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