Automated data modeling is a key component of modern Computer-Aided Software Engineering (CASE) tools. Systems analysts use CASE tools to model business processes and data in using a rapid application development methodology for software development. A key component of the development process is the conversion of entity-relationship diagrams to normalized relational tables. Designers rely on CASE tools to generate normalized table structures; however, a generated relational schema is only as good as the data model diagram upon which it is based. This article outlines situations where errors in entity-relationship models can result in non-normalized table structures.

The modern approach to structured systems analysis and design involves the use of systems analyst workbench technology for computer-aided software engineering (CASE). Active-in-development CASE tools provide systems analysts a graphical user interface for building process and data model diagrams. Ultimately, the conceptual data model diagrams are converted to physical relational database schemas that incorporate data and referential integrity constraints. The process model diagrams are used to generate screens, reports, menus, and computer programming code to implement business processes that maintain the data that are stored in the relational tables.

Many CASE tools incorporate a data modeling approach based on Chen’s (1976) Entity-Relationship (E-R) model. The automatic conversion of an E-R diagram to a relational database schema through the use of CASE technology provides an extremely productive systems development environment. Most CASE tools advertise the ability to produce a normalized schema, at least to third normal form; however, the extent to which the generated schema is actually normalized is a function of the accuracy of the E-R diagram. As Ling (1985) points out, “it is very difficult to determine whether an E-R diagram is the best representation for a given database.”

Experts in E-R modeling will readily admit that data modeling is more an art than a science. Although the transformation of an E-R diagram to a relational schema follows a set of well-defined, straightforward rules, errors in an E-R diagram can lead to normalization problems which the transformation rules fail to capture.

Our objective here is to examine different types of E-R modeling errors in order to understand when they arise and how to avoid them. To facilitate our examination, we adopt intuitive definitions for the various normal forms based on Kent’s definitions in his classic article, A Simple Guide to Five Normal Forms in Relational Database Theory (1983). Kent provides detail guidance on record (table) design. His definitions of the various normal forms will aid the reader in understanding how errors creep into table structures during the transformation of E-R diagrams to relational schema. Readers seeking additional guidance on normalization are referred to Date (1995) who provides what is probably the greatest depth of coverage of normalization theory in a single volume, and to the original article on the relational model by Codd (1970). Further general guidance and detailed steps for converting an E-R diagram to normal form E-R diagrams are given by Ling (1985) who provides a comprehensive algorithm for the transformation process.

Types of Errors

In general, there are two classes of E-R modeling errors that lead to normalization problems: (1) the “incomplete data model” error, and (2) the “mis-modeled problem domain”
error. The incomplete data model error tends to occur in situations where the systems analysts is tasked to build a computer-based information system that is limited in scope. A key objective for successful information system project management is the definition of a limited, yet adequate project scope—a scope that enables the production of system deliverables within a reasonable time period. Limiting a project’s scope often results in information systems that are based on limited data models. Limited information systems are fairly common throughout the IS world where dissimilar technologies prevent data sharing and work against the concept of a shared, enterprise-wide database.

The mis-modeled problem domain error is actually a class of errors including those that arise whenever systems analysts lack a complete understanding of the problem domain. These include errors such as depicting an attribute as single-valued when, in fact, the attribute is multivalued, or depicting a single entity which includes attributes that should be assigned to two separate entities, or mis-modeling the connectivity or degree of a relationship.

In order to portray the errors associated with the various normal forms, we use an example conceptual model that is based on the university modeling problem given in Figure 1. This example is used, in some form or another, in most textbooks on database management systems, and offers the potential for violations of all normal form definitions. While studying the errors depicted below, keep in mind that this university modeling problem is well-known, and, in fact, it is highly unlikely that an experienced database designer would make the mistakes depicted. However, even expert database designers can make the type of mistakes depicted below when they are thrust into situations that require modeling problem domains in which they lack experience, and inexperienced database designers are more prone to make errors.

Normal Form Errors

First Normal Form (1NF)

Kent’s definition for 1NF, “all occurrences of a record type must contain the same number of fields,” is meant to exclude variable repeating fields and groups. A common data modeling error is failing to recognize when a specific attribute of an entity is multivalued. In Figure 1, the STUDENT entity includes the Major field of study attribute. While many textbooks depict this attribute as single-valued, Major is actually a multivalued repeating field since a small number of students may elect to double-major while attending college. This is an example of a mis-modeled problem domain error. That portion of Figure 1 related to the STUDENT entity would yield the erroneous table structure given below. Note that the

![Figure 1: Example Conceptual Model of a University](image-url)