Available online at www.sciencedirect.com
ScienceDirect
Information Systems 32 (2007) 773–792
www.elsevier.com/locate/infosys

An automated entity–relationship clustering algorithm for conceptual database design

Madjid Tavana[a,*], Prafulla Joglekar[a], Michael A. Redmond[b]

[a]*Management Department, La Salle University, Philadelphia, PA 19141, USA*
[b]*Department of Mathematics and Computer Science, La Salle University, Philadelphia, PA 19141, USA*

Received 31 December 2005; received in revised form 6 July 2006; accepted 11 July 2006
Recommended by F. Carino Jr.

Abstract

Entity–relationship (ER) modeling is a widely accepted technique for conceptual database design. However, the complexities inherent in large ER diagrams have restricted the effectiveness of their use in practice. It is often difficult for end-users, or even for well-trained database engineers and designers, to fully understand and properly manage large ER diagrams. Hence, to improve their understandability and manageability, large ER diagrams need to be decomposed into smaller modules by clustering closely related entities and relationships. Previous researchers have proposed many manual and semi-automatic approaches for such clustering. However, most of them call for intuitive and subjective judgment from "experts" at various stages of their implementation. We present a fully automated algorithm that eliminates the need for subjective human judgment. In addition to improving their understandability and manageability, an automated algorithm facilitates the re-clustering of ER diagrams as they undergo many changes during their design, development, and maintenance phases.

The validation methodology used in this study considers a set of both objective and subjective criteria for comparison. We adopted several concepts and metrics from machine-part clustering in cellular manufacturing (CM) while exploiting some of the characteristics of ER diagrams that are different from typical CM situations. Our algorithm uses well established criteria for good ER clustering solutions. These criteria were also validated by a group of expert database engineers and designers at NASA. An objective assessment of sample problems shows that our algorithm produces solutions with a higher degree of modularity and better goodness of fit compared with solutions produced by two commonly used alternative algorithms. A subjective assessment of sample problems by our expert database engineers and designers also found our solutions preferable to those produced by the two alternative algorithms.

Keywords: Computers; Databases; Information systems; Analysis and designs; Planning; Decision analysis

1. Introduction

Entity–relationship (ER) modeling [1] is a popular and effective methodology used to construct a *conceptual* data model. An ER diagram (ERD) is a detailed graphical representation of the data

*Corresponding author. Tel.: +1 215 951 1129; fax: +1 267 295 2854.
E-mail addresses: tavana@lasalle.edu (M. Tavana), joglekar@lasalle.edu (P. Joglekar), redmond@lasalle.edu (M.A. Redmond).
URL: http://lasalle.edu/~tavana.

0306-4379/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2006.07.001

requirements in an organization or business unit. ERDs enhance understanding of the system and improve communication among database engineers, designers, and end-users.

Consider the following business scenario documented during systems analysis for the development of an information system for a retailer of custom-made products:

> A **customer** issues a **purchase order** to the **vendor** (retailer) to buy a **product**. The purchase order consists of an **order item**. When the **product** is delivered, the **customer** is given a **receipt** for the **product**. Because the **customer** has a **line of credit**, the actual payment is deferred until later. In the next billing cycle, the **vendor** issues an **invoice** that includes an **invoice item** related to the **order item** on the **purchase order**. Upon receiving the **invoice**, the **customer** makes **sales payment** against the **receipt** of the **product** and the **vendor** receives **vendor payment** for the **invoice**.

This scenario results in the identification of the entities and relationships presented in a clustered ERD presented in Fig. 1a. Entities are represented by rectangles, relationships by diamond-shaped boxes, and connecting lines show which entities participate in which relationship [1]. For example, the fact that a customer (Entity A) buys a product (Entity D) is represented by the "buy" relationship (Relationship 1). Knowledgeable end-users can comprehend and validate the database design implied by an ERD against actual business practices. Once finalized, an ERD serves as the blueprint for database implementation.

The ER model has become so common in database design that today a number of commercial ER diagraming tools are available (e.g. ERWin by Computer Associates, EasyER by Visible Systems, Visio by Microsoft, and ER/1 by Embarcadero). ER tools differ somewhat in their terminology and notation. However, the basic concepts are the same and all ER tools represent data requirements graphically. Several tools generate the code needed for the database schema, including the necessary tables, indexes, triggers, and stored procedures. Most ER tools support systems analysis and database design, implementation, and maintenance.

Yet, today ER tools fall short of their true potential. This is because ER diagrams are rarely as small as the one presented in Fig. 1a. A typical application data model consists of 95 entities and an average enterprise model consists of 536 entities [2]. Feldman and Miller [3] suggest that diagrams involving 30 or more entities exceed the limits of easy comprehension, and communications. To improve their understandability and manageability, large ER diagrams need to be decomposed into smaller modules by clustering closely related entities and relationships.

The ERD in Fig. 1a is small enough to comprehend without any decomposition. However, the three clusters (retailer's "sales," "accounts receivable," and "accounts payable" subsystems) identified by our algorithm illustrate some of the advantages of decomposing ERDs. The literature on clustering [3–6] has identified several advantages of ERD decomposition. Clustered ER diagrams:

1. Are easier to develop and maintain, since they are more modular;
2. Are easier to document, since clusters are smaller and easier to understand;
3. Are easier to validate, since they provide better organization;
4. Can assist project management by allowing allocation of modular tasks to individuals or teams; and
5. Can assist identification of reusable subsystems that can be added, removed, or modified relatively independently of one another.

In spite of these advantages, today no ER diagramming tool offers any clustering assistance. This is mainly because the available ER clustering algorithms call for intuitive and subjective judgment from "experts" at various stages of their implementation (see Section 2). ER tools support data model construction, communication, and validation by storing all the entities, relationships, relevant assumptions, and constraints in a repository. Using the repository, multiple conceptual and physical-level ERD "views" can be produced for specific purposes such as end-user communication, database design, and development by presenting only relevant portions of the larger design to specific audiences. Users often prefer such views. As such, we do not want to suggest that clustered diagrams would replace functional views of a database.

However, sometimes even these views are too large for adequate comprehension. More importantly, because specific entities and relationships are often duplicated in several views, the "views" do not offer some of the advantages of a clustered ERD. For example, in allocating responsibility for the