with. It is particularly hard for students who lack prerequisite knowledge such that mathematical reasoning skills. This coincides with the observations made in [4, 8, 10].

Doing exercises and practicing with examples is an essential and effective way to enhance the learning outcomes [5, 7]. On the other hand, most database textbooks (e.g., [1, 3, 11, 12]) provide very limited number of examples and exercises for normalization. Even though the lecturers can make more, they cannot show them all to the students due to class time constraints. Furthermore, in our case (and we believe we are not alone), there are always significant percentages of students who cannot attend the interactive classes, and there are students from a large variety of backgrounds with different levels of prerequisite knowledge and intellectual capabilities, making the teaching and learning even more difficult. This highlights the need for a tool that can allow the students to practice with all types of examples at their own pace, at suitable levels of difficulty, with detailed step-by-step instructions.

In this paper, we describe the design and evaluation of a tool for the above purpose. Our tool is freely available at http://www.ict.griffith.edu.au/~jw/normalization/ind.php.

Existing tools and related work. Although there are systems reported for teaching normalization in the literature (e.g., NORMIT [9], [8], LDBN [6], and [4]), these tools cannot meet our requirements: either they are not freely available or they have limited functionalities. For instance, NORMIT is a system which needs licensing, and more importantly it has only a fixed set of 50 exercises. The tool reported in [8] supports normalization into 3NF only using an algorithm not commonly taught in the textbooks, and it does not support finding key attributes, finding minimal cover of functional dependencies, and thus might not work for cases where there are multiple candidate keys. Also the system supports examples with up to 10 functional dependencies only. [4] implemented the same algorithm as in [8], and it focuses more on visualization. Neither the tool in [8] nor that in [4] can be found by repeated search using Google, indicating they may not be freely available.

LDBN is the one we found still available, but it is designed to be more of an evaluation system: students need to choose from a list of given exercises submitted by the lecturer, and solve the exercise themselves, and then they can use the system to check whether their solution is correct or not. Although a solution can be worked out by the system, there is no step-by-step explanation. Earlier non-web-based systems (e.g., [2]) and current commercial systems (such as Visible Analyst Database Engineer (VADE)) do not meet our requirements either, for instance, neither do they provide step-by-step explanation of the normalization process, nor do they provide examples or exercises, as the purpose of these products is not to help students learn.

Compared with previous tools, our system has the following distinct features:

- Our system is publicly available on the internet, with a friendly user interface. It allows the user to choose from

ACM ISBN 978-1-4503-7193-3/19/09…$15.00
https://doi.org/10.1145/3351556.3351589

pre-designed examples, or construct arbitrary new examples, with practically no limitation on the number of attributes a table can have or the number of functional dependencies one can specify. If the user finds some examples interesting, he/she can save them for later reference or for showing them to the lecturer or fellow students.

- Our system covers all functionalities required by normalization using functional dependencies. It can find a minimal cover of a given set of FDs, find all candidate keys (and key attributes), check for 2NF, 3NF and BCNF, and normalize a table to any of these normal forms. For normalization into 3NF, we implemented two algorithms, one preserves FDs, the other may not. We believe this is useful to help the students understand related concepts and algorithms.
- For every functionality, detailed steps for reaching a solution can be shown, by clicking the "Show Steps" button.
- Since the main purpose of our system is helping students learn, the algorithms we implemented are rigorously correct, meanwhile we tried to use algorithms that are also heuristically efficient (for details see Section 2).
- Lecturers can also use our system as a teaching tool: in addition to the examples already in the system collection, the lecturer can construct his own examples, store them on his/her local computer, and re-use them in his/her teaching.

Our system has been used by our Information Technology and Computer Science students, as well as students in some overseas institutions. Our user survey indicates that it is very much liked by all students and lecturers who tried it.

2 TECHNICAL BACKGROUND

Table normalization is an essential process in ensuring a table has less or no data redundancy caused by various data dependencies. While higher normal forms (e.g., 4NF, 5NF) were studied by researchers, in reality, most practitioners are only concerned with data redundancy caused by functional dependencies (FD), that is, they are only concerned with normal forms such as the third normal form (3NF) and Boyce-Codd normal form (BCNF). As such, many introductory database courses only teach normal forms that are based on functional dependencies. Our tool is concerned with normal forms defined using FDs as well. To make it easier to understand our tool, we briefly describe the main concepts used in normalization theory. More details can be found in textbooks such as [3, 12].

A functional dependency (FD) is a formula of the form

$$A \longrightarrow B$$

where A and B are non-empty sets of attributes (i.e., column names) of a table. For convenience, the attributes lying on the left (resp. right) hand side of the arrow is referred to the LHS (resp. RHS) of the FD. A FD is *trivial* if the RHS is a subset of the LHS.

Several normal forms are defined based on FDs, namely second normal form (2NF), third normal form (3NF) and Boyce-Codd normal form (BCNF). Specifically, a table is in BCNF if for every non-trivial FD, the LHS is a *superkey*, here a superkey means a set of attributes which functionally determine all other attributes in the table. A table is in 3NF if for every non-trivial FD, either the LHS is a superkey or the LHS is a *key attribute*, i.e., part of some *candidate key*, where a candidate key is defined as a minimal superkey. A table is in 2NF if there are no *partial dependencies*, which are FDs where the LHS is a proper subset of some candidate key, while the RHS is a non-key attribute. It follows that if a table is BCNF, it is also in 3NF; and if it is in 3NF, it is also in 2NF.

In order to normalize a table to tables of a particular normal form, one needs to decompose the tables into smaller tables, i.e., tables with fewer columns. Several textbooks promote the step-by-step approach, that is, normalize the table to 2NF tables, then to 3NF tables, and finally to BCNF tables. Each step is done by repeatedly (1) finding an offending FD $A \longrightarrow B$, and (2) decomposing the table into (A, B) and $R - B$, where R is the set of all attributes in the table. Using this approach to normalize a table into 3NF tables cannot guarantee the *FD-preservation* property. For example, for the table $R(a, b, c)$ with FDs $a \longrightarrow c$ and $b \longrightarrow c$, the approach may produce two tables (a, c) and (a, b), losing the FD $b \longrightarrow c$. Other algorithm exists that does have the property.

Often a set of FDs contains redundancies, e.g., trivial FDs, or FDs that can be implied by others, or FDs where the LHS contains unnecessary attributes. For normalization algorithms to work correctly, one often needs to find a *minimal cover*, which is a set of FDs equivalent to the original set, but does not contain redundancies.

Points of difficulty. Our experience shows that the concept of FD is quite easy to understand by all students. However, reasoning with FDs (i.e., finding redundant FDs), finding candidate keys, and the normalization process are among points where many students find difficult.

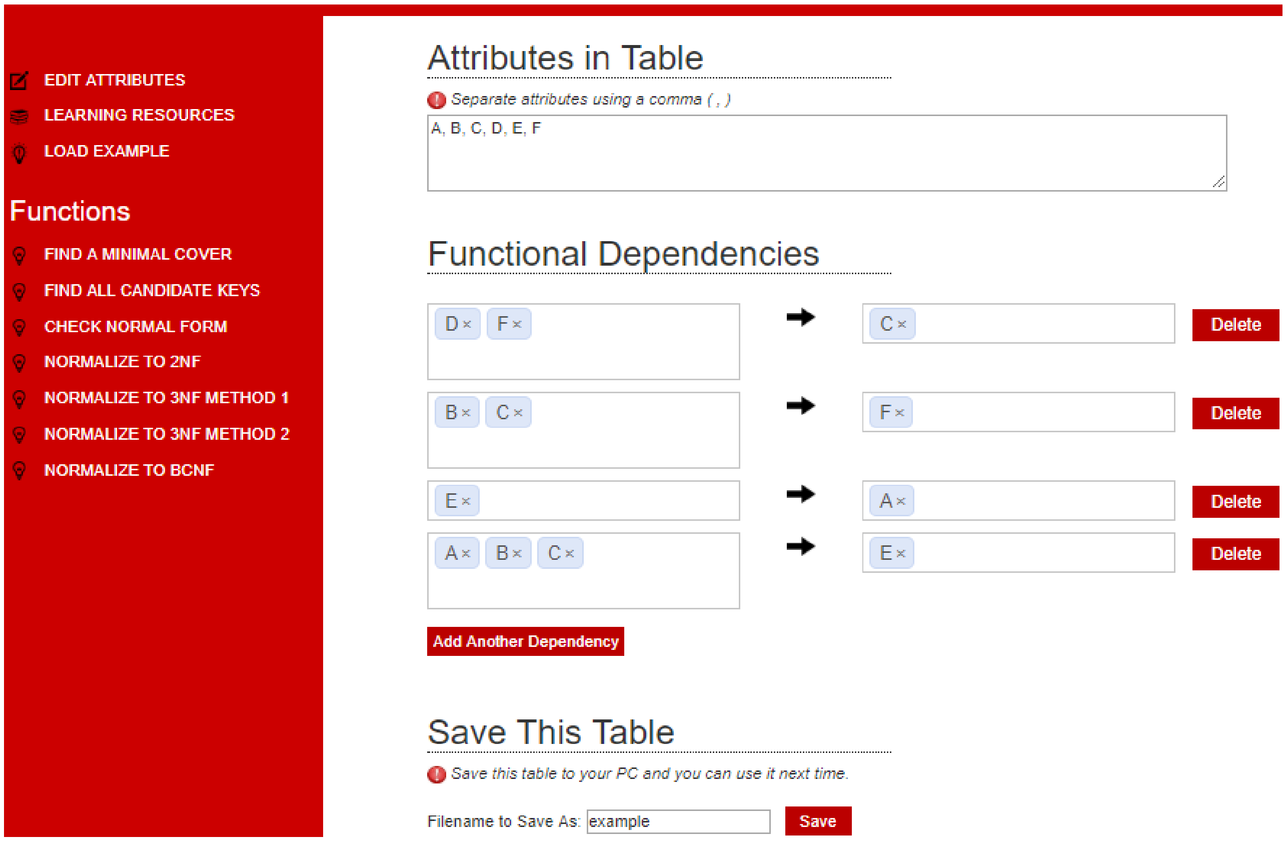
3 DESIGN AND IMPLEMENTATION OF THE TOOL

3.1 User Interface and System Functionalities

Our tool is a web-based system implemented with PHP at the server side, and JavaScript and HTML5 at the client side.

Figure 1 illustrates the main user interface. Here a user can edit the columns of a table, add/or delete a FD by typing in the LHS and RHS attributes (These attributes can only be chosen from those within the table), and click a button on the left side of the screen to complete a specific function. Clicking EDIT ATTRIBUTES will return the system to the main screen, allowing the user to change the attributes and FDs. The system also allows a user to choose from a list of pre-designed examples by clicking LOAD EXAMPLE. Here each example is carefully designed to demonstrate a specific point that is worth attention. At present, the list contains 20 examples, and we can easily insert more examples into the list if we want. If the user finds an example he tried is interesting and wants to keep the example for future reference, he can click SAVE THIS TABLE to save the example to his local computer. These locally saved examples can be uploaded by clicking LOAD EXAMPLE. Actually, clicking LOAD EXAMPLE will pop up a window which allows the user to choose from the list in the server, or from a local disk.

From the Functions menu on the left side on Figure 2, we can see the system include modules for finding minimal cover of the given FDs, finding all candidate keys, checking for normal forms, and normalizing the table to 2NF, 3NF, and BCNF. We implemented



Attributes in Table

Separate attributes using a comma (,)

A, B, C, D, E, F

Functional Dependencies

D × F × → C ×

B × C × → F ×

E × → A ×

A × B × C × → E ×

Save This Table

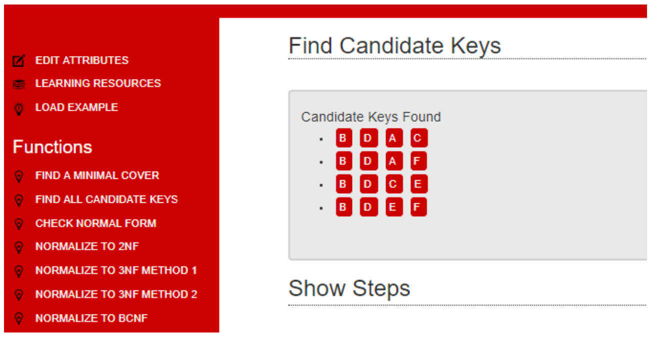
Save this table to your PC and you can use it next time.

Filename to Save As: example

Figure 1: Main user Interface

two algorithms for normalizing the table to 3NF, one has the FD-preservation property, the other does not (which is the step-by-step approach mentioned in Section 2).

Figure 2 through Figure 4 show the screen shots for the outputs of FINDING CANDIDATE KEYS, CHECK NORMAL FORM, and NORMALIZE to BCNF respectively. Each output screen contains a Show Steps button, and clicking the button will show detailed steps for obtaining the output result.

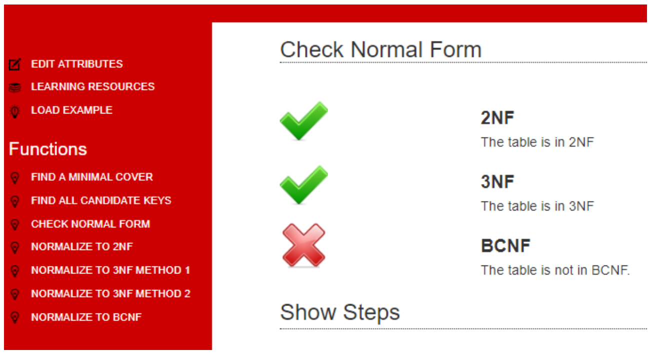


Find Candidate Keys

Candidate Keys Found

- B D A C
- B D A F
- B D C E
- B D E F

Figure 2: Screenshot for finding CKs



Check Normal Form

✓ 2NF The table is in 2NF

✓ 3NF The table is in 3NF

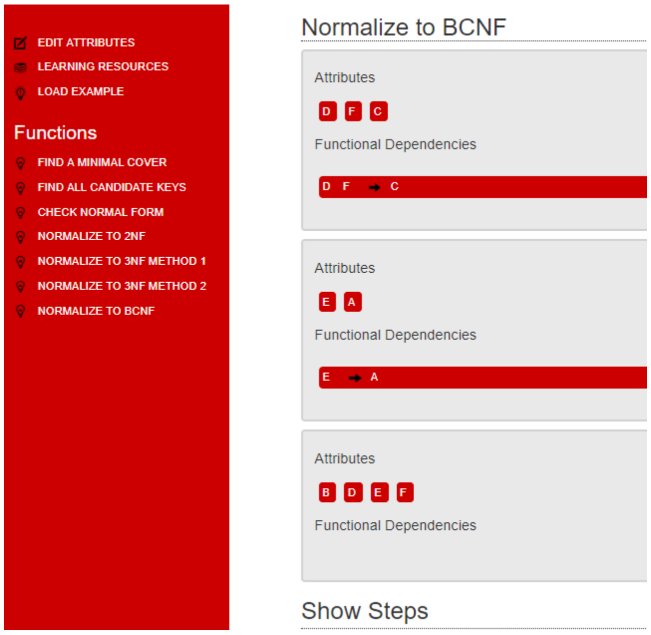
✗ BCNF The table is not in BCNF.

Figure 3: Screenshot for checking normal form

3.2 Efficiency considerations

With modern PCs and even mobile devices, efficiency of the system is not a concern for examples of reasonable size and instant response is almost guaranteed. However, since the system is designed to teach the students the step-by-step details of each functionality, we need to use the most efficient algorithms in practice.

In our implementation, we used a heuristically more efficient method for finding all candidate keys than those taught in the textbooks. All textbooks we have seen use brute-force methods that



Normalize to BCNF

Attributes
D F C
Functional Dependencies
D F → C

Attributes
E A
Functional Dependencies
E → A

Attributes
B D E F
Functional Dependencies

Figure 4: Screenshot for normalizing to BCNF

are equivalent to checking every subset of the table attributes one by one, and there are no guidelines on what attributes one should start with. Our more efficient method is based on the observation that (1) all attributes that do not appear on the right-hand-side (RHS) of any FD must appear in every candidate key, and (2) all attributes that appear on the RHS of some FD but not on the left-hand-side (LHS) of any FD cannot be part of any candidate key. Therefore, to find all candidate keys, we can start with those attributes that do not appear on the RHS of any FD, and we do not need to consider attributes that have appeared on the RHS of some FD but not on the LHS of any FD. The algorithm implemented in this tool is based on the above heuristic, and in fact, in most practical tables, it can help the user identify all candidate keys very quickly.

In addition, the *A Priori* principle for mining frequent item sets [13] is applied when finding all candidate keys and when finding the FDs that hold for the decomposed smaller tables, which allows us generate candidate attribute sets much more efficiently. However, such details are hidden and not shown by clicking the Show Steps button. This is to avoid distracting and confusing students who are not familiar with the technique.

4 APPLICATION AND EVALUATION OF THE TOOL

Our system was first trial-used in our *Data management* course in 2016 taught to the Bachelor of IT and Bachelor of CS students. As mentioned earlier, our students came from a large variety of backgrounds and they have very different levels of prior knowledge and learning capabilities, yet they were co-taught. All students were encouraged to use the system while learning FDs and normalization. At the end of the semester we conducted a voluntary anonymous survey. In addition to asking the number of hours they used the tool and any free text comments, we asked the students to choose one answer from *strongly agree, agree, neutral, disagree,*and *strongly disagree* to each of the following statements:

(1) The tool has helped me significantly in my learning
(2) I will recommend the tool to my friends who are learning FDs and normalization.

39 students responded, among them, 3 used the tool for less than 1 hour, 28 used it between 1 and 3 hours, and the rest stated that they used it for more than 3 hours. For Question 1 above, 22 students chose 'strongly agree', and 17 chose 'agree'. For Question 2, 19 students chose 'strongly agree', 18 chose 'agree', and 2 chose 'neutral'. Due to the anonymity of the user survey, we were unable to observe the performance impact on those who used the system.

An earlier version of the system was used in our 2015 Database Design course, which consists of 116 students from the Bachelors of IT and Master of IT programs. We asked the students to rank the system from 1 to 5 for its usefulness in helping them learn, as well as its user-friendliness. 50 students responded. Despite the fact that the show-steps function and load/save examples functions were unavailable in the earlier version, more than 85% of the respondents agree or strongly agree that the system is useful, and 100% agree that the system is user-friendly.

Our system has been used every year in our Database courses since then. It has also been used in several other institutions, and the feedback has been consistently positive.

5 CONCLUSION AND FUTURE WORK

In this report we shared our experience in developing a web-based tool to help students learn normalization and help lecturers teaching normalization. Our tool has several advantages over existing systems. We used the tool in our teaching and our user survey and feedbacks from lecturers show the tool is well liked by the students who actually used it, and it has helped them in their learning.

REFERENCES

[1] C. Begg and T. Connolly. 2014. *Database Systems: A Practical Approach to Design, Implementation, and Management (6e).* PEARSON.
[2] Hongbo Du and Laurent Wery. 1999. Micro: A normalization tool for relational database designers. *Journal of Network and Computer Applications* 22, 4 (1999), 215–232.
[3] R. Elmasri and S. B. Navathe. 2010. *Fundamentals of Database Systems (6e).* Addison-Wesley.
[4] O. Folorunso and A. Akinwale. 2010. Developing visualization support system for teaching/learning database normalization. *Campus-Wide Information Systems* 27, 1 (2010), 25–39.
[5] H. Fry, S. Ketteridge, and S. Marshall. 2009. *A Handbook for Teaching and Learning in Higher Education.* Routledge.
[6] Nikolay Georgiev. 2008. *A Web-Based Environment for Learning Normalization of Relational Database Schemata.* Master's thesis. Umea University.
[7] N. Hativa. 2001. *Teaching for Effective Learning in Higher Education.* Kluwer Academic Publishers.
[8] H. Kung and H. Tung. 2006. A web-based tool to enhance teaching/learning database normalization. In *SAIS 2006 Proceedings.* http://www.visible.com/Products/Analyst/vadbengineer.htm
[9] A. Mitrovic. 2001. NORMIT: a Web-Enabled Tutor for Database Normalization. Retrieved September 15, 2017 from http://ir.canterbury.ac.nz/bitstream/10092/335/1/12584595_normit.pdf
[10] G. C. Philip. 2007. Teaching Database Modeling and Design: Areas of Confusion and Helpful Hints. *Journal of Information Technology Education* 6 (2007), 481–497.
[11] A. Ramakrisnan. 2005. *Database Management Systems.* McGraw-Hill.
[12] A. Silberschatz, H. F. Korth, and S. Sudarshan. 2010. *Database Systems Concepts (6e).* McGraw-Hill.
[13] P. Tan, M. Steinbach, and V. Kumar. 2006. *Introduction to Data Minging.* Addison-Wesley.