01 Jan 2011

Issues and Guidelines in Modeling Decomposition of Minimum Participation in Entity-Relationship Diagrams

Cecil Eng Huang Chua
Missouri University of Science and Technology, cecq8z@mst.edu

Veda C. Storey

Recommended Citation

Chua, C. E., & Storey, V. C. (2011). Issues and Guidelines in Modeling Decomposition of Minimum Participation in Entity-Relationship Diagrams. *Communications of the Association for Information Systems, 29*(1), pp. 159-184. Association for Information Systems (AIS).

Communications of the Association for Information Systems

CAIS

Issues and Guidelines in Modeling Decomposition of Minimum Participation in Entity-Relationship Diagrams

Cecil Eng Huang Chua

Information Systems & Operations Management Department, University of Auckland

aeh.chua@auckland.ac.nz

Veda C. Storey

Computer Information Systems Department, Georgia State University

Abstract:

The entity-relationship model has long been employed for conceptual modeling of databases. Methodologies and heuristics have been developed, both for effective modeling and for translating entity-relationship models into relational models. One aspect of modeling that is often overlooked in design methodologies is the use of optional versus mandatory participation (i.e., minimum participation) on the development of relational databases. This tutorial complements existing instructional material on database design by analyzing the syntactic implications of minimum participation in binary, unary, and n-ary relationship sets and for the special case where the E-R diagram depicts a database where 3NF is not in BCNF. It then presents design modeling guidelines which demonstrate that (1) for binary 1:1 and 1:M relationship sets, the presence of optional participation sometimes means that the relationship set should be represented in the relational model by a separate relation, (2) unary relationship sets cannot have a (1,1) participation, (3) n-ary relationship sets that have a (1,1) participation can be simplified to be of lower connectivity, and (4) decomposition is not a substitute for normalization. Illustrative examples and modeling guidelines are provided.

Keywords: software design, data base systems, conceptual modeling, computer science theory

Volume 29, Article 9, pp. 159-184, September 2011

The manuscript was received 3/5/2010 and was with the authors 4 months for 3 revisions.

I. INTRODUCTION

Conceptual modeling remains of fundamental importance in efforts to build effective databases. Many database methodologies that employ entity-relationship (E-R) modeling highlight that entity sets have at least two kinds of minimum participation in relationship sets—optional and mandatory participation [Atzeni et al., 1999; Carter, 2003; Connolly and Begg, 2010; Coronel et al., 2011; Dennis et al., 2009; Gillenson, 2005; Hoffer et al., 2011; Hoffer et al., 2010; Kroenke and Auer, 2010; Mannino, 2011; Ricardo, 2011; Silberschatz et al., 2011; Ullman and Widom, 2002; Watson, 2006; Whitten and Bentley, 2007]. It is well-accepted that relationship sets in E-R diagrams have not only a maximum 1:1, 1:M, or M:N participation, but also a minimum participation of 0 or 1. Students and practitioners can quickly appreciate why knowing whether relationship sets have a maximum 1:1, 1:M, or M:N participation is important when they study decomposition [Teorey et al., 1986]. Each maximum participation results in a distinct set of relations and unique placements of primary/foreign key pairs. However, the implications of minimum participation on the final database implementation is often not considered fully in database methodologies (see Tables 2 to 4). Minimum participation certainly facilitates the understanding of E-R models [Bodart et al., 2001], but more importantly, minimum participation impacts the structure of the logical design in numerous ways.

The objective of this tutorial is to complement existing instructional material by explaining how modeling minimum participation at the conceptual level impacts logical database design. The specific focus of this tutorial is on the syntax of E-R diagrams, and how that syntax impacts the development of a relational database. This tutorial shows that (1) for binary 1:1 and 1:M relationship sets, the presence of optional participation sometimes means the relationship set is translated as a separate relation during logical design, (2) unary relationship sets having a (1,1) participation cannot exist, (3) n-ary relationship sets that have (1,1) participation constraints can be simplified to be of lower connectivity (e.g., 3-ary relationship sets can be expressed as binary), and (4) decomposition is not a substitute for normalization. Table 1 summarizes the issues covered and their impact on database design through explicitly stated guidelines that assist in the modeling process. The intended contribution is to augment existing database design methodologies so that better designs can be created by both students and practitioners.

Table 1: Issues Covered

Guideline	Description	Issue Addressed
1	*A (0,1):(0,1) binary relationship set can be decomposed into a relation separate from the relations representing its participating entity sets.*	Binary Relationship Sets
2	*For a (0,1):(1,1) binary relationship set, the entity set that has the (1,1) participation can take the primary key of the other entity set as both a candidate key and a foreign key.*	Binary Relationship Sets
3	*Attributes placed in a (0,1):(0,1) binary relationship set can not be rewritten by placing them in a participating entity set.*	Binary Relationship Sets
4	*In a 1:M binary relationship set, where one entity set's participation is (0,1), the relationship set can be decomposed into an independent relation.*	Binary Relationship Sets
5	*Attributes placed in a 1:M relationship set where one entity set has a (0,1) participation cannot be rewritten by placing them in a participating entity set.*	Binary Relationship Sets
6	*It is reasonable to use associative entity sets to represent binary relationship sets having any other participation than (1,1).*	Binary Relationship Sets
7	*No entity set can have a (1,1) participation with a unary relationship set.*	Unary Relationship Sets
8	*It is reasonable to have attributes of unary relationship sets. Similarly, it is reasonable to use associative entity sets to represent unary relationship sets.*	Unary Relationship Sets
9	*(1,1) participations are not information efficient for n-ary relationship sets, where n > 2.*	N-ary Relationship Sets
10	*There are cases where after decomposing an ER diagram, the relations are not yet in DKNF.*	Decomposition

II. CONCEPTUAL MODELING WITH ENTITY-RELATIONSHIP DIAGRAMS

This section reviews basic terms and foundational "rules" needed to discuss issues related to minimum participation. These foundational rules are critical to a full understanding of how E-R diagrams work. Such an understanding requires an appreciation of Chen's [1976] original formulation of E-R diagrams. Hence, Chen's original terms and meanings are employed. Terms and concepts used in practice are often "simplified" versions of terms used by Chen [1976]. For example, it is commonplace to refer to "entity sets" as "entities" [Bennett et al., 2010; Carter, 2003; Connolly and Begg, 2010; Gillenson, 2005; Hoffer et al., 2011; Hoffer et al., 2010; Kroenke and Auer, 2010; Ricardo, 2011; Valacich et al., 2009]. Similarly, participation is often considered to be between entity sets [Bennett et al., 2010; Carter, 2003; Connolly and Begg, 2010; Gillenson, 2005; Hoffer et al., 2011; Hoffer et al., 2010; Kroenke and Auer, 2010; Ricardo, 2011; Valacich et al., 2009]. In the original work [Chen, 1976], entity sets participate in relationship sets, and the participation between entity sets is inferred from the entity set participation in the relationship set. The distinctions between entity sets and entities, relationship sets and relationships, attributes and values, and entity keys and primary keys are important to fully appreciate the guidelines proposed in this article. This section also reviews how inferences are obtained in E-R diagrams, which is important for effective modeling.

Scope

Entity-relationship diagrams may be interpreted in different ways. First, the way in which E-R diagram semantics should best be represented has been analyzed and debated [Burton-Jones and Meso, 2002; Shanks et al., 2008; Shanks et al., 2004; Siau et al., 2004; Wand et al., 1999; Weber, 1997]. This article does not address that issue, nor does it assume anything about the cognitive processes of systems analysts. The examples presented in this article may be represented in other ways. However, the focus of our arguments rest wholly on the syntax of E-R diagrams.

Second, the primary purpose of an E-R diagram is not to form a foundation for logical database design, but to model a real-world domain at an appropriate level of richness. An E-R diagram is a tool intended to capture the rich information semantics of a problem [Navathe, 1992], and as a "unifying framework" [Chen, 1976] for different kinds of database implementations, including relational, hierarchical (e.g., XML), and object-oriented. E-R diagramming, or indeed any conceptual data model, should not be constrained by the needs of logical design. There is, therefore, a broad array of ways in which E-R diagrams can be applied to database design problems. Considerable work has focused on finding effective ways to translate E-R models into relational models [Fahrner and Vossen, June 1995; Markowitz and Makowsky, 1990; Markowitz and Shoshani, 1992; Storey, 1991; Teorey et al., 1986], so much so that the translation of E-R models into relational models is material commonly found in most textbooks [Bennett et al., 2010; Carter, 2003; Connolly and Begg, 2010; Gillenson, 2005; Hoffer et al., 2011; Hoffer et al., 2010; Kroenke and Auer, 2010; Ricardo, 2011; Valacich et al., 2009]. This tutorial focuses purely on this narrow use of E-R diagrams.

Third, there are multiple processes of database design, and separate instructors employ their own preferred methodologies. It is assumed that a database course follows Navathe's [1992] database design levels, where the entity-relationship diagram is created first (the conceptual level). A set of rules, called *decomposition* rules [Teorey et al., 1986], are then applied to convert the entity-relationship diagram into a set of relational database relations (logical level) [Navathe, 1992]. These relations are checked to eliminate insertion, update and deletion anomalies using 1-5NF normalization, which results in the final set of relations that will be implemented in a relational database. This tutorial focuses on decomposition, and thus the material covered assumes that all requirements have been determined, but the logical database design has not yet been developed.

Fourth, there are known flaws in the expressiveness of E-R diagrams [Date, 2004; Dey et al., 1999; Wintraecken, 1989]. As one simple example, E-R diagrams provide a facility to represent information about primary keys, but not entity keys. Thus, in some cases, the diagrams will not fully reflect the mathematics of our presentation. Material that is not presented in the diagrams will be described in the text.

Finally, there are many variations in E-R diagramming conventions, including the original Chen notation [Chen, 1976], Crow's foot, and IDEF1X (http://www.idef.com/IDEF1x.htm). The merits and drawbacks of various E-R diagram notations have been analyzed elsewhere [Song et al., 1995]. This tutorial adopts the Chen notation, also known as the "box and diamond" notation, and includes weak and associative entity sets. This notation is adopted, because it:

- Emphasizes that entity sets have a participation in a relationship set. In contrast, many other notations encourage thinking that entity sets have participation with other entity sets.
- Equally emphasizes the idea of attributes of relationship sets as well as attributes of entity sets.
- Cleanly allows the presentation of n-ary relationship sets.
- Employs a symbol for associative entity sets that is distinct from the symbol for weak entity sets.

All of the above characteristics of the Chen notation are employed below. The use of the Chen notation should not be construed as a preference for the Chen over other notations, which might be more suitable in other domains [Song et al., 1995].

The supertype/subtype construct is commonly employed in modern entity-relationship diagram practice. This tutorial does not discuss the decomposition of supertypes/subtypes, because they incorporate complex semantics that are not easy to address structurally in the relational model (i.e., solely using relations and attributes of relations). A discussion of subtypes would divert attention away from the core issue this article discusses, i.e., decomposition of minimum participation. For example, the subtype concept of disjointness is represented in relational databases as a condition where the intersection of any two relations representing subtypes is always empty. Similarly, completeness is represented as a condition where the number of tuples in the union of all foreign keys in the subtypes linking to the supertype is the same as the number of tuples in the supertype. An adequate logical representation of both require that relational algebra be layered on top of the relations and attributes representing the database. The complexity inherent in subtyping suggests that a proper discussion of subtyping is best left for another tutorial. The semantics of subtypes that can be expressed structurally can be represented at the conceptual level as a relationship set having a (0,1):(1,1) participation. In some cases, examples below are better represented as subtypes. The (0,1):(1,1) participation is used to keep the example simple.

Throughout this article, we will employ a running example of a university student tracking system to illustrate various issues concerning minimum participation. This example is selected mainly because it is a typical example in many database textbooks (e.g., Coronel et al., 2011; Hoffer et al., 2010; Mannino, 2011). Different aspects of this example will be described in natural language. E-R diagram versions of the example, and (where appropriate) the corresponding decomposed relations will be presented to illustrate points.

Definitions

In Chen's [1976] notation, entity sets are presented as rectangles, relationship sets as diamonds, and attributes as bubbles. Entities in entity sets may be "linked" to relationships in relationship sets. The range of times an entity may be linked to relationships is denoted by minimum and maximum participation. Every relationship in the relationship set must have exactly one and only one "link" to the connected entity sets. The participation of the entity set to other entity sets is then inferred from the participation of the entity set to the relationship set. Because relationship sets are sets, the combined entities that the relationship in the relationship set links to is unique within the relationship set.

Entity sets (the rectangles) contain entities that are individual "things." These entity "things" are unnamed, and described by attributes. For simplicity, one can think of entities as equivalent to the "record" of an implemented database. For example, the entity set *Student* would have the entities student1, student2, student3, etc. These students would have attributes like name, matriculation number, and gender. The entities identify the values of the attributes. Thus, the entity student1 might have the name "Joe," matriculation number "10000," and gender "Male." There also exist particular sets of attributes that identify entities, which are called entity keys. One entity key is chosen by the developer as the entity set's primary key. In the case of the entity set *Student*, the matriculation number identifies each entity, and hence is the primary key.

Figure 1 provides an example of an entity-relationship diagram in which there exists an entity set called *Student*. Every individual student entity has a matriculation number (*Matric*) and *Name*. If one can identify the student entity, one will be able to determine the value of the matriculation number and name. Thus, knowing student1 allows us to determine the name is *Joe* and the matriculation number is *10000*. If one knows the matriculation number value of a student, one will be able to identify that particular student entity, and by inference, the student's name. Thus, knowing the matriculation number *10000*, one can identify student1, and by inference, the name *Joe*. There is also an entity set called *Club*. If one can identify the specific club entity, one can determine the club registration number (*Crno*), and *Advisor*.

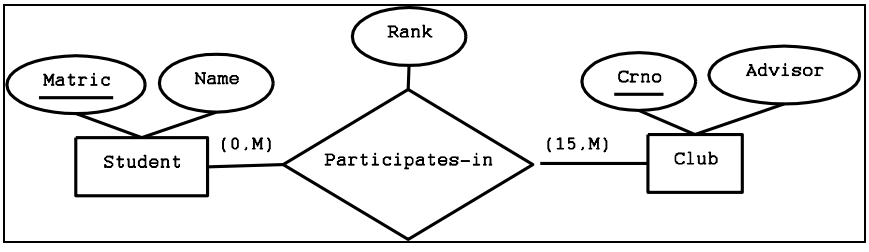


Figure 1. Basic E-R Diagram

Every student entity in the *Student* entity set is linked to at least zero, and possibly more, relationships in the *Participates-in* relationship set. Every club is linked to at least fifteen relationships, and possibly more, in the *Participates-in* relationship set. Every relationship in the *Participates-in* relationship set is linked to exactly one student and one club. By *inference*, a student in the *Student* entity set is linked to at least zero, and possibly more club entities. This is because the student is linked to zero or more participates-in relationships, which in turn are linked to exactly one club. Similarly, every club is linked to at least fifteen students. Given any relationship in *Participates-in*, one can determine the value of *Rank*. Therefore, once one is given a matriculation number in *Student*, and a club registration number in *Club*, one can determine the relationship and, therefore, the rank.

Note that, compared to the Crow's foot, or similar notations, the participation appears reversed. In the Crow's foot, the participation would appear on the opposite side of the relationship set. In the original Chen notation, participation is from the entity set to the relationship set; the link to other entity sets is inferred. Thus, a student has zero or more participates-in, and, therefore, zero or more clubs. Similarly, a club has fifteen or more participates-in, and hence fifteen or more students. In the Crow's foot notation, the link to the relationship set is inferred, whereas the link to other entity sets is made explicit. Given that this tutorial emphasizes decomposition of relationship sets, the Chen notation better highlights the issues involved.

Basic Inferences

There are two basic inferences which form the foundation for the modeling heuristics detailed in the remainder of this article. They are (1) inferences about the characteristics of relationship and entity sets; and (2) inferences about the participation (1,1):(1,1).

Relationship and Entity Sets

There are two main structural distinctions between entity sets and relationship sets. First, entity sets comprise sets of entities, whereas relationship sets link entity sets. In the original formal definition [Chen, 1976], relationship sets are sets of entity tuples (i.e., ordered sets where each element of the set comprises two or more entities). Second, entity sets have explicit entity and primary keys; relationship sets do not.

Beyond these structural conventions is also a set of "practices" associated with entity sets and relationship sets. Entity sets are usually employed to model "nouns," whereas relationship sets model "verbs" [Chen, 1997]. In natural language, sometimes one wants to treat a verb as a noun. For example, one might use the verb *run* in "Jill runs up the slope." The noun *running* is also used in "Running is good for you." Similarly, in conceptual modeling, there is sometimes a desire to "objectify" a relationship set [Nijssen and Halpin, 1989].

A relationship set can be transformed to an entity set by performing the following [Atzeni et al., 1999; Ullman, 1983]:

1. Insert new binary relationship sets between the original relationship set and its participating original entity sets.
2. Replace the original relationship set with a new entity set.
3. Define the participation of the entity sets to the new relationship sets. Write the participation of the original entity set to the new relationship set as the original participation of the original entity set to the original relationship set. Write the participation of the new entity set to each new relationship set as (1,1).
4. Define the primary key of the new entity set. The primary key will be a subset of the primary keys of the original entity sets.

The fact that the primary key of the relationship-set-converted-to-entity-set can be a subset is critical to understanding many of the further issues. One rule for all entity and primary keys is that they must be minimal. This means if an attribute can be removed from the entity/primary key and the entity/primary key can still uniquely identify all entities in the entity set, then it is better to remove the attribute. Figure 2 illustrates the transformation of the *Participates-in* relationship set of Figure 1.

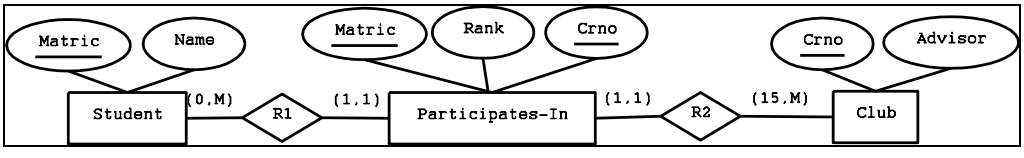


Figure 2. Transformation of Participates-in from Relationship Set to Entity Set

The construct for identifying a relationship set transformed into an entity set is called various names, including associative entity [Hoffer et al., 2011; Hoffer et al., 2010; Satzinger et al., 2009; Whitten and Bentley, 2007], intersection entity [Dennis et al., 2009], bridge entity [Coronel et al., 2011], or aggregation [Elmasri and Navathe, 2011; Silberschatz et al., 2011; Ullman and Widom, 2002]. This article employs the term *associative entity set.* The associative entity set is primarily used within a context in which the developer wishes to link a relationship set to another relationship set. For example, consider a situation where a *Student* who *Participates-in* a *Club* must pay a *Due*. Ideally, one would like to represent this as shown in Figure 3. However, the representation is syntactically incorrect.

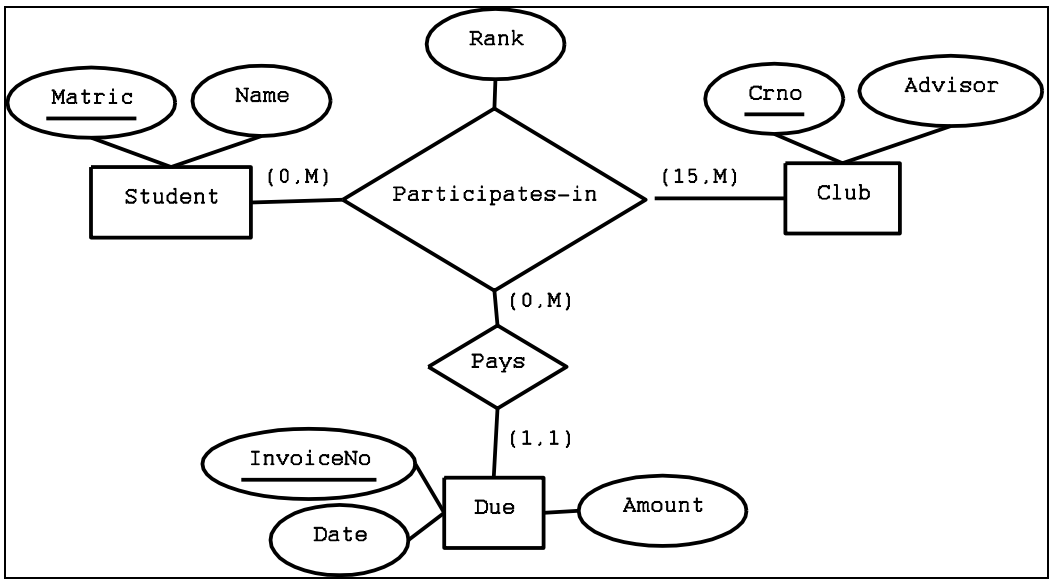


Figure 3. Syntactically Incorrect Way of Linking Due to Participates-in

As noted in Figure 2, *Participates-in* can be transformed into an entity set, which can then be linked to *Due*. Figure 4 presents this (syntactically legal) equivalent.

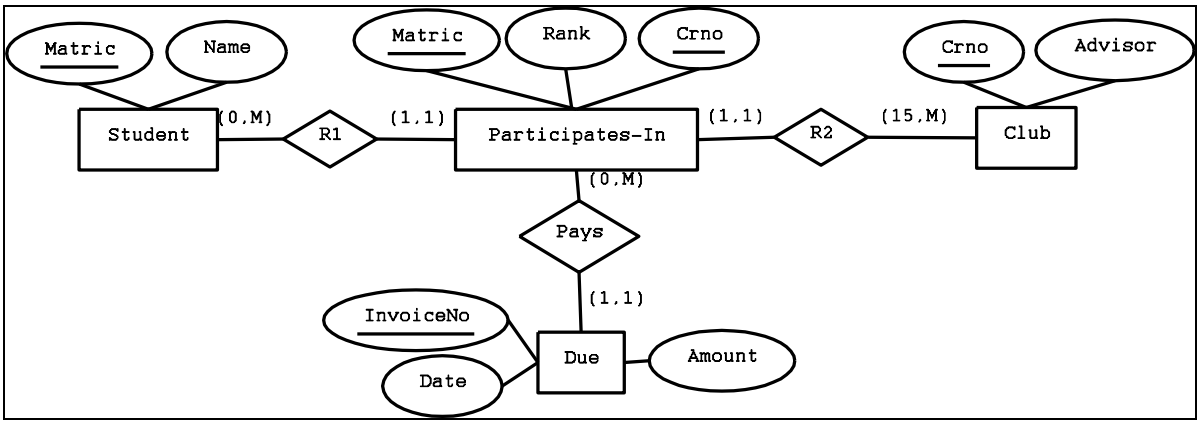


Figure 4. Syntactically Correct Way of Linking Due to Participates-in

The associative entity set is shorthand for conveying this idea. Figure 5 is equivalent to Figure 4.

Participation (1,1):(1,1)

Within E-R Diagrams, every "thing" has a direct link to other things, which may be expressed in terms of participation. Thus, within an entity set, each entity is linked to attribute values, where every entity has a minimum of 1 and a maximum of 1 (1,1) link to each value. Each value has a link to entities. For example, the value "Buckaroo Banzai" of the attribute *Advisor* would have a link to one or more (1,M) entities in *Club.* The convention (A,B):(X,Y) denotes the participation of the relationship between two things. Thus, the participation of the relationship between *Club* and *Advisor* is (1,1):(1,M).

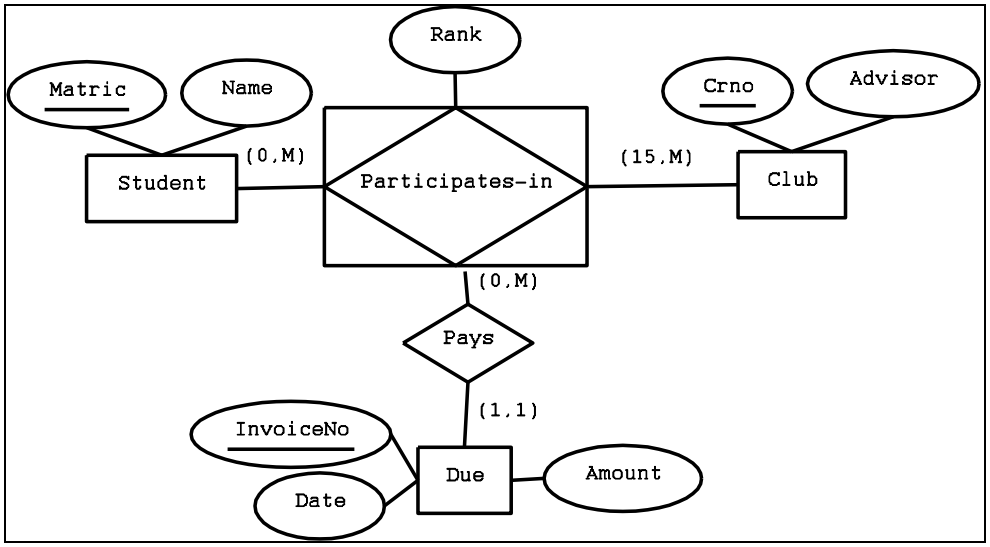


Figure 5. Figure 4 as an Associative Entity Set

Many of the important pieces of information from an E-R diagram are actually established through inferences. For example, the link between a primary key and a nonprimary key attribute in an entity set is established through the entity—each value of the primary key has one and only one entity, which in turn has one and only one of that attribute's value.[1] Similarly, the link between entity sets is inferred from the link between the entity sets and the relationship sets those entity sets are linked to. Because every relationship in *Participates-in* links to a minimum of 1 and maximum of 1 entity in *Club*, once it is established which student is of interest, the particular clubs that student participates in can also be determined. Such inferences can be further extended by identifying the advisors in the clubs that each particular value of matriculation number participates in through the chain.

A particularly important special case of this (1,1) link occurs when two things have a (1,1) link to each other (i.e., a (1,1):(1,1) participation). In this case, anything one might infer using one of those things can be inferred using the other of those things. For example, the same information is derived regardless of whether the primary key of an entity set, or the entity of the entity set is known.

This special case is especially important when evaluating an E-R diagram in terms of information efficiency and information equivalence. Two E-R diagrams that convey the same idea are information equivalent. From a purely syntactic perspective, given two information equivalent E-R diagrams, the one that has the fewest constructs is more information efficient. Consider the example in Figure 6, where all staff are taxpayers and vice-versa (student taxpayers of interest must be staff). Note that in Figure 6(b), *Taxpayerno* is an entity key. To establish all the participations among the "things" of Figure 6(a) and (b), one would need the same participations among the entities and attributes. Thus, Figure 6(a) and 6(b) are information equivalent, and Figure 6(b) is more information efficient.

Although information efficiency is good in principle, it may not always be desirable to maximize information efficiency. As an analogy, whereas shorter, crisper English sentences are generally better, there are many instances where more elaborate sentence structures are preferred. However, given that this tutorial focuses only on E-R modeling syntax, it assumes that given two information equivalent E-R diagrams, the more information efficient one is better.

[1] Null is considered a value.

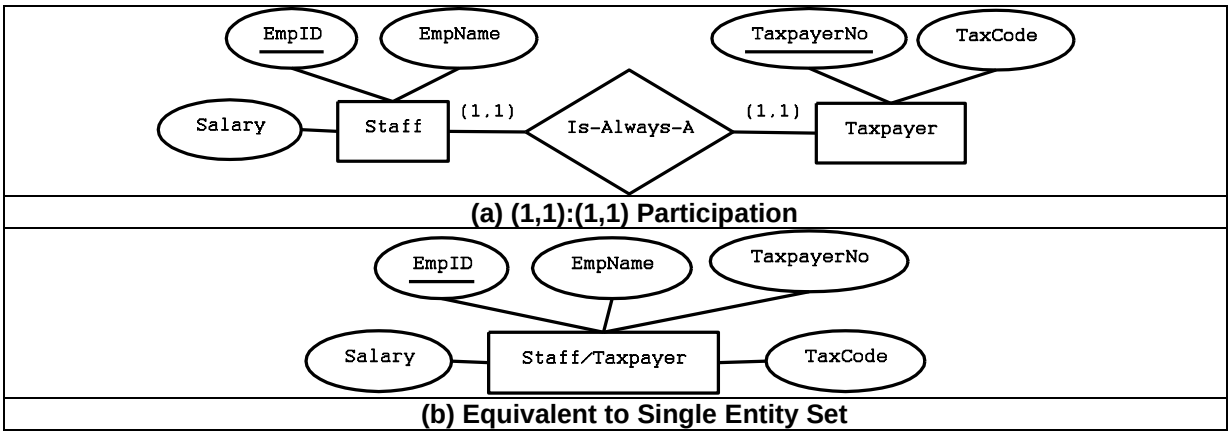


Figure 6. Equivalencies Between (1,1) Participation Relationship Sets

III. BINARY RELATIONSHIP SETS

This section re-examines beliefs about binary relationship sets, focusing principally on 1:1 and 1:M relationship sets. The analysis is then extended to focus on what it means to have associative entity sets on binary relationship sets.

1:1 Relationship Sets

A common belief among E-R diagram practitioners is that all the various forms of 1:1 relationship sets should decompose into, at most, two relations, where a foreign key is placed in one relation to link the two. Table 2 presents some examples.

This practice may stem in part from Teorey et al.'s [1986] classic work on decomposition, which explicitly employed foreign keys as a way to represent binary 1:1 relationship sets of all forms, i.e., (1,1):(1,1), (0,1):(1,1), and (0,1):(0,1). However, Teorey et al.'s [1986] objective was to demonstrate that E-R diagrams could map to relational models, not to demonstrate the most appropriate mapping of one to the other.

In the particular case of (0,1):(0,1) relationship sets, it is often better if the relationship set is represented as a separate relation/table. Consider the following example: There are students (identified by a matriculation number), and athletes (identified by a sports council ID number). A student may or may not be (0,1) an athlete, and an athlete may or may not be (0,1) a student. Student-athletes receive a stipend to compensate them for their reduced ability to get a student job, where the stipend value varies between student-athletes. Figure 7 presents the E-R diagram illustrating this example.

The attribute *Stipend* is not appropriate as an attribute of *Student*, because not all students receive stipends, only student-athletes. Thus, the participation from a student entity to any value in stipend is (0,1). Similarly, *Stipend* is not an attribute of *Athlete*, because only student-athletes receive them. *Stipend* is thus an appropriate attribute of the relationship set *CouldBe*. However, if the E-R diagram is decomposed following the rule where foreign keys are placed in either entity set, then the stipend attribute is placed in an inappropriate relation. If the *Stipend* attribute is placed in *Athlete*, its value will be null for all non-student athletes. Similarly, when placed in *Student*, it will be null for all non-athlete students. This design choice is problematic for two reasons: (1) A new value (null) is added to the domain of the attribute, and (2) earlier research has demonstrated serious problems with including nulls in relational databases, and it is generally agreed that if one can design nulls away, one should [Biskup, 1979; Goldstein, 1981; Grant, 1977].

A relational database design that addresses the problem of nulls would comprise [Bernick, 2003]:

- Student [MatricNo]
- Athlete [SCID]
- Student-Athlete [MatricNo, SCID, stipend], where *SCID* is a candidate key of *Student-Athlete*.

Table 2: Textbook Decomposition Rules for 1:1 Relationship Sets	
Textbook	Quote
Bennett et al., 2010, p. 522	One-to-one associations are implemented as foreign key attributes.
Carter, 2003, p. 26	In a one-many relationship, the primary key of the entity type at the "1" side of the relationship also appears in the entity type at the "many" side of the relationship. A similar method is used in representing one-one relationships.
Connolly and Begg, 2010, p. 493	Instead, the participation constraints … are used to help decide whether it is best to represent the relationship by combining the entities involved into one relation or by creating two relations and posting a copy of the primary key from one relation to the other.
Coronel et al., 2011, p. 160	… there are two options for selecting and placing the foreign key: 1. Place a foreign key in both entities … place a foreign key in one of the entities.
Gillenson, 2005, p. 135	… what if the modality was zero on both sides? Then we would have to make a judgment call … to decide whether it is more likely that a salesperson is not assigned to an office … or that an office is empty.
Hoffer et al., 2011, p. 366	For a binary or unary one-to-one (1:1) relationship between two entities A and B … the relationship can be represented by any of the following choices: 1. Adding the primary key of A as a foreign key of B 2. Adding the primary key of B as a foreign key of A Both of the above
Hoffer et al., 2010, p. 208	Binary one-to-one relationships can be viewed as a special case of one-to-many relationships…. The process of mapping such relationships to relations requires two steps. First, two relations are created, one for each of the participating entity types. Second, the primary key of one of the relations is included as a foreign key in the other relation.
Kroenke and Auer, 2010, p. 198	…. a 1:1 relationship between these entities can be represented in one of two ways. You can place the primary key of the first table in the second as a foreign key, or you can place the primary key of the second table in the first as a foreign key.
Kroenke, 2011, p. 156	Represent relationships — Use foreign keys — Add additional tables for N:M relationships
Ricardo, 2011, p. 120	If A and B are truly separate entities having a one-to-one relationship, then we can put the key of either relation in the other table to show the connection….
Silberschatz et al., 2011, p. 289	In the case of one-to-one relationships, the relation schema for the relationship set can be combined with the schemas for either of the entity sets.
Valacich et al., 2009, p. 304	For a binary or unary one-to-one (1:1) relationship between the two entites A and B … the relationship can be represented by any of the following choices: 1. Adding the primary key of A as a foreign key of B 2. Adding the primary key of B as a foreign key of A 3. Both of the above.
Watson, 2006, p. 137	Since mapping a 1:1 relationship follows the same rules as for any other data model, the major consideration is where to place the foreign key(s). There are three alternatives: 1. Put the foreign key in dept … Put the foreign key in emp … Put a foreign key in both dept and emp.

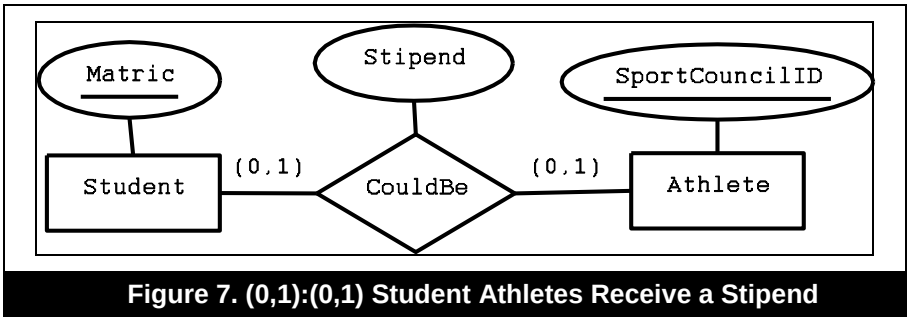


Figure 7. (0,1):(0,1) Student Athletes Receive a Stipend

This would eventually be implemented as the three tables below. Observe that *Student* and *Athlete* have more records than *Student-Athlete*.

Student
MatricNo
1000
1001
1003
1004

Athlete
SCID
A101
A102
A103
A104

Student-Athlete		
MatricNo	SCID	Stipend
1001	A103	3000
1004	A104	5000

The resultant relations demonstrate that:

> Guideline #1: *A (0,1):(0,1) binary relationship set can be decomposed into a relation separate from the relations representing its participating entity sets. The keys of the entity sets are, separately, candidate and foreign keys of the relation representing the relationship set.*

A further demonstration of Guideline #1 can be obtained by performing a transformation of the relationship set of a (0,1):(0,1). The transformation of Figure 7 would result in Figure 8, where *SportCouncilID* is an entity key of *CouldBe*. In this transformation, the subset of primary key rule is followed. *MatricNo*, *SportCouncilID* combined is not an entity key. Each attribute individually is an entity key.

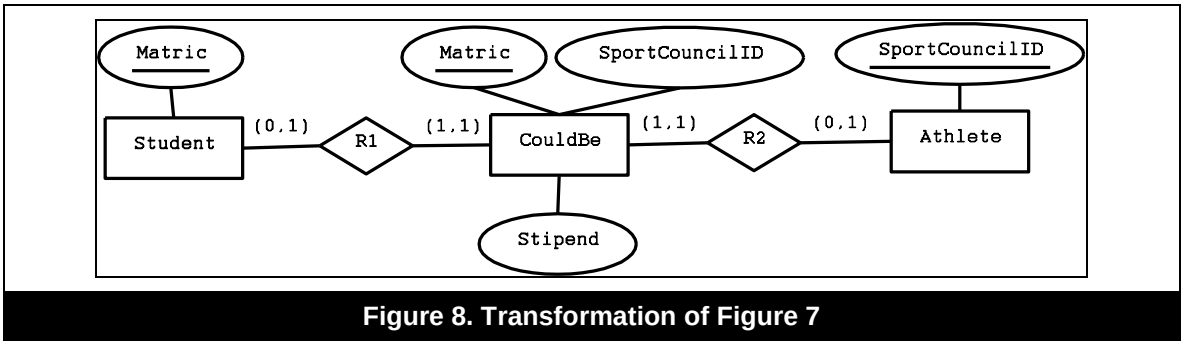


Figure 8. Transformation of Figure 7

A decomposition of Figure 8 would result in the three relations *Student*, *Athlete*, and *Student-Athlete* (renamed from *CouldBe*).

Given that (0,1):(0,1) binary relationship sets can be decomposed in a manner different from that espoused by many database textbooks, the question arises as to whether other forms of binary 1:1 E-R diagrams can be decomposed differently. As highlighted in Section 2, the binary (1,1):(1,1) E-R diagram is information equivalent to a single entity set. What about the binary (0,1):(1,1) E-R diagram?

Consider the following example: A student may or may not be (0,1) a Ph.D. student. However, a Ph.D. student is always a student (1,1). Students are identified by matric numbers. A Ph.D. student has a special, uniquely identifying Ph.D. student number. The thing that makes a Ph.D. student a special kind of student is that the Ph.D. student has a dissertation. This results in the E-R diagram presented in Figure 9.

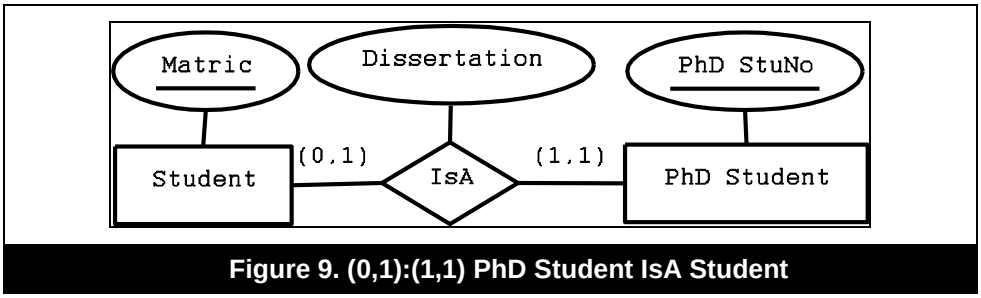
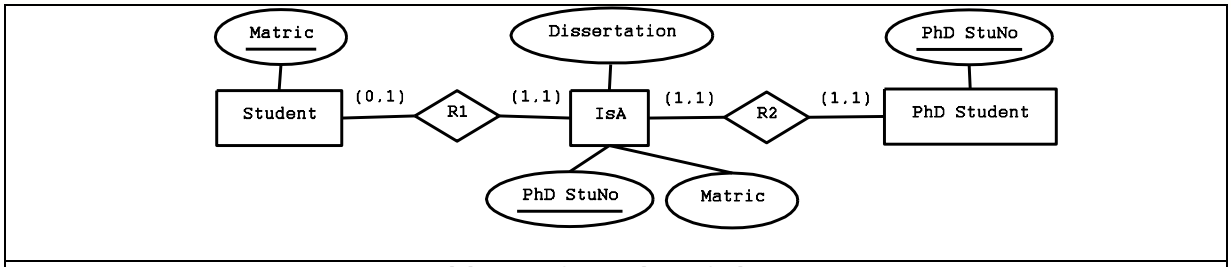
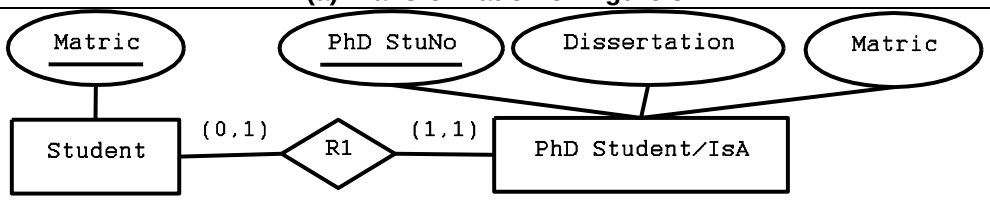


Figure 9. (0,1):(1,1) PhD Student IsA Student

Transforming Figure 9 results in Figure 10(a), where both *PhD StuNo*, and *Matric* are entity keys of *IsA*. However, *IsA* and *PhDStudent* share a (1,1):(1,1) participation. Thus, Figure 10(a) is informationally equivalent to Figure 10(b), where *PhD StuNo* and *Matric* are entity keys of *PhD Student/IsA*.



(a) Transformation of Figure 8



(b) Information Equivalent Version of (a)

Figure 10. Alternate Versions of PhD Student IsA Student

Notice that the principal difference of Figure 10(b) from Figure 9 is that the attribute *Dissertation* is now wholly in the entity set *PhD Student*/*IsA*. Hence, when decomposing Figure 9, there is no need to create a separate relation for *IsA*. The decomposition of Figure 9 would thus be:

- Student [Matric]
- PhD Student [PhDStuNo, Matric, Dissertation], where *Matric* is a candidate and foreign key.

This decomposition would result in the below tables. Observe that there are always fewer *PhD Students* than *Students*.

Student
Matric
1000
1001
1002
1003

PhD Student		
PhDStuNo	Matric	Dissertation
A100	1000	Why chickens cross roads
A101	1001	Rabbits' uses of IT

This decomposition is a refinement of the rule stated in the textbooks [Teorey et al., 1986]:

> Guideline #2*: For a (0,1):(1,1) binary relationship set, the entity set that has the (1,1) participation can take the primary key of the other entity set as both a candidate key and a foreign key.*

There is a third insight regarding attributes that can be drawn from these examples:

> Guideline #3*: Attributes placed in a (0,1):(0,1) binary relationship set can not be rewritten by placing them in a participating entity set.*

1:M Relationship Sets

Substantial work on database design similarly suggests that a 1:M relationship set should decompose into two relations, with the primary key of the entity set that participates many times in the relationship set becoming the foreign key of the side that participates once. Table 3 presents some examples.

However, there are occasions when it is desirable that the 1:M binary relationship set be preserved as a separate relation. Consider the following example: Students at a university may be affiliated or unaffiliated with an academic department. A student affiliated with the IT department is attempting to obtain an IT specialization. An unaffiliated student is taking a general degree. When students affiliate, they may only do so with one department. A student who affiliates must choose a specialization. For example, a student in IT might specialize in the technical track or the management track. Departments must have students (otherwise they do not exist). The E-R Diagram in Figure 11 depicts this situation.

Table 3: Textbook Decomposition Rules for 1:M Binary Relationship Sets	
Textbook	Quote
Bennett et al., 2010, p. 522	One-to-many associations can be treated like collections. Many-to-many associations become separate tables.
Carter, 2003, p. 26	In a 1:N relationship between two entity types A and B, the primary key in A will also appear as the foreign key in B.
Connolly and Begg, 2010, p. 493	For each 1:* binary relationship, the entity on the "one side" of the relationship is designated as the parent entity and the entity on the "many side" is designated as the child entity. To represent the relationship, we post a copy of the primary key attribute(s) of the parent entity into the relation representing the child entity, to act as a foreign key.
Coronel et al., 2011, p. 77	The one-to-many (1:M) relationship is easily implemented in the relational model by putting the primary key of the 1 side in the table of the many side as a foreign key.
Gillenson, 2005, pp. 135–136	The rule is that the unique identifier on the "one side" of the one-to-many relationship is placed as a foreign key in the table representing the entity on the "many side."
Hoffer et al., 2011, p. 336	A binary one-to-many (1:N) relationship in an E-R diagram is represented by adding the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation that is on the many side of the relationship.
Hoffer et al., 2010, p. 207	For each binary 1:M relationship …. Next, include the primary key attribute (or attributes) of the entity on the one-side of the relationship as a foreign key in the relation that is on the many-side.
Kroenke and Auer, 2010, p. 199	For 1:N relationships between strong entities, that's all there is to it. Just remember: "Place the primary key of the parent in the child as a foreign key."
Kroenke, 2011, p. 156	The Adviser entity has a 1:N relationship to the Student entity …. The correct choice is to place AdviserName in the Student table.
Ricardo, 2011, p. 119	If A and B are strong entity sets and binary relationship A:B is one-to-many, we place the key of A (the one side) in the table for B (the many side), where it becomes a foreign key.
Valacich et al., 2009, p. 304	A binary one-to-many (1:N) relationship in an E-R diagram is represented by adding the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation that is on the many side of the relationship.
Watson, 2006, p. 92	The 1:m relationship is mapped by adding a column to the entity at the many end of the relationship.

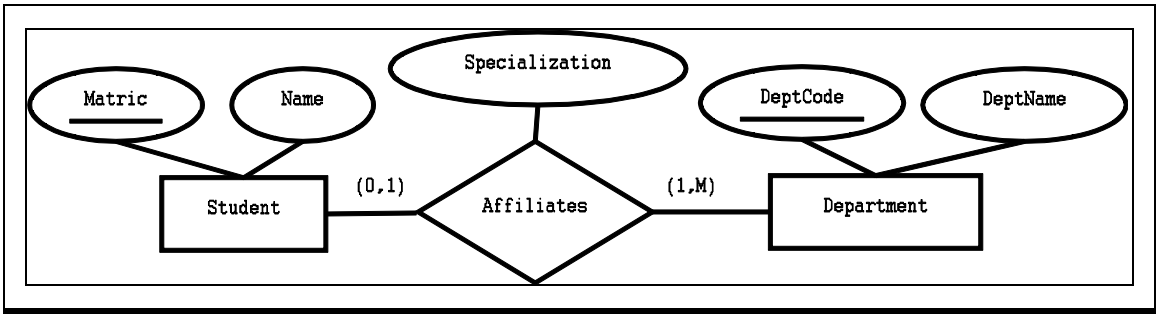


Figure 11. (0,1):(1,M) Students Affiliate to Departments

The attribute *Specialization* is not appropriately an attribute of *Student*, because students not affiliated with departments do not specialize. *Specialization* is similarly not an attribute of *Department*, since a department has many specializations. Specialization is thus, appropriately, an attribute of the relationship set *Affiliates*. However, if the E-R diagram is decomposed following the rule where foreign keys are placed in the "many" side, the *Specialization* attribute is placed in *Student*. Its value will be null for all students taking a general degree.

A relational database design that would address the problem of nulls would comprise [Mannino, 2011, Ullman, 1983]:

- Student [Matric, Name]
- Department [DeptCode, DeptName]
- Affiliates [Matric, DeptCode, Specialization]

Observe that in the implemented tables, *Affiliates* will have fewer records than *Student*.

Student	
Matric	Name
1000	Ian Farmer
1001	Annie Mulover
1002	Basketball Jones
1003	Nowan Stahdee

Department	
DeptCode	DeptName
BIO	Biology
IT	Info Tech

Affiliates		
Matric	DeptCode	Specialization
1000	BIO	Farming
1001	IT	Management
1002	BIO	Human Phys

A further demonstration that this is the appropriate decomposition can be obtained by performing a transformation of the relationship set of a (0,1):(1,M). The transformation of Figure 11 would produce the result in Figure 12. Note that in the figure, *Affiliates* is a weak entity set of *Student*, because its key depends on the key of student.

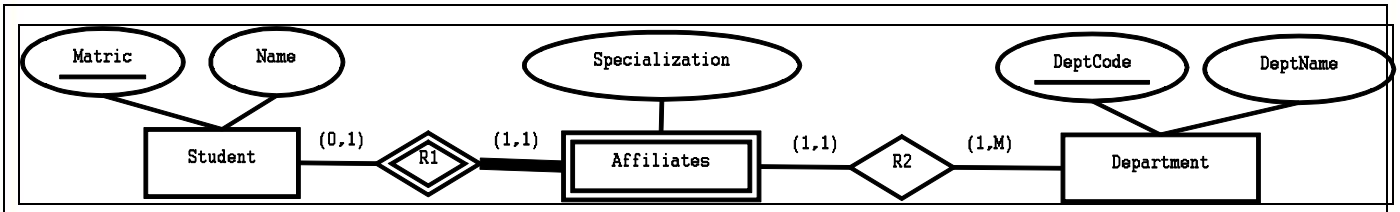


Figure 12. Transformation of Student Affiliates to Departments

A similar rule applies to the (0,1):(0,M) relationship set. For example, the university grants students the right to amnesty for a single course. A student taking amnesty is considered to have taken the course, but does not receive the grade assigned. A student may elect never to invoke his or her right to amnesty. An administrator types in notes about any student taking amnesty. The E-R diagram depicting this situation would be as shown in Figure 13, which would be transformed as shown in Figure 14.

This results in the following relations:

- Student [Matric, Name]
- Course [CNo, CourseName]
- InvokeAmnesty [Matric, CNo, Remarks]

Student	
Matric	Name
1000	Ian Farmer
1001	Annie Mulover
1002	Basketball Jones
1003	Nowan Stahdee

Course	
CNo	CourseName
BIO100	Intro to Bio
BIO110	Harder Bio

InvokeAmnesty		
Matric	CNo	Remarks
1000	BIO100	Student doing research
1002	BIO100	Spent too much time playing basketball
1003	BIO110	Student should not be in class

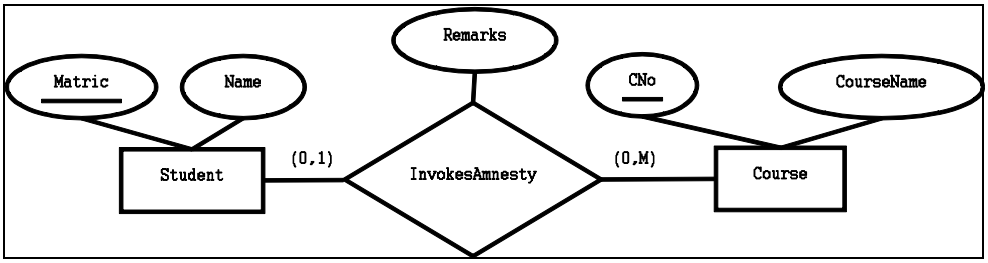


Figure 13. (0,1):(0,M) Student Invokes Amnesty for Course

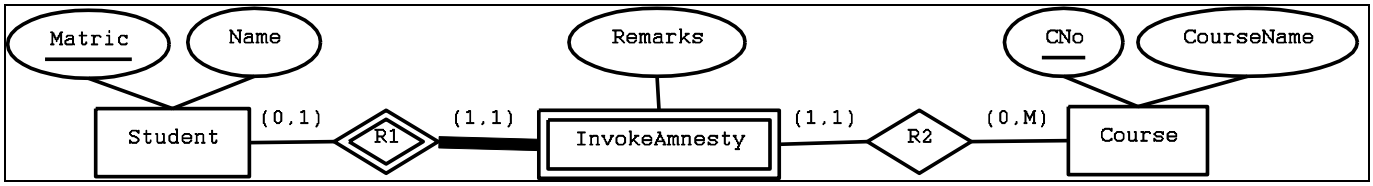


Figure 14. (0,1):(0,M) Transformation of Student Invokes Amnesty for Course

These relations demonstrate that:

> Guideline #4: *In a 1:M binary relationship set, where one entity set's participation is (0,1), the relationship set can be decomposed into an independent relation. The primary key of the entity set having the (0,1) participation in the relationship set is a candidate key and a foreign key of the relation mapping to the relationship set. The primary key of the entity set having the (0,M) participation to the relationship set is a foreign key in the relation mapping to the relationship set.*

Also:

> Guideline #5: *Attributes placed in a 1:M relationship set where one entity set has a (0,1) participation can not be rewritten by placing them in a participating entity set.*

Associative Entity Sets

Often, there is a belief that the associative entity set should only be employed when the maximum relationship participation is many-to-many. Table 4 presents examples of this belief.

However, consider an extended version of our earlier (0,1):(0,1) example. There are students (identified by a matriculation number), and athletes (identified by a sports council ID number). A student may or may not be (0,1) an athlete, and an athlete may or may not be (0,1) a student. Student-athletes always receive funding. Some student-athletes receive several kinds of funding. The funding amount differs among student-athletes. Figure 15 presents this E-R diagram.

Since an associative entity set is a simplified transformation of a relationship set, Figure 15 in transformed form would appear as Figure 16.

In Figures 15 and 16, it is the student-athlete (*CouldBe*) that receives the stipend, not the athlete or the student separately. Thus, the most appropriate place to put *Receives* and *Stipend* is in *CouldBe.* In contrast, consider the idea of Ph.D. students receiving several sources of funding. An E-R diagram incorporating associative entity sets would look like Figure 17.

Figure 17 transforms into Figure 18(a), which is informationally equivalent to Figure 18(b). Figure 18(b), however, is more information efficient than Figure 17.

The same issue of information equivalence can be demonstrated in the various permutations of binary 1:M relationship sets. Specifically, it is always possible to "move" a relationship set connected to an associative entity set to a participating (1,1) entity set. However, it is not possible to "move" a relationship set connected to an associative

Table 4: Textbook Definitions of Associative Entities	
Textbook	Quote
Coronel et al., 2011, p. 81	Fortunately, the problems inherent in the many-to-many (M:N) relationship can be easily avoided by creating a composite entity (also referred to as a bridge entity or an associative entity).
Dennis et al., 2009, p. 223	Intersection entities are added to a data model to store information about two entities sharing an M:N relationship.
Gillenson, 2005, p. 50	An occurrence of the Sale associative entity does exactly what the many-to-many relationship did.
Hoffer et al., 2011, p. 284	Is a many-to-many relationship actually an entity in disguise? Often the distinction between entity and relationship is simply a matter of how you view the data. An associative entity (sometimes called a gerund) is a relationship that the data modeler chooses to model as an entity type.
Hoffer et al., 2010, p. 118	How do you know whether to convert a relationship to an associative entity type? Following are four conditions that should exist: 1. All the relationships for the participating entity types are "many" relationships.
Kroenke and Auer, 2010, p. B-4	Like the crow's foot model, IDEF1X treats many-to-many relationships as poor stepchildren of nonidentifying connection relationships. This is done because such relationships have no direct expression in the relational model.
Satzinger et al., 2009, p. 184	Associative entity: A data entity that represents a many-to-many relationship between two other data entities.
Valacich et al., 2009, p. 220	Because many-to-many and one-to-one relationships may have associated attributes, the E-R diagram poses an interesting dilemma: Is a many-to-many relationship actually an entity in disguise? An associative entity is a relationship that the data modeler chooses to model as an entity type.
Watson, 2006 p. 114	When we get an m:m relationship, we create a third entity to link the entities through two 1:m relationships…. In this case, this third entity, typically known as an associative entity ….
Whitten and Bentley, 2007 p. 276	An associative entity is an entity that inherits its primary key from more than one other entity (called parents).

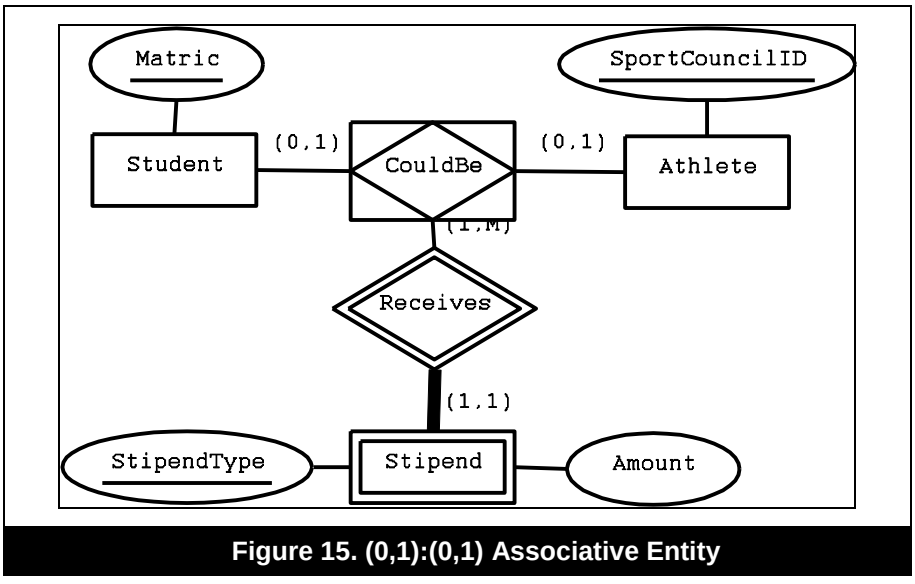


Figure 15. (0,1):(0,1) Associative Entity

entity set to a participating (0,1) entity set. The move can occur in the (1,1) case, because a relationship set always has a (1,1) participation to entity sets. Thus, the participation of the entity set and relationship set will be (1,1):(1,1). This condition does not hold for participating (0,1) entity sets. Hence:

> Guideline #6: *It is reasonable to use associative entity sets to represent binary relationship sets having any other participation than (1,1).*

In other words, associative entity sets are appropriately used for (0,1):(0,1), (0,1):(1,M), (0,1):(0,M) and all possible many-to-many relationship sets.

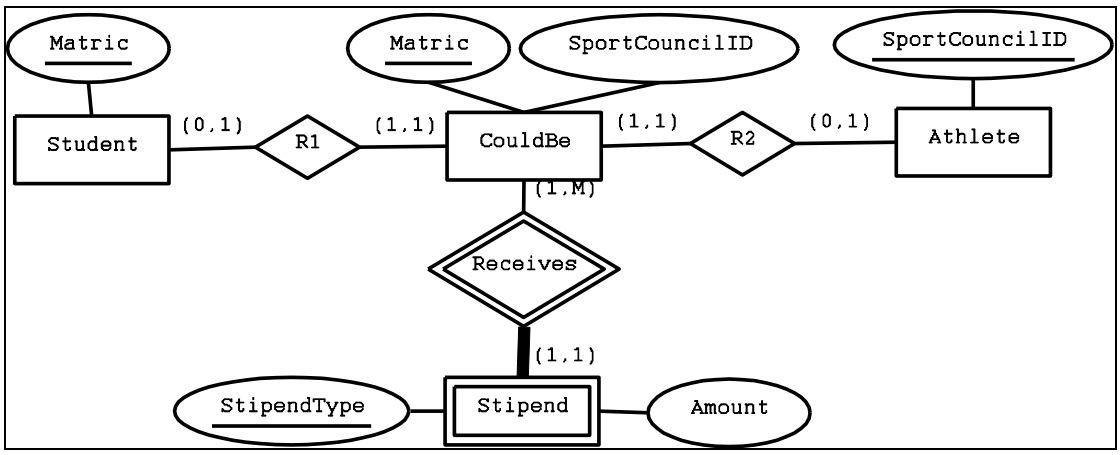


Figure 16. Transformation of Figure 15

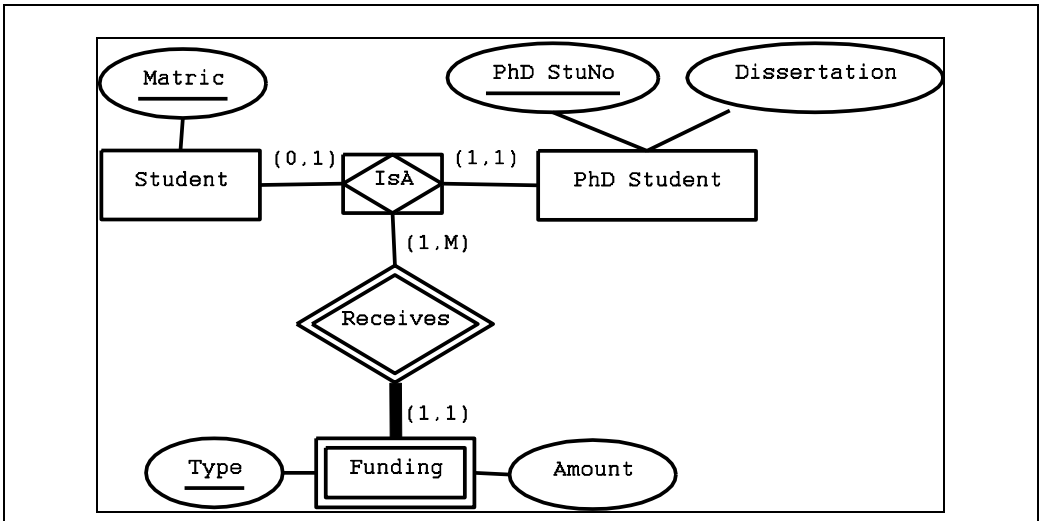


Figure 17. (0,1):(1,1) Associative Entity

IV. SPECIAL CASE RELATIONSHIP SETS

Unary Relationship Sets

Minimum participation on unary relationship sets impacts relational database design by impacting the structural validity of E-R diagrams. Simply put, certain kinds of unary relationship sets are not possible. This can be demonstrated by labeling the smaller (or equal) "side" of a relationship set as the "left" side, and labeling the other as the "right." For example, in Figure 19, which depicts the (0,1):(0,M) relationship set *Student Coaches Student*, the (0,1) side is designated as the "left" side, and the (0,M) side is designated the "right." It can be demonstrated that in certain cases, particular combinations of "left" and "right" participations are not possible [Dullea et al., 2003].

- (0,1):(1,1): Left allows that there can be no relationships in the relationship set, as every entity in the entity set could not participate. Right insists that at a minimum there must be a number of relationships in the relationship set equal to the number of entities. This is a contradiction.
- (1,1):(0,M): Left insists there must be a number of relationships in the relationship set equal to the number of entities. Right allows that there are no relationships in the relationship set, as every entity may elect not to participate. This is a contradiction.
- (1,1):(1,M): Left insists that at most the total number of relationships will always be equal to the number of entities in the entity set. Therefore, the total number of right will also be the number of entities in the entity set. It is, therefore, impossible to ever have many rights for one left.

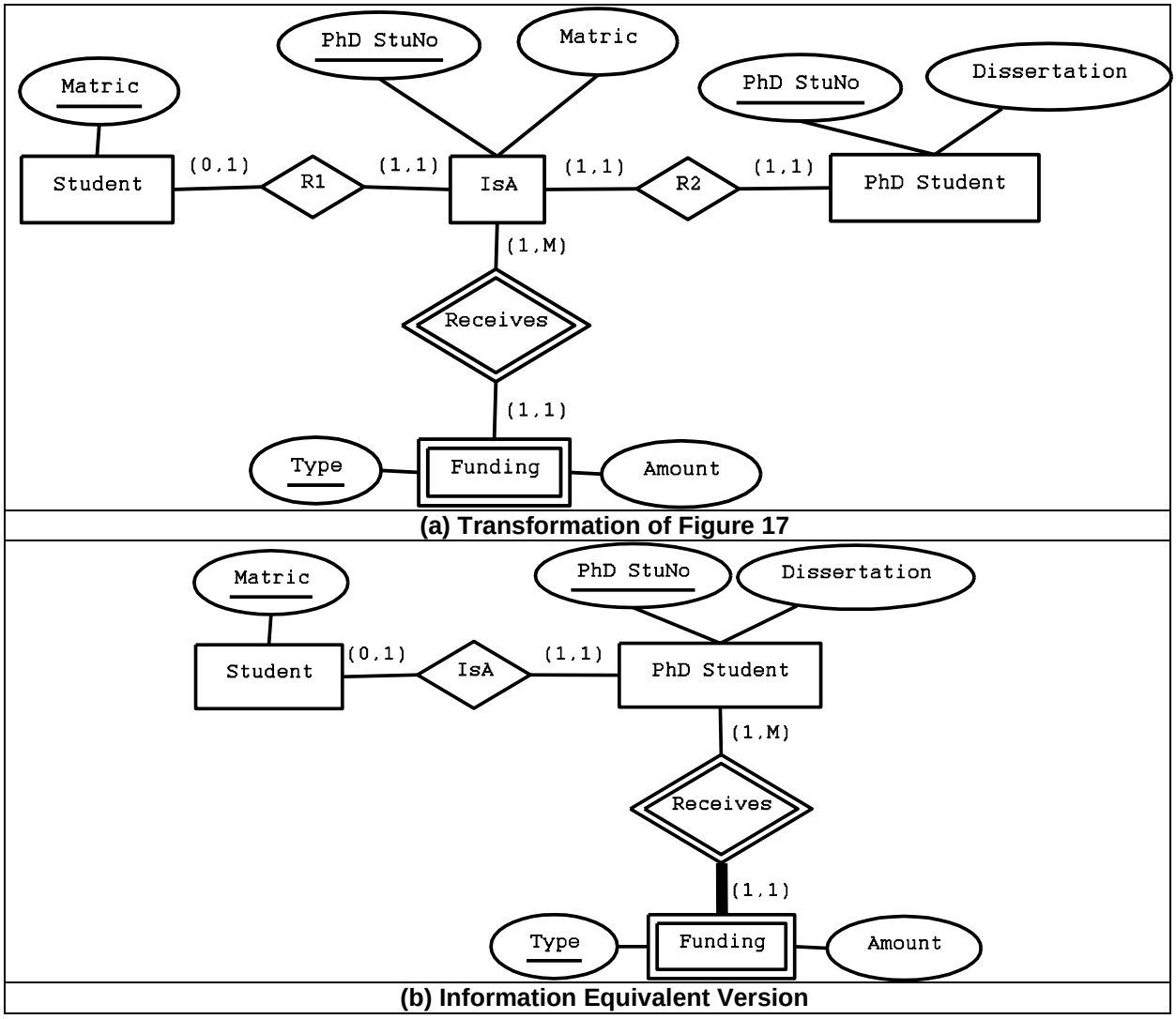


Figure 18. Various Equivalent (1,1):(0,M) Relationship Sets

A simple rule that summarizes these logically impossible unary relationship sets is:

Guideline #7: *No unary relationship set can have an entity set with a (1,1) participation.*

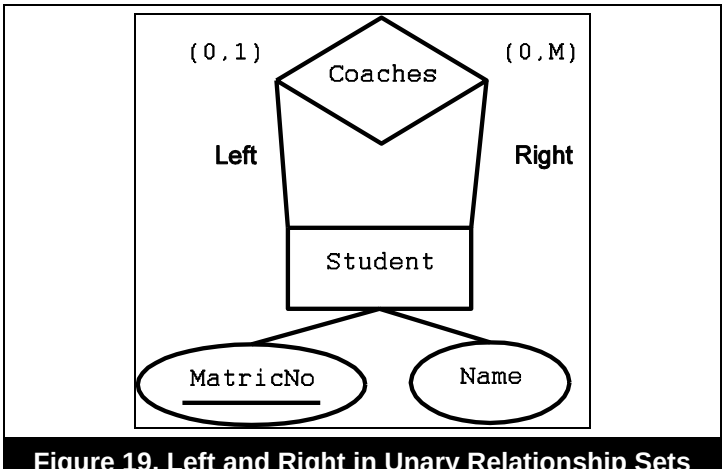


Figure 19. Left and Right in Unary Relationship Sets

In Section 3, it was highlighted that information efficiency concerns about placing attributes in relationship sets, or converting relationship sets to associative entity sets, only existed when one entity set had a participation of (1,1). Given that entity sets cannot have a (1,1) participation on unary relationship sets, this implies:

Guideline #8: *It is reasonable to have attributes of unary relationship sets. Similarly, it is reasonable to use associative entity sets to represent unary relationship sets.*

N-Ary Relationship Sets

Minimum participation also impacts n-ary relationship sets. For example, Ph.D. students may be allocated to zero or more projects. Each project can have zero or more Ph.D. students. Every time a Ph.D. student is allocated to a project, he or she is simultaneously allocated a faculty supervisor. The faculty supervisor only supervises that one Ph.D. student on that project. Furthermore, faculty supervisors must supervise Ph.D. students on projects. Ph.D. students have matric numbers and names. Faculty have employee IDs and names. Projects have project IDs and project names. For every Ph.D. Student/Faculty/Project group, there is a remarks column which is an evaluation of the student's work.

Figure 20 presents various information equivalent versions of this 3-ary relationship set. Figure 20(a) presents the E-R diagram as specified in the problem. Figure 20(b) is a transformation of Figure 20(a). Because *Faculty* and *Participates-in* have a (1,1):(1,1) relationship, Figure 20(c) is information equivalent.

Thus:

Guideline #9: *(1,1) participations are not information efficient for n-ary relationship sets, where n > 2.*

V. ISSUES WITH NORMALIZING E-R DIAGRAMS

One conceptual gap that is not always explained and illustrated is the relationship between entity-relationship diagrams, decomposition, and normalization. Why is it necessary to perform normalization on a decomposed entity-relationship diagram? It is generally assumed that, if E-R diagrams are correctly drawn, the relations to which the diagram decomposes are already normalized. Many tutorials discuss normalization in the context of functional dependencies. Strictly, the objective of normalization is to achieve domain key normal form (DKNF) [Fagin, 1981]. To achieve domain key normal form, one must consider not only functional dependencies, but also multivalue dependencies [Fagin, 1977]. Given that this tutorial is targeted at an introductory level, multivalue dependencies are not discussed here. Functional dependencies are relevant to four of the known normal forms intermediate to DKNF, 1st, 2nd, 3rd, and Boyce-Codd Normal Form (BCNF) [Codd, 1972].

In the relational data model, a functional dependency means that given an attribute or a set of attributes, one can determine the value of some other attribute. Thus, given the matriculation number (*Matric)*, one can determine the *Name*. Given the matriculation number (*Matric*) and club registration number (*Crno)*, one can obtain the *Rank*. There are two analogous constructs to the functional dependency in E-R diagrams- the (1,1) and (0,1) participation. A (1,1) participation means that, given an attribute or set of attributes, one can derive the value of some other attribute, and the value of that attribute will not be null. A (0,1) participation means that given an attribute or set of attributes, one can obtain the value of some other attribute, and the value of that attribute may be null. The various "many" relationships relate to multivalue dependencies.

In 1st, 2nd, 3rd, and Boyce-Codd Normal Form, the database designer employs the functional dependencies to design the database. The recommendations for these normal forms primarily, but not exclusively, recommend that the database designer structure the database by creating separate relations. Note, however, normalization principles cannot be wholly achieved by database structuring alone. In particular, 1st, 2nd, and 3rd normal form can be achieved solely through a careful structuring of the database, but Boyce-Codd normal form cannot.

In entity-relationship diagrams, structuring is carried out by carefully defining the entity and relationship sets. The rules for 1st to 3rd normal form are frequently incorporated in conceptual data model designs, because entity sets that do not incorporate 1st to 3rd normal form violate entity-relationship "best practice." An entity set that violates first normal form will have an array of attributes. If an entity set violates 2nd or 3rd normal form, there would be general agreement that the entity set should really be several other entity sets. As a result, a properly drawn entity-relationship diagram is generally already in 3rd normal form [Jajodia et al., 1983b].

Boyce-Codd normal form is a very subtle normal form. Its formal definition (every nontrivial dependency is a superkey) [Codd, 1974] is not helpful for understanding what it is about. Essentially, the Boyce-Codd normal form

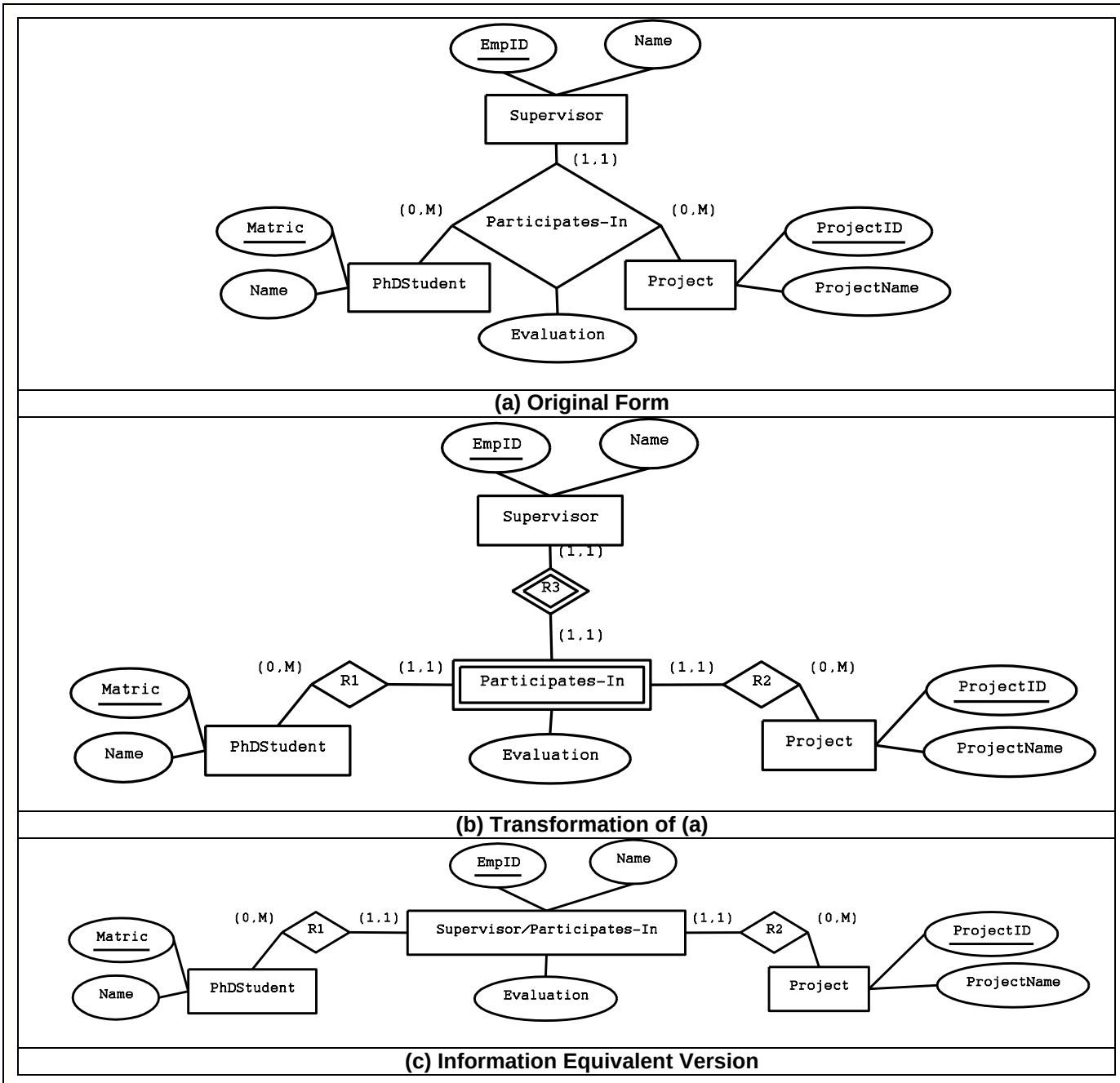


Figure 20. Ph.D. Students in Projects Have Supervisors

problem arises when a relation has at least two candidate keys, where the functional dependencies of the two candidate keys are different. The relation is in 3rd normal form if one of the candidate keys is the primary key, but is not in 3rd normal form when the other is the primary key.

To illustrate, consider the following example. Ph.D. students participate in many projects. Projects can have several Ph.D. students. Every Ph.D. student assigned to a project has an advisor. Every advisor has exactly one project. The functional dependencies of this example are:

1. Matric→ StudentName
2. ProjectID → ProjectName
3. Matric, ProjectID→ EmpID
4. EmpID → FacName, ProjectID

Note especially the functional dependencies in (3) and (4). ProjectID is functionally dependent on EmpID, but EmpID is partially dependent on ProjectID.

Many explanations of BCNF are similar to that below. First, the relations will be in 3NF. This will result in State 1.

1. PhDStudent [Matric, StudentName]
2. Project [ProjectID, ProjectName]
3. ParticipatesIn [Matric, ProjectID, EmpID]
4. Faculty [EmpID, FacName, ProjectID]

State 1: Relation in 3NF but not BCNF

It will then be highlighted that relation 3 (ParticipatesIn [Matric, ProjectID, EmpID]) has another candidate key, i.e., (Matric, EmpID). Thus (observe the underlines):

3. ParticipatesIn [Matric, ProjectID, EmpID]

Given functional dependency 4 (EmpID → FacName, ProjectID), this is a violation of 2NF. Therefore, the BCNF equivalent of State 1 is:

1. PhDStudent [Matric, StudentName]
2. Project [ProjectID, ProjectName]
3. ParticipatesIn [Matric, EmpID]
4. Faculty [EmpID, FacName, ProjectID]

State 2: Relation in 3NF but not BCNF

Where ProjectID is "moved" to relation 4, where it has always been.

The standard database design guidelines end the example at this point. The problem with State 2 is that, while it fully captures the structural aspect of the database design, it does not capture all the functional dependencies. Specifically, functional dependency 3 (Matric, ProjectID → EmpID) is not captured. Consider what happens if Faculty is joined with Participates-In. This results in the relation FacultyParticipatesIn [Matric, EmpID, FacName, ProjectID]. But based on functional dependency 3, a candidate key of this relation is Matric, ProjectID. This candidate key cannot be derived from the database structure.

To observe this, consider the following sample tables derived from this relational structure:

PhDStudent	
Matric	StudentName
1000	Ian Farmer
1001	Annie Mulover
1002	Basketball Jones
1003	Nowan Stahdee

Project		
ProjectID	ProjectName	
P1	BigFunded	
P2	Notsofunded	

ParticipatesIn	
Matric	EmpID
1000	B100
1001	B101

Faculty		
EmpID	FacName	ProjectID
B100	Mai Zuogang	P1
B101	Had Woking	P2
B102	Dr. Moebius	P1

Let us assume student 1000 (Ian Farmer) would like a second supervisor. Ian Farmer must take on Had Woking (B101) as a supervisor. He cannot take on Dr. Moebius (B102), because Dr. Moebius is working on the same project as Mai Zuogang (B100). This restriction is not identified in the *ParticipatesIn* relation, because the primary key is {*Matric*, *EmpID*}, and not {*Matric*, *ProjectID}*.

Thus, the proper representation of this example is four relations plus a constraint.

1. PhDStudent [Matric, StudentName]
2. Project [ProjectID, ProjectName]
3. ParticipatesIn [Matric, EmpID]
4. Faculty [EmpID, FacName, ProjectID]
5. Matric, ProjectID is a candidate key of ParticipatesIn join Faculty.

State 3: Correct Representation of Functional Dependencies

How does this impact E-R diagrams? First, assume the naïve view that a "properly drawn" E-R diagram always captures the final structural form of a relational database. The "structurally correct" E-R diagram will appear as in Figure 21.

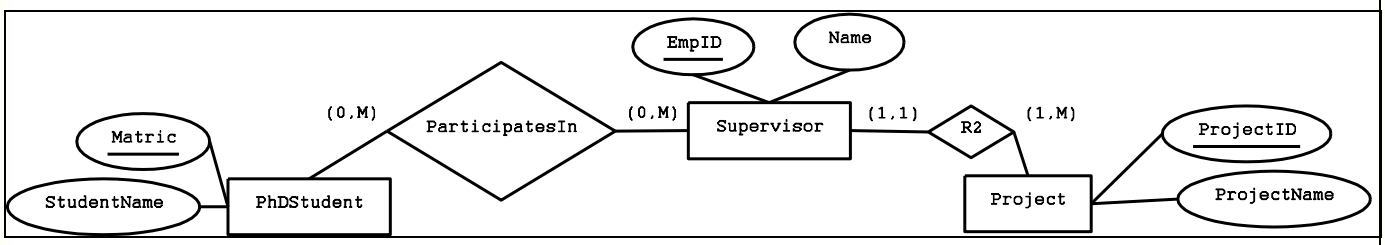


Figure 21. Structurally Representative E-R Diagram of BCNF

However, because this E-R diagram is structurally correct, it does not capture a critical requirement, namely that the EmployeeID can be identified by (is functionally dependent on) the student and project, and by implication Matric, and Project.

The E-R diagram that captures all the functional dependency requirements will be similar to Figure 22. Note that Figure 22 is a cycle, where supervisor connects to project "twice." One connection is via the *Supervises* relationship set, while the other is via *ParticipatesIn.* This cycle is explained via the two functional dependencies involving EmpID and ProjectID, i.e., Functional Dependency 3 (Matric, ProjectID → EmpID), and Functional Dependency 4 (EmpID → FacName, ProjectID). In fact, it can be demonstrated that, for all E-R diagrams capturing requirements where a BCNF version of the relations is not identical to a 3NF version, that the E-R diagram will always have at least one cycle [Jajodia et al., 1983a; Jajodia et al., 1983b].

If the E-R diagram of Figure 22 is decomposed, State 1, which is not in BCNF, is obtained. Thus:

> Guideline #10: *The link between normalization, decomposition, and E-R diagrams is as follows. The (0,1) and (1,1) participations of E-R diagrams relate to functional dependencies, while the "many" participations relate to multivalue dependencies. Furthermore, decomposition is not a substitute for normalization. There are cases where after decomposition of an E-R diagram, it is still not yet in DKNF.*

VI. GUIDELINE APPLICATION

The nuances of minimum participation and how they impact the relational database can be appreciated in at least three ways: (1) by highlighting the "science" of E-R diagrams, (2) as a replacement for existing decomposition rules, and (3) as a link between E-R diagrams and normalization.

Science of E-R Diagrams

The technical nature of database means that designers may generally seek a "mathematically correct" or math-logic answer. Given an E-R diagram, there exists exactly one decomposition. Given a set of functional dependencies, there is exactly one solution in Domain Key Normal Form. One can ascertain whether a relational algebra or SQL expression correctly resolves a query.

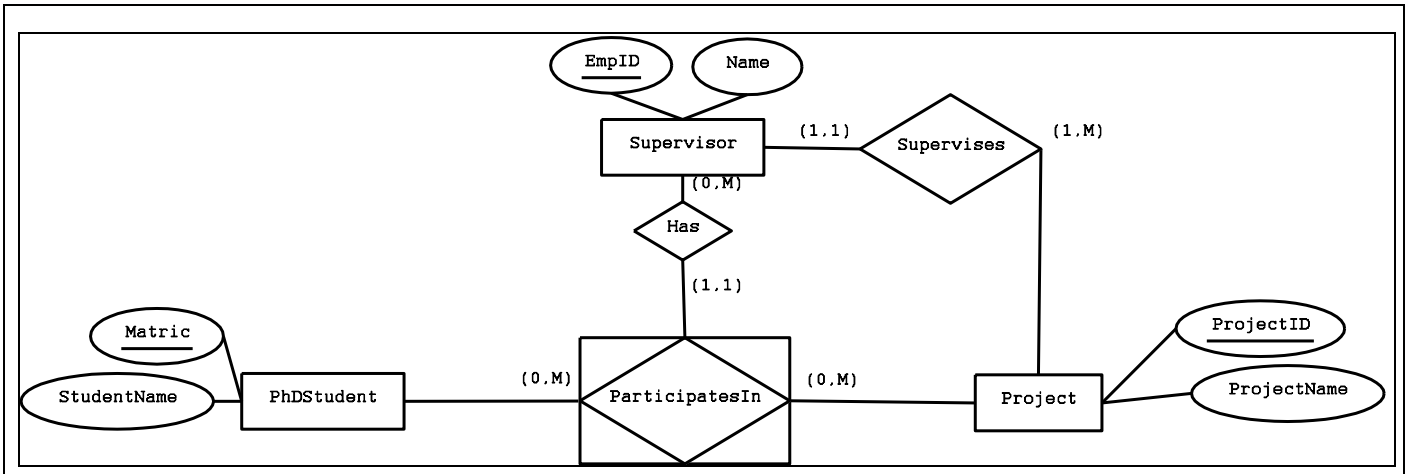


Figure 22. Version of E-R Diagram That Captures All Functional Dependencies

It is difficult to identify one best E-R diagram. Indeed, substantial empirical research is being performed to identify optimal E-R diagrams [Burton-Jones and Meso, 2002; Shanks et al., 2008; Shanks et al., 2004; Siau et al., 1997; Wand et al., 1999; Weber, 1997]. That research is based on theories of cognitive psychology and philosophy and helps to articulate why things are modeled in a particular way. The guidelines in this article are based on the mathematical notions of transformation, information efficiency and information equivalence.

Decomposition

There remain exactly three rules for decomposition:

1. For any relationship set that has a (1,1) participation: the primary key of the entity set having the (1,1) participation becomes the foreign key of the other entity set. In a (0,1):(1,1) situation, this also becomes a candidate key.
2. Otherwise, for any relationship set that has a (0,1) participation: The relationship set becomes a relation. Its candidate key is the primary key of the entity set having the (0,1) participation.
3. Otherwise (M:N relationship): The relationship set becomes a relation. Its candidate key is a subset of the primary keys of all participating entity sets.

E-R Diagrams and Normalization

There is often a disconnect in the discussion of E-R diagrams and normalization, where the link between decomposition and normalization is not properly articulated. A discussion of minimum participation will reveal that the relational database structure itself is insufficient for capturing the E-R diagram. Consider the decomposition of two one-to-many relationship sets where one has a (0,M) and the other a (1,M) participation. Structurally, both of these decompose to identical relations (see Figures 11–14 for illustrations). However, in one, the “many” side may not participate, while in the other, it must. A proper relational database logical design must add relational algebra constraints reflecting these disparities. This sets the stage for a later discussion on BCNF. BCNF highlights that the structure of the relations in the database is also insufficient to ensure proper normalization based on functional dependencies. Constraints must be included to properly model BCNF. Finally, BCNF itself demonstrates the importance of normalization above decomposition. In a BCNF situation, decomposition itself will not result in a properly designed database. Normalization is also needed.

VII. CONCLUSION

This article has argued that minimum participation in relationship sets is an important piece of information needed to ensure proper database design. A set of guidelines is proposed that focuses on modeling minimum participation. The guidelines derived from the analysis are intended to help practitioners and educators with complex conceptual design representations.

.

ACKNOWLEDGMENTS

This article has received help from numerous sources. We would like to thank Roger Chiang for comments and advice, Bill English for assistance with proofreading and copyediting, Sheryl Baster and the staff at the University of Auckland library for helping us to obtain the latest editions of various textbooks, and colleagues at the University of Auckland for their inputs and the loan of their textbooks.

This research was supported by the J. Mack Robinson College of Business, Georgia State University.

REFERENCES

Atzeni, P., et al. (1999) *Database Systems: Concepts, Languages and Architectures*, New York, NY: McGraw-Hill.

Bennett, S., S. McRobb, and R. Farmer (2010) *Object-Oriented Systems Analysis and Design Using UML*, *4th edition,* New York, NY: McGraw Hill.

Bernick, J.P. (2003) "A Translation of the One-to-One Relationship for Introductory Relational Database Courses", *Inroads—The SIGCSE Bulletin* (35)4, pp. 66–67.

Biskup, J. (1979) "A Formal Approach to Null Values in Database Relations", *Advances in Data Base Theory,* pp. 299–341.

Bodart, F., et al. (2001) "Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests", *Information Systems Research* (12)4, pp. 384–405.

Burton-Jones, A., and P. Meso. (2002) "How Good Are These UML Diagrams? An Empirical Test of the Wand and Weber Good Decomposition Model", *Proceedings of the Twenty-Third International Conference on Information Systems,* pp. 101–114.

Carter, J. (2003) *Database Design and Programming with Access, SQL, Visual Basic and ASP*, *2nd edition,* Berkshire, UK: McGraw-Hill.

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

Chen, P. P.-S. (1997) "English, Chinese and ER Diagrams", *Data and Knowledge Engineering* (23)1, pp. 5–16.

Codd, E.F. (1972) "Further Normalization of the Database Relational Model", *Data Base Systems,* pp. 65–98.

Codd, E.F. (1974) "Recent Investigations in Relational Database Systems", *Proceedings of the IFIP Congress,* pp. 1017–1021.

Connolly, T., and C. Begg (2010) *Database Systems: A Practical Approach to Design, Implementation, and Management*, *5th edition,* Essex, UK: Pearson.

Coronel, C., S. Morris, and P. Rob (2011) *Database Systems: Design, Implementation, & Management*, *9th edition,* Boston, MA: Cengage Learning.

Date, C.J. (2004) *An Introduction to Database Systems*, *8th edition*, Reading, MA: Addison-Wesley.

Dennis, A., B.H. Wixom, and R.M. Roth (2009) *Systems Analysis and Design*, Hoboken, NJ: John Wiley & Sons.

Dey, D., V.C. Storey, and T.M. Barron (1999) "Improving Database Design Through the Analysis of Relationships", *ACM Transactions on Database Systems* (24)4, pp. 453–486.

Dullea, J., I.-Y. Song, and I. Lamprou (2003) "An Analysis of Structural Validity in Entity-Relationship Modeling", *Data & Knowledge Engineering* (47)2, pp. 167–205.

Elmasri, R., and S.B. Navathe (2011) *Fundamentals of Database Systems*, *6th edition,* San Francisco, CA: The Benjamin/Cummings Publishing Company Inc.

Fagin, R. (1977) "Multivalued Dependencies and a New Normal Form for Relational Databases", *ACM Transactions on Database Systems* (2)3, pp. 262–278.

Fagin, R. (1981) "A Normal Form for Relational Databases That Is Based on Domains and Keys", *ACM Transactions on Database Systems* (6)3, pp. 387–415.

Fahrner, C., and G. Vossen (1995) "A Survey of Database Design Transformations Based on the Entity-Relationship Model", *Data and Knowledge Engineering* (15)3, June, pp. 213–250.

Gillenson, M.L. (2005) *Fundamentals of Database Management*, Hoboken, NJ: John Wiley and Sons.

Goldstein, B.S. (1981) "Constraints on Null Values in Relational Databases", *Proceedings of the Very Large Database (VLDB) Conference,* pp. 101–110.

Grant, J. (1977) “Null Values in a Relational Data Base”, *Information Processing Letters* (6)5, pp. 156–157.

Hoffer, J.A., J.F. George, and J.S. Valacich (2011) *Modern Systems Analysis and Design, 6th edition,* Upper Saddle River, NJ: Pearson.

Hoffer, J.A., V. Ramesh, and H. Topi (2010) *Modern Database Management*, *10th edition,* Upper Saddle River, NJ: Pearson.

Jajodia, S., P.A. Ng, and F.N. Springsteel (1983a) “Entity-relationship Diagrams Which Are in BCNF”, *International Journal of Parallel Programming* (12)4, pp. 269–283.

Jajodia, S., P.A. Ng, and F.N. Springsteel (1983b) “The Problem of Equivalence for Entity-Relationship Diagrams”, *IEEE Transactions on Software Engineering* (9)5, pp. 617–630.

Kroenke, D.M. (2011) *Using MIS*, *3rd edition,* Upper Saddle River, NJ: Pearson.

Kroenke, D.M., and D.J. Auer (2010) *Database Processing: Fundamentals, Design & Implementation*, *11th edition,* Upper Saddle River, NJ: Prentice Hall.

Mannino, M.V. (2011) *Database: Design, Application, Development, & Administration*, *5th edition,* New York, NY: McGraw-Hill.

Markowitz, V.M., and J.A. Makowsky (1990) “Identifying Extended Entity-Relationship Object Structures in Relational Schemas”, *IEEE Transactions on Software Engineering* (16)8, pp. 777–790.

Markowitz, V.M., and A. Shoshani (1992) “Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach”, *ACM Transactions on Database Systems* (17)3, pp. 423–464.

Navathe, S.B. (1992) “Evolution of Data Modeling for Databases”, *Communcations of the ACM* (35)9, pp. 112–123.

Nijssen, G.M., and T. Halpin (1989) *Conceptual Schema and Relational Database Design*, Upper Saddle River, NJ: Prentice Hall.

Ricardo, C.M. (2011) *Databases Illuminated*, *2nd edition,* Boston, MA: Jones and Bartlett.

Satzinger, J., R. Jackson, and S. Burd (2009) *Systems Analysis & Design in a Changing World*, Boston, MA: Course Technology.

Shanks, G., et al. (2008) "Representing Part-Whole Relations in Conceptual Modeling: An Empirical Evaluation”, *MIS Quarterly* (32)3, pp. 553–573.

Shanks, G., E. Tansley, and R. Weber (2004) “Representing Composites in Conceptual Modeling”, *Communications of the ACM* (47)7, pp. 77–80.

Siau, K.L., H.C. Chan, and K.K. Wei (2004) “Effects of Query Complexity and Learning on Novice User Query Performance with Conceptual and Logical Database Interfaces”, *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans* (34)2, pp. 276–281.

Siau, K.L., Y. Wand, and I. Benbasat (1997) “The Relative Importance of Structural Constraints and Surface Semantics in Information Modeling”, *Information Systems* (22)2/3, pp. 155–170.

Silberschatz, A., H.F. Korth, and S. Sudarshan (2011) *Database System Concepts*, *6th edition,* New York, NY: McGraw Hill.

Song, I.-Y., M. Evans, and E.K. Park (1995) “A Comparative Analysis of Entity-Relationship Diagrams”, *Journal of Computer and Software Engineering* (3)4, pp. 427–459.

Storey, V.C. (1991) “Relational Database Design Based on the Entity-Relationship Model”, *Data and Knowledge Engineering* (7)1, pp. 47–83.

Teorey, T.J., D. Yang, and J.P. Fry (1986) “A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model”, *ACM Computing Surveys* (18)2, pp. 197–222.

Ullman, J.D. (1983) *Principles of Database Systems*, New York, NY: W.H. Freeman & Co.

Ullman, J.D., and J. Widom (2002) *A First Course in Database Systems*, Upper Saddle River, NJ: Prentice Hall.

Valacich, J.S., J.F. George, and J.A. Hoffer (2009) *Essentials of Systems Analysis and Design, 4th edition,* Upper Saddle River, NJ: Pearson.

Wand, Y., V.C. Storey, and R. Weber (1999) “An Ontological Analysis of the Relationship Construct in Conceptual Modeling”, *ACM Transactions on Database Systems* (24)4, pp. 494–528.

Watson, R.T. (2006) *Data Management: Databases and Organizations*, *5th edition,* New York, NY: John Wiley & Sons.

Weber, R. (1997) "The Link Between Data Modeling Approaches and Philosophical Assumptions: A Critique”, *Third Americas Conference on Information Systems, 1997*, pp. 306–308.

Whitten, J.L., and L.D. Bentley (2007) *Systems Analysis and Design Methods*, *7th edition,* Boston, MA: Irwin.

Wintraecken, J.J.V.R. (1989) *The NIAM Information Analysis Method: Theory and Practice*, Dordrecht, The Netherlands: Kluwer Academic Publishers.

ABOUT THE AUTHORS

Cecil Eng Huang Chua is an associate professor at the University of Auckland. He received a Ph.D. in Information Systems from Georgia State University, a Masters of Business by Research from Nanyang Technological University and both a Bachelor of Business Administration in Computer Information Systems and Economics and a Masters Certificate in Telecommunications Management from the University of Miami. His research has been published in such journals as *Decision Support Systems, Journal of the Association for Information Systems, MIS Quarterly,* and the *VLDB Journal.*

Veda C. Storey is the Tull Professor of Computer Information Systems, J. Mack Robinson College of Business Administration, Georgia State University. She has research interests in intelligent information systems, the Semantic Web, and design science research. Her research has been published in *Information Systems Research*, *ACM Transactions on Database Systems*, *IEEE Transactions on Knowledge and Data Engineering* and the *Journal of Data Management*. She has served on the boards of several journals including *Information Systems Research*, the *Journal of Data Management*, *MIS Quarterly*, *Data Base*, and *Decision Support Systems*.

Communications of the Association for Information Systems

ISSN: 1529-3181

Communications of the Association for Information Systems