Entity-Relationship Modeling and Normalization Errors

Douglas B. Bock
Southern Illinois University at Edwardsville

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, straight-forward 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"

Manuscript originally submitted October 20, 1995; Revised April 11, 1996; Accepted April 19, 1996 for publication.

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

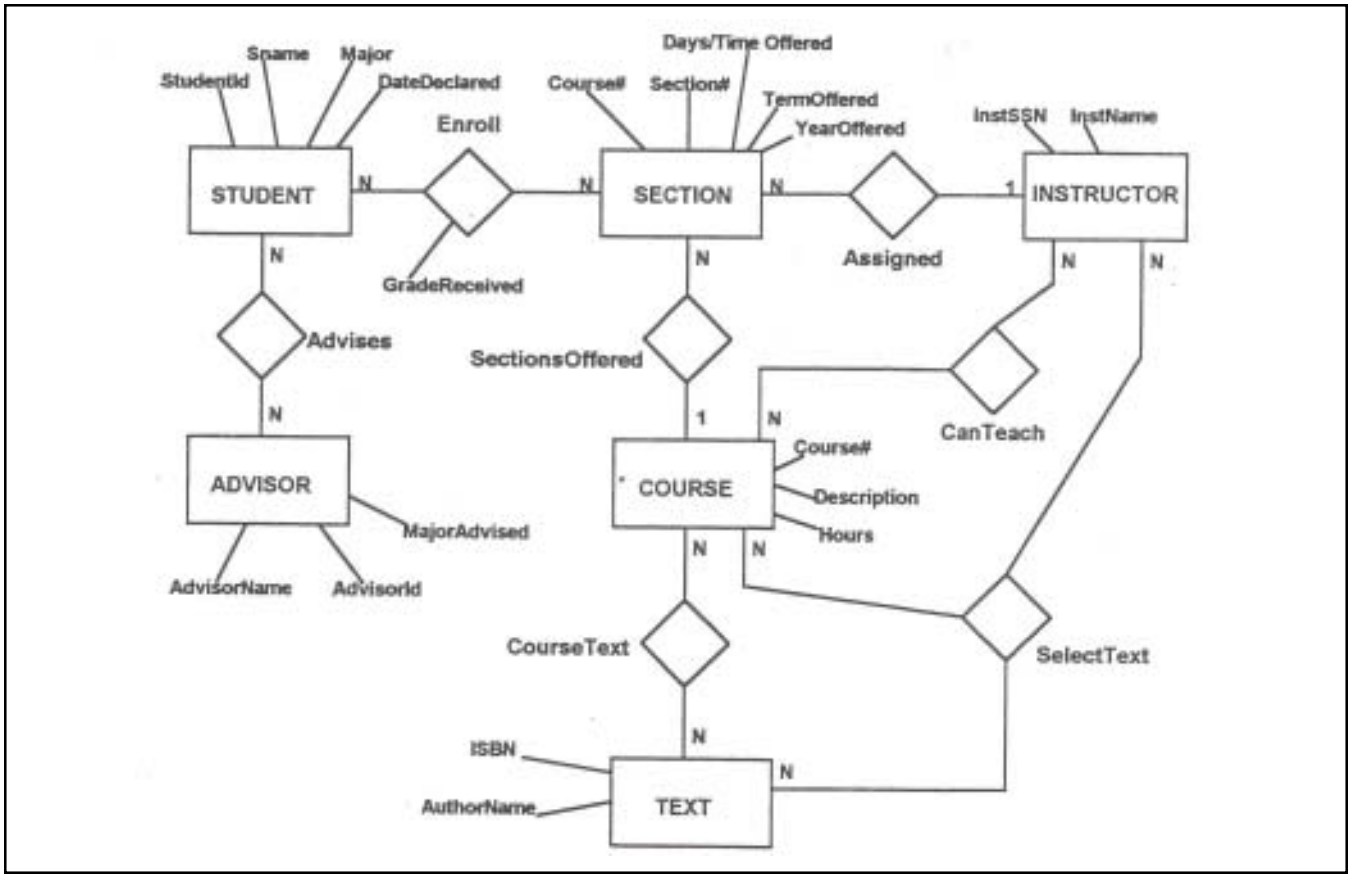

Figure 1: Example Conceptual Model of a University

primary key field(s) is indicated by underlining.

STUDENT (StudentId, SName, Major)

If the systems analyst recognizes that *Major* is multivalued, then a simple transformation rule is applied — the repeating field is removed to a second table along with the primary key field from the STUDENT table to form a composite key in the new table. The proper schema definition yields two tables.

STUDENT (StudentId, Sname)
STU_MAJ (StudentId, Major)

Extending this data modeling problem, there may be situations where repeating fields are related, as is the case where the university tracks not only each student's major, but also the date the major was declared (*DateDeclared* attribute). Again, an erroneous E-R model would yield a single table structure as given below.

STUDENT (StudentId, Sname, Major, DateDeclared)

Normalizing this table requires the systems analyst to understand which fields determine which other fields. Here, the combination of *StudentId* and *Major* uniquely determines values for the DateDeclared field. Note that the combination of *StudentId* and *DateDeclared* do not determine *Major* since a student may declare two majors on the same date. Again, the proper schema definition yields two tables. The solution procedure is easily extended to several sets of repeating fields with each set of repeating fields requiring an additional table structure.
STUDENT (StudentId, Sname)
STU_MAJ (StudentId, Major, DateDeclared)

Second Normal Form (2NF)

Kent combines the definition for 2NF with that for third normal form (3NF), "a nonkey field must provide a fact about the key, the whole key, and nothing but the key." Additionally, a table must satisfy 1NF. Violations of 2NF occur when a table's primary key is a composite of two or more attributes, and the table contains a non-key field that is not fully determined by the composite primary key, thus, a *non-key field is not a fact about the whole key*.

E-R modeling errors related to 2NF arise whenever the E-R diagram represents a restricted or incomplete view of the problem domain, or a grossly mis-modeled E-R diagram. This error is most likely to occur when a binary many-to-many relationship has been mis-modeled, because this type of relationship yields table structures with composite keys. This type of error can also occur for ternary and higher order relationships, and can also occur where the systems analyst has modeled the relationship as a *gerund*, i.e., an abstract object which is at the same time an entity and a relationship. Ternary and higher order relationships are covered later in this article.

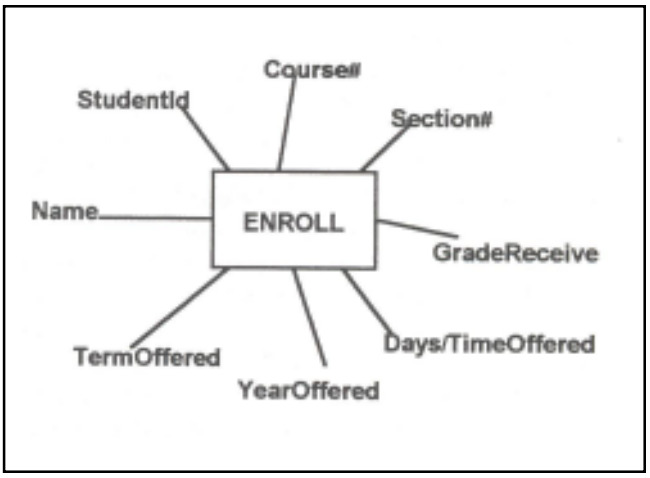

Figure 2: Second Normal Form Modeling Error

Consider the enrollment of students in sections of a course as is depicted by the Enroll relationship in Figure 1. Figure 2 gives a restricted model of enrollment data where the systems analysts has developed a view of the problem domain that consists of a single entity named ENROLL. This model fails to recognize that the STUDENT and SECTION entities exist. The transformation of Figure 2 to a relational schema yields a single table.

ENROLL (StudentId, Course#, Section#, TermOffered,
YearOffered, SName, Grade, Days/TimeOffered)

The table violates 2NF since the attribute *StudentId* determines Sname while the composite of *Course#*, *Section#*, *TermOffered*, and *YearOffered* determine *Days/TimeOffered*. Only the attribute *Grade* is fully, functionally dependent on the primary key. A complete view of the problem domain as shown in Figure 1 would yield the same three table structures that the application of normalization theory will produce by applying the simple transformation rules for transforming a binary many-to-many relationship to a relational schema.

STUDENT (StudentId, Sname)
ENROLL (StudentId, Course#, Section#, TermOffered,
YearOffered, Grade)
SECTION (Course#, Section#, TermOffered, YearOffered,
Days/TimeOffered)

Third Normal Form (3NF)

Data maintenance problems associated with 3NF arise only when a table contains two or more nonkey fields, and one of these nonkey fields determines another nonkey field, or as

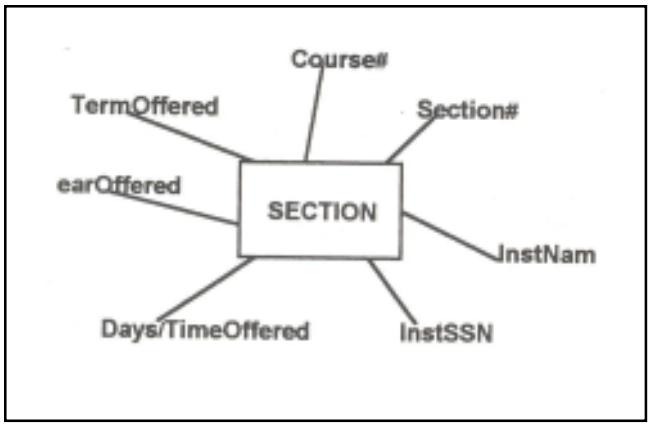

Figure 3: Third Normal Form Modeling Error

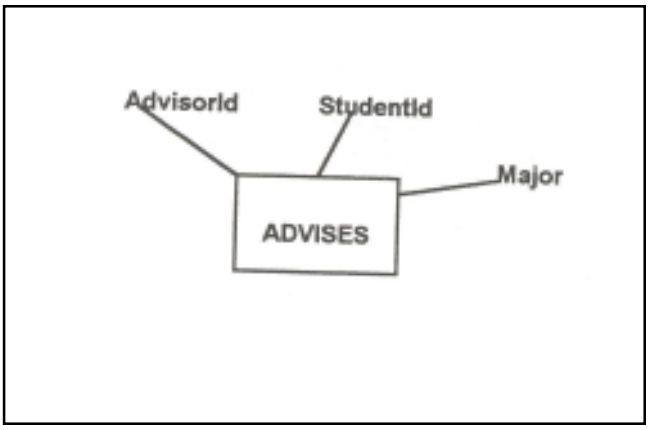

Figure 4: Boyce-Codd Normal Form Modeling Error

Kent states, "a nonkey field is a fact about another nonkey field." E-R diagrams that fail to capture information properly about binary one-to-many relationships can result in table structures that violate third normal form. This is similar to the second normal form problem in that the E-R diagram represents a restricted or incomplete view of the problem domain. The typical error here is for the systems analyst to model a one-to-many binary relationship between two entities as a single entity.

Figure 1 gives a one-to-many relationship named Assigned that relates occurrences of the SECTION and INSTRUCTOR entities. Figure 3 gives an incorrect model of the relationship where all attributes are attached to the SECTION entity, and the INSTRUCTOR entity is not modeled. A transformation of Figure 3 yields the table:

SECTION (Course#, Section#, TermOffered, YearOffered,
Days/TimeOffered, InstSSN, InstName)

The table violates 3NF since the attribute InstSSN determines InstName, and neither field is key. The correct E-R diagram shown in Figure 1 yields two tables with the InstSSN field in the SECTION table serving as a foreign key link.

SECTION (Course#, Section#, TermOffered, YearOffered,
Days/TimeOffered, InstSSN)
INSTRUCTOR (InstSSN, InstName)

Boyce-Codd Normal Form (BCNF)

BCNF is often referred to as a more stringent definition of third normal form. In fact, Kent's intuitive definition of second normal form and third normal form also encompasses BCNF. It is also helpful to examine the formal definition of BCNF in order to differentiate this more stringent definition of 3NF from Codd's original definition of 3NF. The formal definition can be found in any textbook on the topic: a table is in BCNF if every determinant is a candidate key. This rather non-intuitive definition simply means that data maintenance problems can arise whenever a table has more than one field or combination of fields that can serve as the key identifier for the table.

Figure 4 depicts a model which violates BCNF. This particular modeling problem is found in both McFadden and Hoffer (1994) as well as Kroenke (1995). The model in Figure 4 gives a restricted view of what should be modeled as a binary many-to-many relationship named Advises that links occurrences of the STUDENT and ADVISOR entities (see Figure 1 for the correct E-R diagram for the Advises relationship). Additionally, the model given in Figure 4 fails to recognize that the *Major* attribute is multivalued.

The relational transformation of Figure 4 yields a single table structure with *StudentId* and *Major* as a composite primary key since this combination determines the *AdvisorId* attribute.

ADVISES (StudentId, Major, AdvisorId)

Assuming that advisors provide academic guidance for only a single major, there are two functional dependencies:

StudentId + Major → *AdvisorId*
AdvisorId → *Major*

Note that no single attribute determines the other two attributes; further, the selection of *StudentID + Major* as the primary key for the table still results in potential data maintenance problems (see McFadden & Hoffer for a more detailed explanation). BCNF is achieved by moving the attribute that is a determinant, but not a candidate key to a separate table; in this case, *AdvisorId.*

ADVISES (StudentId, AdvisorId)
ADVISOR (AdvisorId, Major)

It is interesting to compare the above solution to that obtained by applying standard transformation rules to the

Advises binary many-to-many relationship shown in Figure 1. The transformation of Advises and the associated entities in Figure 1 yields four relational tables as given below. Note that this solution subsumes the above table structures, and recognizes that the Major and DateDeclared attributes for the STUDENT entity are multivalued (see the section on 1NF above). Further, the BCNF problem is avoided when the data model is complete with respect to this relationship.

STUDENT (StudentId, Sname)
STU_MAJ (StudentId, Major, DateDeclared)
ADVISES (StudentId, AdvisorId)
ADVISOR (AdvisorId, AdvisorName, MajorAdvised)

Fourth Normal Form (4NF)

To this point we have examined data modeling errors that have dealt with, at most, two entities. 4NF errors are more rare than those discussed above because they result from situations where a relational table structure includes attributes drawn from three different entities, and where the facts stored in the table represent multivalued dependencies between these attributes. Kent's definition of 4NF focuses on the multivalued dependencies: *a record type should not contain two or more independent multivalued facts about an entity*, in addition to satisfying lower normal form definitions. Consider the following example table structure from McFadden and Hoffer (1994, page 224).

SELECT_TEXT (Course#, InstSSN, ISBN)

This table attempts to capture information about the textbooks used in courses by instructors. By itself, the table gives a very restricted view of reality. Figure 1 provides an unrestricted view of the SelectText relationship which is actually a ternary many-to-many-to-many relationship among the INSTRUCTOR, COURSE, and TEXT entities. This relationship properly models the situation where a course may have several instructors, each course uses several textbooks, and the texts used for a given course are selected by the instructor. In this situation the relationship is a ternary one, and the SELECT_TEXT table structure correctly models the relationship even though its restricted view does not depict the associated base tables that would store the detailed information about courses, instructors, and textbooks. Nonetheless, no data maintenance anomalies exist for SELECT_TEXT.

Suppose, however, that the situation is changed. What if there is no relationship between the text used for a course and the instructor who teaches the course, i.e., the instructor doesn't select the text; rather, the instructor uses whatever text is specified by the academic department offering the course. In this situation, there are two multivalued dependencies among the data elements, but no determinants, and the InstSSN and ISBN attributes are independent:

Course# ->> InstSSN
Course# ->> ISBN
InstSSN <—/—>> ISBN

For each Course# , there is a well-defined set of instructors who teach the course and a well-defined set of textbooks used for the course. Modeling this as a ternary relationship, as is done with the SELECT_TEXT table, can lead to difficulties in updating the table as depicted in Table 1a, and represents a mis-modeled problem domain error. Here three instructors teach MNGT 444. A departmental decision to add a textbook with ISBN 0-9999-9999-9 to the Management 444 course requires adding three rows to the table, one for each instructor, because the textbook is used by all instructors who teach Management 444. Adding a row for only one of the Management 444 instructors would imply that there is a relationship between instructor and textbook when no such relationship exists.

The SELECT_TEXT table is converted to 4NF by dividing the table into the two new tables given below and depicted in Table 1b. These two new tables recognize the

Course#	InstSSN	ISBN
IS 402	444-44-4444	0-1111-1111-1
IS 402	555-55-5555	0-1111-1111-1
MNGT 444	111-11-1111	0-2222-2222-2
MNGT 444	222-22-2222	0-2222-2222-2
MNGT 444	333-33-3333	0-2222-2222-2
MNGT 444	111-11-1111	0-9999-9999-9
MNGT 444	222-22-2222	0-9999-9999-9
MNGT 444	333-33-3333	0-9999-9999-9

Table 1a.

TABLE_A (Course#, InstSSN)

Course#	InstSSN
IS 402	444-44-4444
IS 402	555-55-5555
MNGT 444	111-11-1111
MNGT 444	222-22-2222
MNGT 444	333-33-3333

TABLE_B (Course#, ISBN)

Course#	ISBN
IS 402	0-1111-1111-1
MNGT 444	0-2222-2222-2
MNGT 444	0-9999-9999-9

Table 1b.

Table 1: Example SELECT_TEXT Table Data

independence between instructors and textbooks. Note that this is very similar to the situation in 1NF, except here there are two independent multivalued facts about courses.

TABLE_A (Course#, InstSSN)
TABLE_B (Course#, ISBN)

Interestingly, a more complete E-R model that captures the data relationships properly avoids the potential 4NF problem. The proper E-R model for the situation where INSTRUCTOR and TEXT are independent is depicted in Figure 1 by two relationships, CanTeach and CourseText. A transformation of the CanTeach and CourseText binary relationships and their associated base tables yields the following relational table schema. Note that the CAN_TEACH and COURSE_TEXT tables are identical to those produced above by normalizing the SELECT_TEXT table.

INSTRUCTOR (InstSSN, InstName)
CAN_TEACH (Course#, InstSSN)
COURSE (Course#, Description, Hours)
COURSE_TEXT (Course#, ISBN)
TEXT (*ISBN,* AuthorName)

Fifth Normal Form (5NF)

Like 4NF, errors associated with 5NF deal with multivalued facts such as those that arise with ternary relationships or situations that look like ternary relationships. Kent notes that *5NF deals with cases where information can be reconstructed from smaller pieces of information which can be maintained with less redundancy.*

Consider the example depicted in Figure 5a and 5b. Instructors can teach for any department, departments offer courses, and instructors can teach many courses, though not necessarily every course. Figure 5a depicts the situation where we want to track which instructors are currently teaching courses for specific departments. The ternary relationship is only appropriate if instructors can be assigned to multiple departments simultaneously, can teach multiple courses, and some courses are offered by more than one department, e.g. the course IS 100 is offered by both the Department of Management Information Systems and Department of Computer Science. The binary relationships depicted are inadequate to capture this information. The relational table structure includes tables for the entities as well as the binary and ternary relationships.

INSTRUCTOR (InstSSN, InstName)
COURSE (Course#, Description, Hours)
DEPT (DeptName, DeptPhone)
TAUGHT (InstSSN, Course#, DeptName)
CAN_TEACH (InstSSN, Course#)
OFFERS_COURSE (DeptName, Course#)

Suppose, however, that the situation changes only slightly, and that courses belong to only a single department. Now the appropriate E-R diagram is that given in Figure 5b, and the relational table structure is:

INSTRUCTOR (InstSSN, InstName)
COURSE (Course#, Description, Hours, DeptName)
DEPT (DeptName, DeptPhone)
TAUGHT (InstSSN, Course#)
CAN_TEACH (InstSSN, Course#)

The simplified table structure enables the identification of which instructor taught which course for which department since courses only belong to a single department. This is accomplished through the TAUGHT table and the placement of *DeptName* as a foreign key in the COURSE table.

Keys to Avoiding E-R Modeling Errors

The first key to avoiding data modeling errors is to understand the type of errors that can occur. Experts in data modeling tend to assess the accuracy of E-R data models by comparing portions of an E-R model diagram to various data modeling patterns they have learned through experience. Recall that there are two general classes of errors: (1) incomplete data model errors, and (2) mis-modeled problem domain errors. Table 2 summarizes the various normalization errors, classifies the errors as resulting from incomplete data models or mis-modeled problem domains, and gives guidance for verifying the accuracy of an E-R model diagram. This table identifies the various data modeling patterns to be checked as you verify an E-R diagram.

Table 3 gives a listing of guidelines for avoiding E-R modeling problems. First and foremost, the database designer must develop a good understanding of the problem domain. A key aspect of the modeling effort is the early identification of functional dependencies among attributes of entities, i.e., which attributes are determinants of which other attributes. This includes identifying multivalued attributes, related groups of attributes, and multivalued dependencies.

Second, the database designer must, to the extent possible, develop a complete model of the problem domain. This is best accomplished by gathering additional views of the problem domain through interviews with potential system users. This effort must go beyond those end-users who use the current information system. In cases where the project deadline allows sufficient time, it is advisable to model a slightly larger problem domain than that for which one envisions developing application programs. This approach helps eliminate 2NF and 3NF modeling errors that can result from mis-modeled binary relationships.

Third, develop table definitions by first converting the

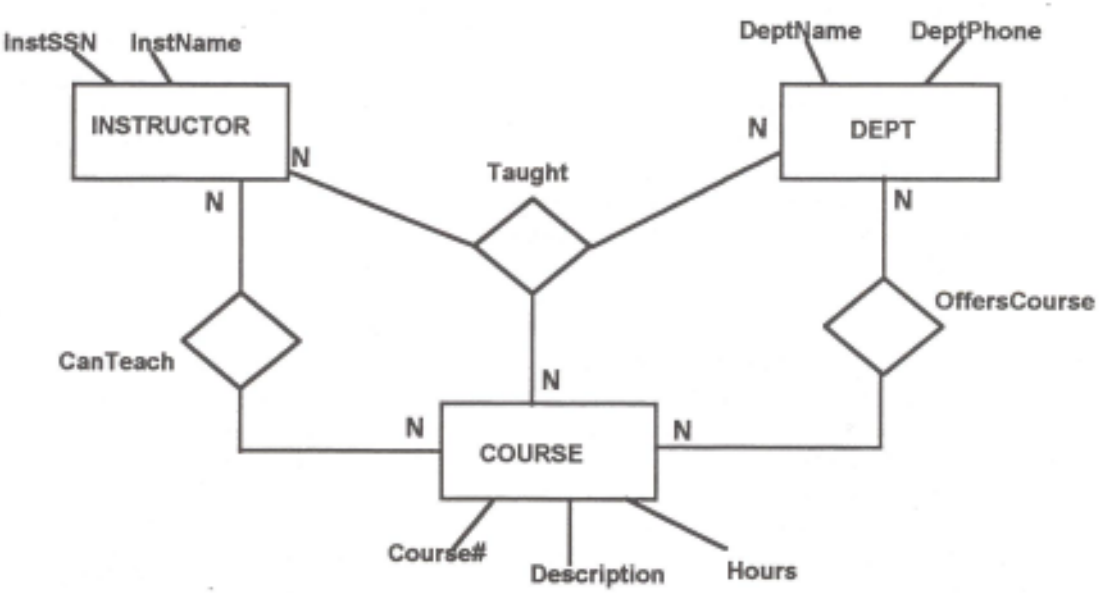

Figure 5a.

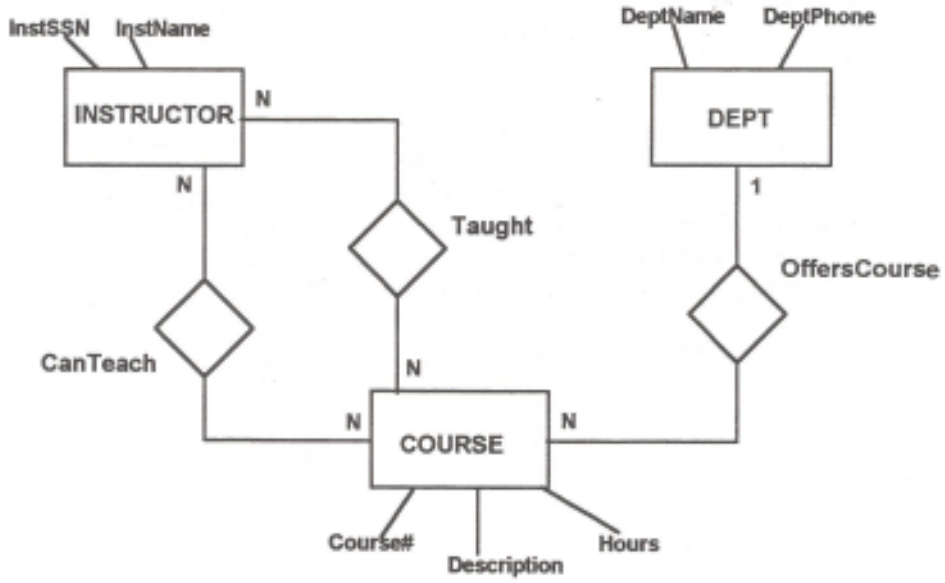

Figure 5b.

Figure 5: Example 5NF E-R Diagram

Normalization Error	Error Type	Verification Procedure
1NF - Attribute is multivalued.	Mis-modeled problem domain - attribute modeling error.	Check all attributes to ensure they are single-valued.
1NF - Set of attributes are multivalued and related to one-another.	Mis-modeled problem domain - multiple attribute modeling errors.	Check attributes identified as multivalued to determine if they are related to one-another.
2NF - Partial Key Dependency Errors.	Mis-modeled problem domain and/or incomplete data model - binary M:N modeling error.	Check entities with composite identifiers to ensure they are not, in fact, more than one entity - identify functional dependencies among the attributes.
3NF - Transitive Key Dependency Errors.	Mis-modeled problem domain and/or incomplete data model - binary 1:M modeling error.	Check entities with multiple non-identifier attributes for functional dependencies among the attributes.
BCNF - Determinants not serving as candidate keys.	Incomplete data model - binary M:N modeling error.	Check entities with composite identifiers, especially where identifier attributes appear to belong to multiple entities.
4NF - Entity has independent multivalued facts.	Mis-modeled problem domain - modeling a relationship as ternary that should be two or more binary relationships.	Check all ternary relationships for multivalued dependencies and an absence of determinants, and multivalued attributes that are independent of one-another.
5NF - Inability to join tables for binary relationships to form a necessary data view.	Mis-modeled problem domain - modeling relationships as two or more binary relationships where the data should be modeled as a ternary relationship.	Check original views of the database that were used in developing the E-R model, especially those involving three or more entities, for the ability to join entity attributes to form the original view.

Table 2: Summary of E-R Modeling Errors

Guideline	Critical Steps
1. Understand the problem domain.	1a. Identify functional dependencies among attributes (determinants) during logical data modeling.
	1b. Identify multivalued attributes, and related groups of attributes.
2. Develop a sufficiently comprehensive conceptual model.	2a. Gather additional end-user views of the data.
	2b. Model a slightly larger problem domain than that required for the project at hand.
3. Use a CASE tool to develop a relational schema (physical model) from the E-R diagram	3a. Determine if each view is supported by the relational schema—use a quick QBE query generator and report generator as a double-check of view support.
	3b. Use normalization as a double-check on view support and data maintainability.
	3c. Normalize through 5NF

Table 3: Guidelines for Avoiding E-R Modeling Errors

E-R diagram to a relational schema, then check to see if each view is supported. This step should be combined with normalization as a double check on view support and data maintainability, and on the enforceability of integrity constraints.

As a final word of caution, avoid the tendency in the information systems industry to be satisfied with normalizing only to 3NF. As was demonstrated above, 4NF and 5NF problems, although rare, can occur where the problem domain is modeled inadequately.

Acknowledgment: The author wishes to acknowledge the comments provided by two anonymous referees on an earlier version of the article. The author is also indebted to Dr. Jeffrey A. Hoffer, University of Dayton, for his comments on an early version of the article.

References

Chen, Peter P.S. (1976). The Entity-Relationship Model—Toward a Unified View of Data, *ACM Transactions on Database Systems*, 1(1), 9-36.

Codd, E.F. (1970). A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM*, 13(6), June 1970, 377-387.

Date, C.J. (1995). *An Introduction to Database Systems*, 6th Edition, Addison-Wesley.

Kent, William. (1983). A Simple Guide to Five Normal Forms in Relational Database Theory, *Communications of the ACM*, 26(2), 120-125.

Kroenke, David M. (1995). *Database Processing*, 5th Edition, Prentice-Hall.

Ling, Tok-Wang (1985). A Normal Form for Entity-Relationship Diagrams, *Proceedings of the 4th International Conference on Entity-Relationship Approach*, Chicago, Illinois, 24-35.

McFadden, Fred R. & Hoffer, Jeffrey A. (1994). *Modern Database Management*, 4th Edition, Benjamin/ Cummings.

Douglas B. Bock is Associate Professor and Chairman, Department of Management Information Systems, School of Business Administration, Southern Illinois University at Edwardsville. He received a Ph.D. in MIS from the Graduate School of Business, Indiana University under the tutelage of Dr. James H. Patterson. He has published articles in Decision Sciences, Communications of the ACM, Journal of Systems and Software, Journal of Database Management, and other learned journals. He serves on the editorial board of the Journal of Business and Economic Perspectives in the area of MIS. His research focuses on applying database management technology to common business problems, and on measuring systems development productivity.

0 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/article/entity-relationship-modeling-normalization-errors/51172

Related Content

Semantic Heterogeneity in Multidatabase Systems: A Review and a Proposed Meta-Data Structure
Te-Wei Wangand Kenneth E. Murphy (2004). *Journal of Database Management (pp. 71-87).*
www.irma-international.org/article/semantic-heterogeneity-multidatabase-systems/3321

A Novel Crash Recovery Scheme for Distributed Real-Time Databases
Yingyuan Xiao (2009). *Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends (pp. 769-787).*
www.irma-international.org/chapter/novel-crash-recovery-scheme-distributed/20763

Interesting Knowledge Patterns in Databases
Rajesh Natarajanand B. Shekar (2009). *Database Technologies: Concepts, Methodologies, Tools, and Applications (pp. 1652-1662).*
www.irma-international.org/chapter/interesting-knowledge-patterns-databases/7997

A Crash Course in Metaphysics for the Database Designer
John M. Artz (1997). *Journal of Database Management (pp. 25-30).*
www.irma-international.org/article/crash-course-metaphysics-database-designer/51186

Multi-Level Modeling of Web Service Compositions with Transactional Properties
K. Vidyasankarand Gottfried Vossen (2013). *Innovations in Database Design, Web Applications, and Information Systems Management (pp. 139-170).*
www.irma-international.org/chapter/multi-level-modeling-web-service/74392