

Web Based Learning Tool for Relational Database Design

First A. Harish Shakya¹, Second B. Mrs. Aruna Tiwari²

Shri G.S. Institute of Technology & Science/ Department of Computer Engg. Indore-452003, India

Email: harish.shakya@gmail.com

Shri G.S. Institute of Technology & Science/ Department of Computer Engg. Indore-452003, India

Email: atiwari@sgsits.ac.in

Abstract: A web based tool is designed and implemented. This tool is giving an interactive hands-on experience in relational database design such as database normalization process, check the decompositions during normalization that it is lossless or lossy and check for dependency preservation. It act as a tutor for learning relational database design concepts as well as design the information using relational model with insuring all sorts of anomalies. This paper presents the web based tool and its effectiveness in teaching relational data model. The effectiveness of the tool has been evaluated on the basis of number of test cases.

Index Term: Database Normalization, Relational Data Model, Functional Dependency, Normal Forms, Lossless & Lossy, and Preservation

I. INTRODUCTION

Database normalization is a well-developed field since the introduction of Codd's seminal work on normal forms in 1970. Bernstein (1976), Diederich and Milton (1988), Conception and Villafuerte (1990), and Rosenthal and Reiner (1994) proposed algorithms and tools to synthesize a normalized database using functional dependencies. 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. The poor performance of normalization indicated that teaching normalization is a challenge to IS/IT educators. The classical database normalization technique has often relied on the definition of normal forms. Some database textbooks include normalization algorithms to find the canonical cover by removing extraneous attributes of functional dependencies (FDs) and then to convert each FD in the canonical cover to a relation/table. A review of the literature reveals that during some years, number of books and papers are written based on the database design tool and numbers of tools are practically implemented through different-2 companies i.e. DB designer ,Case studio,Datanamic,DB zipper, Toad data modeler, smart draw tool.

DB designer allows you to construct your DB in an intuitive and easy to use environment, where you have a visual representation of the tables and relations contained in your project.

Case Studio is a highly professional and customizable database-modeling tool it visually creates and maintains Entity Relationship Diagrams (ERD) and Data Flow

Diagrams (DFD) and generates SQL scripts for various databases automatically.

DB Zipper is a database tool that processes table data and table structures from any database into a compressed database archive file (*.dbz). Toad™ Data Modeler is designed to visually create and maintain ERD (Entity Relationship Diagrams) for numerous databases.

Smart draw is a tool for design the graphical view of database and number of graphical view are already added into the tool and directly converted into Ppt, Pdf, ms-word, form etc.

The main objective of this paper is to describe the web-based tool for Relational database design concepts and its effectiveness in teaching and learning of database normalization, Section 2 illustrates basic concepts used in that web based tool, and section 3 describes various facilities of the web-based tool. The effectiveness of the web-based tool is evaluated and interpreted in section 4, and section 5 concludes the paper.

II. BASIC CONCEPTS

This tool is basically developed for learning purpose. It provides the learning of all fundamental concepts of relational database design; it enhances the teaching and learning of relational database design. The number of relational database design concepts such as

Find the Key with the help of clouser of attributes algorithm. Find the database Normalization: First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), Byco coded Normal Form (BCNF),Find the Candidate key with the help of canonical cover algorithms, Check the decomposition whether it is lossless or lossy and preservation dependencies or not.

The notations used for learning and applying all these things in our tool are described next. Let $R(x, y, z)$ is a relation schema with the group of attributes & functional dependencies (FD's) are represented as follows.

$R(x, y, z)$ R is a relation & R: group of attribute

Functional dependencies:

$x \rightarrow y$ if $(x \rightarrow R)$ x, y, and z: It is an attributes

$y \rightarrow z$ Then x is a key attribute.

If the key is finding with the help of clouser of attribute algorithm, then this key used in the database normalization algorithm, with the help of this key we will find the normal form. Some conditions are there for different-2 normal form, if any particular conditions are lies in given input functional dependencies and relation.

So we can define, given relation lie in that normal form (1NF, 2NF, 3NF, BCNF). Conditions are there:

A. First Normal form

In this normal form we will check the partial functional dependency. Partial Functional Dependency Indicates that if A and B are attributes of a table, B is partially dependent on A if there is some attribute that can be removed from A and yet the dependency still holds.

$R(a, b, c)$ ($ab \rightarrow R$) ab is a key. a, b is prime attribute, Prime attribute: It is a part of key's attribute.

B. Second Normal Form

A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key. In this normal form lie condition is full functional dependency.e.g.

$R(a, b, c, d)$ ($ab \rightarrow R$) ab is a key.

Nonkey \rightarrow Nonkey means a & b is not present in both side of FD's. ($c \rightarrow d$) Both are non-prime attributes. It is full functional dependency.

In some conditions 3NF and BCNF is differ so we can define it. A Relation R is said to be in BCNF if whenever $x \rightarrow A$ holds in R, and A is not in x, then x is a candidate key for R.

It should be noted that most relation that are in 3NF are also in BCNF infrequently a 3NF relation is not in BCNF and this happen only if....

- The candidate key in the relation are composite keys(i.e. they are not in single attribute)
- There is more than one Candidate key in the relation, and
- The keys are not disjoint, that is some attribute in the keys are come.

C. Third Normal Form

An entity is in the third normal form if it is in the second normal form and all of its attributes are not transitively dependent on the primary key. Transitive dependence means that descriptor key attributes dependent not only on the whole primary key, but also on other descriptor key attributes that, in turn, depend on the primary key. In this normal form the condition is transitive dependency.

$R(a, b, c)$ $a \rightarrow R$ a is key attribute.

$a \rightarrow b, b \rightarrow c$ c is transitive dependent on attribute a.

Non-key elements are transitive dependent on the key element then we can call 3NF.

D. Boyce- Codd Normal Form (BCNF)

In this normal form lie condition is non key attribute directly dependent on candidate key.

$R(a, b, c)$ ($x \rightarrow R$) x is key attributes.

Candidate key \rightarrow nonkey

$x \rightarrow a$ x is composite key. (Group of attributes)

BCNF follow some conditions:

- x contains a or
- x is super key

Apart from checking against various normal forms, the notations which are decomposed due to the normalization must also further being checked that whether the

decomposed relations are satisfying lossless property and preserving dependencies.

E. Check the Decomposition

In Lossless and lossy: In this concept we have to check the relations are lossless or lossy after decomposition.e.g A schema $R(a,b,c,d,e)$ the decomposed relations are $R1(a,b,c)$ & $R2(d,e)$ the condition is

$R(a, b, c, d, e)$ $R1(a, b, c)$ $R2(d, e)$

$R1 \cap R2 = \emptyset$, then relations are Lossless

$R1 \cap R2 \neq \emptyset$, then relations are Lossy

Dependency Preservation: In this concept we have to check the given Fd's is preserve or not, means after decompose a relation preservation of Fd's is valid or not.

$R(a, b, c, d, e)$

$R1(a, b, c)$ $R2(d, e)$

Functional Dependencies's:

$a \rightarrow b$ Dependency Preserve

$b \rightarrow c$ Dependency Preserve

$c \rightarrow d$ Dependency is not preserve

Find the Candidate key with the help of Canonical Cover Algorithm: In this algorithm, work is divided into two main parts. Find the Non redundant cover and Find the Extraneous attribute.

In first part we have to remove the redundant FD's, after this next step is removing the extraneous attributes. Then remaining FD's used and find the candidate key (Single key) with the help of "Clouser of attribute" algorithm. These are all basic concepts of relational database. These are main feature of that tool, so basically we can call that is Relational Database Design Tool.

III. THE WEB BASED TOOL

It has been noticed that the decompositions happened when attributes on the right-hand side of functional dependencies have more than one copy, and the number of the decomposed relations is exactly the same as the number of Functional dependencies. To decompose a relation into the third normal form, one simply eliminates extraneous attributes on the right-hand side of the functional dependencies. The following steps describe the normalization process by using the example of a universal table T and a set of FDs: FD_1 FD_2 and FD_3 . The universal relation T (A, B, C, D, E, F)

$FD_1: A \rightarrow B, C, D$

$FD_2: B \rightarrow C, D$

$FD_3: A, E \rightarrow B, C, D, F$

1. All functional dependencies on the left-hand side must keep their attributes intact.

2. Extraneous attributes on the right-hand side should be eliminated. This step is to eliminate partial dependencies and transitive dependencies. Repeated right-hand side attributes should be identified in all the functional dependencies, one copy of the redundant attributes should be kept and the others should be deleted. The rules of thumb about which copy of attributes to keep are

The attributes that have fewer numbers of attributes on the left-hand side should be kept (this step will eliminate partial dependency), Example: Attributes B, C, and D

depend on part of the whole key (attribute A). Attributes B, C, and D in functional dependency FD_3 will be deleted, since attributes B, C, and D appear in FD_1 and FD_1 has only one attribute on the left-hand side. A new functional dependency FD_3 : $A, E \rightarrow F$ is formed.

When two FD_s have the same number of attributes on the left-hand side, the attributes that have the fewer numbers of attributes on the right-hand side should be kept (this step will eliminate transitive dependency). Example: Attributes C and D appear in functional dependencies FD_1 and FD_2 . Attributes C and D are transitively dependent on Attribute A. Attributes C and D will be deleted from functional dependency FD_1 , since FD_1 has more right-hand side attributes than functional dependency FD_2 . A new functional dependency FD'_1 : $A \rightarrow B$ is formed.

3. Construct relations. This step is to convert the functional dependencies without extraneous right-hand side attributes to relations. The new functional dependencies are as followings:

- FD'_1 : $A \rightarrow B$
- FD_2 : $B \rightarrow C, D$
- FD'_3 : $A, E \rightarrow F$

The final normalized relations are exactly the same as the results of the decomposition algorithm and SA&D normalization techniques.

- T_1 (A, B)
- T_2 (B, C, D)
- T_3 (A, E, F)

The web-based tool is developed using a Java, JSP, Servlet and Html based on the above Normalization technique. The tool has the following features:

Main Window: The main window contains fields to key in and display functional dependencies and buttons to submit functional dependency and relation at a time, to reset/clear the memory for another exercise/practice.

Result window: The user will see the normalized result in the normalized relations window. The tool will normalize the database based on the functional dependencies and the relation the user submitted. In result window show the number of contents means first of all given input relation & functional dependencies and output portion is input relation, key and super key block and relation normal form block that is define the given input relation and functional dependencies lies which particular normal form.

IV. EXPERIMENTAL WORK

The tool developed is tested with different test cases for checking the evaluation of candidate key identifies the normal form, check the lossless or lossy and check the dependencies preservation. It is found that it is working for any number of attribute set contained in schema with upto five total FD 's. In this tool we have given two types of input: Relation and Functional dependencies, calculating the keys and normal form but some restriction for input blocks of FD 's:

- Left part of the FD 's is restricted for single input attribute.

- We can not give two same types of FD 's.
- We can not give same left hand part of the FD 's.

Main advantages are all possible keys are evaluated so we can divide into two parts primary key and super keys. We will use only primary key. After normalization we can check decomposition is lossless & lossy and dependencies are preserving or not, in this tool have some restriction for check decomposition is lossless or lossy means more than two relations we can not check relations is lossless or lossy. These tools support at most two relations. The application does not crash under most of cases. The system work properly into satisfy able, we can prove it with the help of number of test cases:-

Table 1: Number of test cases

No. of input	INPUT		OUTPUT	
	Relation	Functional dependencies	Key	Normal Form
1	abcd	$a \rightarrow bcd, c \rightarrow a$	a	BCNF
2	abcd	$a \rightarrow bcd, c \rightarrow d$	a	2NF
3	abcd	$a \rightarrow b, b \rightarrow d$	ac	2NF

V. CONCLUSION AND FUTURE RESEARCH DIRECTION

A web-based normalization tool has been developed to enhance teaching and learning of database normalization. The tool is user friendly and can be accessed through the Internet. The tool resolve few basic design issues for designing the good database such as normalizing it to appropriate normal form helps in designing the candidate keys, check the lossless or lossy and dependencies preservation. Students' responses to the tool were mostly favorable. The tool has a positive impact on students' perceptions.

Currently, the web-based tool allows a set of 5 functional dependencies, which is adequate for teaching purposes. More features, e.g., 'Load', 'Save', and 'Print' are still under development. The tool can be expanded for check the multivalued dependencies concepts.

REFERENCES

- [1] Hsiang-Jui Kunag, Hui-Lien Tung "a web-based tool to enhance teaching/learning database normalization" 2006 Southern Association for Information Systems Conference pp 251.
- [2] Fagin, R. (1981), "A Normal Form for Relational Databases that is Based on Domains and Keys", ACM-TODS, Vol 6, No 3, Sept 1981, pp 387-415.
- [3] Bernstein, P. A. 1976. Synthesizing Third Normal Form Relations from Functional Dependencies, *ACM Transactions on Database Systems*, Vol. 1. No. 4, pp 277-298.
- [4] Silberschatz, A., Korth, H. F., and Sudarshan, S., 1996, *Database System Concepts*, 4th Edition, Boston, MA: McGraw Hill, pp 217-231.
- [5] R. Elmasri, S.B. Navathe, *Fundamentals of database system*. Benjamin/Cummings, redwood, 1994, pp 254-277.