

Improving Database Design through the Analysis of Relationships

DEBABRATA DEY

University of Washington

VEDA C. STOREY

Georgia State University

and

TERENCE M. BARRON

University of Toledo

Much of the work on conceptual modeling involves the use of an entity-relationship model in which binary relationships appear as associations between two entities. Relationships involving more than two entities are considered rare and, therefore, have not received adequate attention. This research provides a general framework for the analysis of relationships in which binary relationships simply become a special case. The framework helps a designer to identify ternary and other higher-degree relationships that are commonly represented, often inappropriately, as either entities or binary relationships. Generalized rules are also provided for representing higher-degree relationships in the relational model. This uniform treatment of relationships should significantly ease the burden on a designer by enabling him or her to extract more information from a real-world situation and represent it properly in a conceptual design.

Categories and Subject Descriptors: H.1.0 [**Models and Principles**]: General; H.2.1 [**Database Management**]: Logical Design—*Data models*; H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*

General Terms: Design

Additional Key Words and Phrases: Conceptual model, ER model, integrity constraint, min-max cardinality, relationship degree, weak relationship

1. INTRODUCTION

Conceptual design has long been recognized as the most crucial phase of the database design process, with further development of database technology

Authors' addresses: D. Dey, University of Washington, Seattle, WA; V. C. Storey, Georgia State University, Atlanta, GA; T. M. Barron, University of Toledo, Toledo, TX.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 2000 ACM 0362-5915/99/1200-0453 \$5.00

not likely to change that situation Batini et al. [1992, p. 11]; Ramesh and Browne [1999]. During conceptual design the information requirements of different end-users are elicited and represented in a conceptual model; this enables the designer to obtain a system-independent global view of a given application. At this stage the task of a database designer is to understand, structure, and represent the meaning of relevant real-world objects. Hence, models are needed that naturally and directly reflect the user's conception of the real world [Mattos et al. 1992-93].

The entity-relationship (ER) model, originally proposed by Chen [1976], has been used widely for conceptual design Lenzerini and Nobili [1990].¹ In the ER model, real-world objects are represented as *entities*, and associations among entities are captured by *relationships*. Properties of entities and relationships are represented as *attributes*. The number of entities participating in a relationship is usually referred to as the *degree* of the relationship. Since its inception, the ER model has undergone many additions and enhancements that increase its expressive power and obtain better conceptual designs [Batini et al. 1992; Biskup 1995; Hull and King 1987].

In conceptual modeling, entities are usually considered to be the most important component. This is partially evident from the existence of many special constructs for entities, such as a *weak* entity, *composite*, or *aggregate* entity, and *generalization* hierarchy. Relationships have not received the same amount of attention in the database design literature [Batra and Zanakis 1994; Siau et al. 1995]. Ternary and other higher-degree relationships, in particular, have received very little rigorous analysis; the primary focus has been on binary relationships [Batra and Zanakis 1994; McAllister and Sharpe 1998]. Supporters of the binary models advocate that higher-degree relationships are rare in real-world applications and, when they occur, can be represented by multiple binary relationships [Batini et al. 1992, p. 45]. However, there is ample evidence that these higher-degree relationships may occur frequently in some real-world situations, and many database designers, in the absence of a general framework for representing higher-degree relationships, tend to represent them, often inappropriately, as binary relationships [Batra and Antony 1994; Batra and Zanakis 1994]. Furthermore, the semantics of a higher-degree relationship is not always the same as multiple binary relationships or a gerund [Song et al. 1995]. Finally, even if a higher-degree relationship can be represented by multiple binary relationships, the former often provides a more "natural" representation of the real world [Batini et al. 1992, p. 45].

The "naturalness" of a representation depends upon the application being modeled and the users' perceptions of it. Consequently, naturalness is a relative and context-dependent concept. However, irrespective of what is perceived to be "natural," a conceptual model should have sufficient rich-

¹The NIAM model Nijssen and Halpin [1989], although not as widely used as the ER model, has received some attention as a conceptual model; Laender and Flynn [1993] provide a useful overview and comparison of the two.

ness so that all possible user perceptions can be modeled. Moreover, there must be a systematic process for translating a conceptual design into a logical database design. Thus, a generalized framework for representing relationships is needed.

Empirical evidence [Batra and Antony 1994; Batra et al. 1990; Goldstein and Storey 1989; Ramesh and Browne 1999], as well as our own practical experience, suggests that the misrepresentation of relationships is a common error in the design process; such mistakes in the representations of entities and attributes are less frequent. According to Batra and Antony [1994], part of the problem may be that, given a set of entities, there are many different possible configurations of relationships. We believe that the lack of a generalized framework for the representation and analysis of relationships is also partially responsible for these design errors. It should be possible to reduce these errors significantly by developing a uniform treatment for all types of relationships. Furthermore, this framework is useful in developing database design systems where the conceptual modeling is done by a machine, without the benefit of the intervention of a human designer [Lloyd-Williams 1993; Storey and Goldstein 1988; Storey and Goldstein 1993; Storey et al. 1997].

In developing a design framework based on relationships, the issue of completeness is a critical one. There are four important aspects to consider when analyzing relationships: (1) cardinalities or mapping ratios; (2) the degree of a relationship; (3) the recursive nature of a relationship (i.e., an entity participating more than once in a relationship); and (4) interrelationship constraints. Most real-world situations can be modeled by analyzing these four aspects [Batini et al. 1992; Rochfeld and Negros 1992-93]. Semantic relationships such as generalization/specialization (is-a) are also captured by this framework. For example, a specialization hierarchy can be represented by binary relationships between the generic and the specific entity types having mapping ratios of (1,1) on the specialization side and (0,1) on the generalization side, together with interrelationship constraints that specify the overlapping-disjoint/partial-total features of the hierarchy. The objective of this research is to provide a uniform analysis of all the four aspects mentioned above. We note that the completeness of a database design approach is ultimately an empirical issue. By applying this approach in real-world situations, one could address the necessity, completeness, and usability of different constructs for conceptual modeling. The specific contributions of our research are as follows:

- A generalized analysis of relationships is presented, in which common binary relationships become a special case (Section 2). This should eliminate the usual bias, as observed by Batra and Antony [1994], towards binary relationships.
- Guidelines are provided for identifying ternary and higher-degree relationships that are commonly misrepresented as binary relationships (Section 3).

- The existence dependence of a relationship on other relationships is analyzed and rules are provided for identifying derived relationships (Section 4). In this context, a special relationship construct, called the *weak* relationship, is introduced and its semantics analyzed.
- The alternative representation of a higher-degree relationship as several binary relationships is analyzed (Section 5).
- Generalized notations are developed for recursive relationships so that they can be treated in a uniform manner under our framework (Section 6). In doing so, we also identify a particular class of recursive relationships called *symmetric recursive relationships*.
- Basic design implications obtained from our analysis are stated so that they can be useful to a designer. They are explicit enough to be incorporated into an automated database design tool.
- Rules are provided for representing higher-degree relationships in the relational model (Section 7).

The results of our research have been implemented as a prototype database design system, called the *Database Designer*; it is implemented using Microsoft Visual Basic and runs on the Microsoft Windows platform. The system implements most of the important results discussed in this paper. Currently, the system acquires an ER model, checks it for consistency and appropriateness, makes suggestions for improvement, and generates the SQL code for the creation of the relational structure. Minor improvements to user interfaces, and implementation of other results from previous work (not explicitly discussed here), would yield a fully working database design system.

2. BASIC NOTATION FOR RELATIONSHIPS

An *entity instance* is a “thing” or an object that has a separate identity in the real world. Each entity instance possesses certain properties; these are usually referred to as *attributes*. An *entity type* is a set of entity instances with a similar set of properties. For brevity, here we refer to an entity type as an *entity*; an entity instance is specifically distinguished from an entity type. A *relationship instance* is an association among several entity instances. A *relationship type* (or a *relationship* in short) is a set of similar relationship instances. Consider, for example, the relationship shown in Figure 1. There are four entities (DOCTOR, PRESCRIPTION, PATIENT, and DRUG) connected by this relationship; it therefore has a *degree* of four [Elmasri and Navathe 1994]. Each relationship instance can be viewed as a tuple, {dr,pt,rx,dg}, where “dr” is an instance of DOCTOR, “pt” is an instance of PATIENT, “rx” is an instance of PRESCRIPTION, and “dg” is an instance of DRUG. Table I shows some sample instances of the relationship and the participating entities. In Figure 1, the numbers next to the line joining an entity with the relationship are the min-max cardinalities representing the participation constraints of an entity in the relationship [Batini et al. 1992,

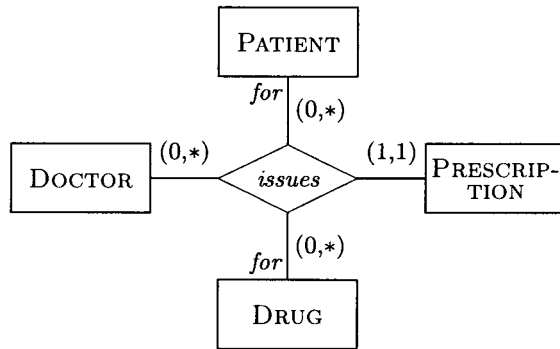


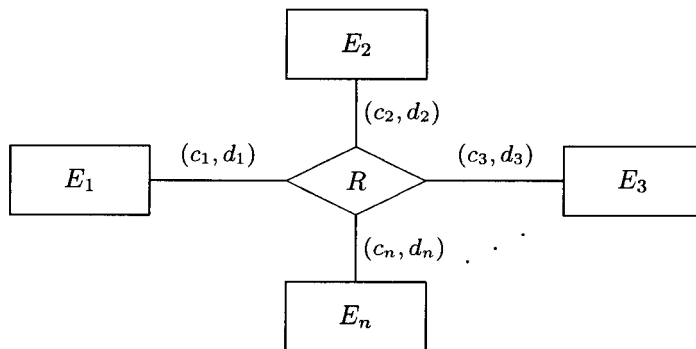
Fig. 1. A relationship of degree four.

Table I. Example Instances of Relationship in Figure 1 and Participating Entities

DOCTOR	PATIENT	PRESCRIPTION	DRUG
Davis	Perry	prx01	Drefludan
Dickert	Peters	prx02	Drisperdol
Deng	Porter	prx03	Drenova
Dougan	Phang	prx04	
	Phang	prx04	
		prx05	
		prx06	
<i>issues</i>			
Davis	Porter	prx01	Drisperdol
Davis	Phang	prx02	Drefludan
Dougan	Peters	prx03	Drefludan
Davis	Porter	prx04	Drefludan
Dougan	Peters	prx05	Drefludan
Deng	Phang	prx06	Drisperdol

p. 23]. For example, the numbers $(0,*)$ next to DOCTOR indicate that a particular doctor may not participate in the relationship at all (such as Dickert), or may do so many $(*)$ times (such as Davis). So given an instance of DOCTOR, there may be no relationship instance for that doctor, or there may be several. Similarly, the numbers $(1,1)$ next to PRESCRIPTION indicate that, for each prescription instance, there is exactly one instance of the relationship.

Formally, given entities E_1, E_2, \dots, E_n , a relationship R among them is a set of ordered n -tuples $\langle e_1, e_2, \dots, e_n \rangle$ such that $e_i \in E_i$; i.e., $R \subset E_1 \times E_2 \times \dots \times E_n$. The *degree* of a relationship is the number of entities connected to it. Figure 2 shows a relationship R of degree n . If $\mathcal{E} \subset \{E_1, E_2, \dots, E_n\}$, then the restriction of R on \mathcal{E} , written $R[\mathcal{E}]$, is the set $\{r[\mathcal{E}] \mid r \in R\}$, where $r[\mathcal{E}]$ is a subtuple containing only instances of entities in \mathcal{E} (in the appropriate order).

Fig. 2. A relationship of degree n .

Let E_i be an entity participating in a relationship R . Not all instances of E_i need to participate in R . Given an instance $e \in E_i$, we define $R_{(E_i, e)} \subset R$ as the set of all ordered tuples containing e in the appropriate position of E_i . By $E'_i(R) \subset E_i$, we define the set of all instances of E_i that participate in R at least once. In other words, $E'_i(R) = \{e \mid R_{(E_i, e)} \neq \emptyset\}$. When there is no room for confusion, we write E'_i in place of $E'_i(R)$. The minimum (lower) and maximum (upper) cardinalities of E_i with respect to R are given by (respectively):

$$m_L(E_i, R) = \min_{e \in E_i} |R_{(E_i, e)}| \text{ and } m_U(E_i, R) = \max_{e \in E_i} |R_{(E_i, e)}|,$$

where $|X|$ stands for the number of elements in set X . For example, $m_L(E_i, R) = c_i$ and $m_U(E_i, R) = d_i$ in Figure 2. The min-max cardinalities can be any nonnegative integer; however, three values are generally accepted as the most useful: zero (0), one (1), and “many” (*, an *unrestricted* value of more than one). Based on the min-max cardinalities of the participating entities, a relationship can be classified into one of three categories:

- *Functional relationship*: If a relationship involves an entity with (1,1) min-max cardinalities, it is a functional relationship. In Figure 1, the min-max cardinalities of PRESCRIPTION are (1,1). This implies that, given an instance of PRESCRIPTION, we can find exactly one instance of the relationship “issues,” and hence, exactly one instance of each of DOCTOR, PATIENT, and DRUG. Therefore, the relationship can be viewed as a function, $issues: PRESCRIPTION \rightarrow DOCTOR \times PATIENT \times DRUG$.
- *Partial functional relationship*: A partial functional relationship is one in which no entity has (1,1) min-max cardinalities, but at least one entity has min-max cardinalities of (0,1). An example is shown in Figure 3, where a student must choose a faculty advisor at the time of selecting his/her major area. A student is not allowed to have more than one major, and some students may not have chosen their area yet. Denoting

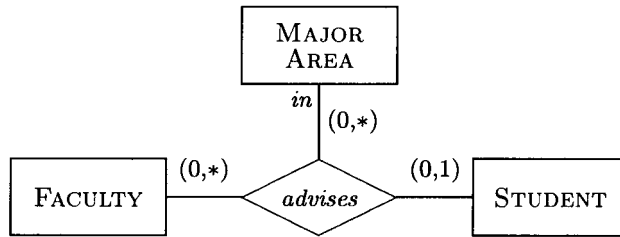


Fig. 3. A partial functional relationship.

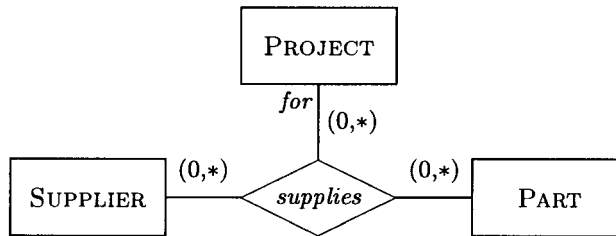


Fig. 4. A general relationship.

by S' the set of students who have already chosen their major area, we can express the relationship as a function, $advises: \rightarrow FACULTY \times MAJOR\ AREA$.

- *General relationship*: All other relationships fall into this category; i.e., a general relationship is one in which the min-max cardinalities of a participating entity is neither (1,1) nor (0,1), as shown in Figure 4.

A relationship may have attributes [Elmasri and Navathe 1994; Tsichritzis and Lochovsky 1982]; these are the properties of the combination of *all the participating entities*, and not just of a few of them. In other words, an attribute A is a function from the participating entity instances to the value-set V of the attribute, $A : E'_1(R) \times E'_2(R) \times \dots \times E'_n(R) \rightarrow V$. For example, the relationship in Figure 1 may have an attribute “date,” which specifies the date on which a prescription is issued.

In the ER model, a relationship and its participating entities are sometimes aggregated into a single concept called an *aggregate* or *composite* entity [Elmasri and Navathe 1994, p. 636; Smith and Smith 1977]. The composite entity can then participate in other relationships. The attributes of the relationship are also the attributes of the composite entity. Consider the relationship “offers” in Figure 5. This relationship is also represented as a composite entity SECTION. Relationship attributes such as “final-exam-date” could be considered as attributes of SECTION. In addition, SECTION participates in two separate relationships with STUDENT and TEXTBOOK. Note that STUDENT and TEXTBOOK must be related to the composite entity

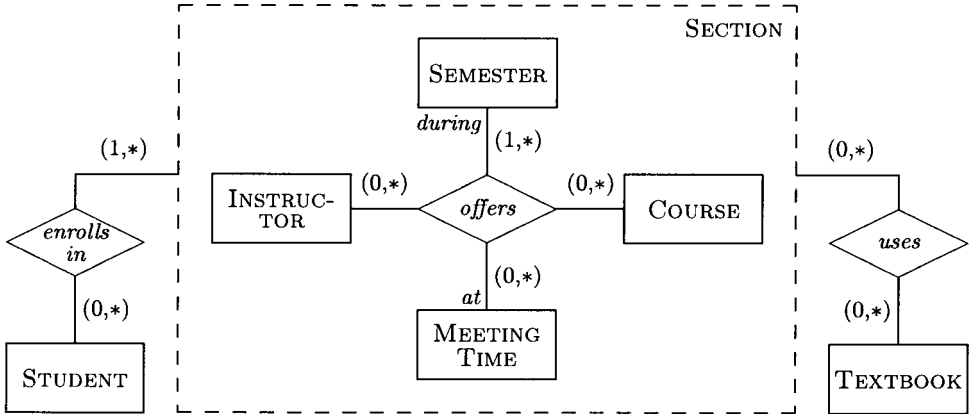


Fig. 5. SECTION as a composite entity.



Fig. 6. An alternative representation of the relationship in Figure 4.

SECTION only, and not to just one of its constituents. This is because a textbook is used for (or a student decides to enroll in) a specific course taught by a specific instructor during a specific semester. The same course taught by two different instructors (or during two different semesters) may use different textbooks.

3. ASSESSING THE DEGREE OF A RELATIONSHIP

The designer’s lack of familiarity with higher-degree relationships might result in inappropriate representations. Batra and Antony [1994] found that design errors pertaining to the degree of a relationship occur more frequently than connectivity errors. Their work suggests that there is a greater tendency to model higher-degree relationships as lower degree relationships than vice versa. It appears that the designer’s familiarity with binary relationships create an “availability bias” towards these relationships over those of higher-degree.

To illustrate the types of mistakes a designer is likely to make, consider the relationship in Figure 4, and its alternative representation in Figure 6 where this situation is modeled as two independent binary relationships. These two representations are not equivalent [Elmasri and Navathe 1994], so only one of them can be appropriate in a specific situation. To understand why, assume that Figure 6 is the appropriate representation for a given application. Then the representation in Figure 4 results in a violation of 4NF in the corresponding relational representation. Alternatively, if Figure 4 is the appropriate representation, then some spurious relationship

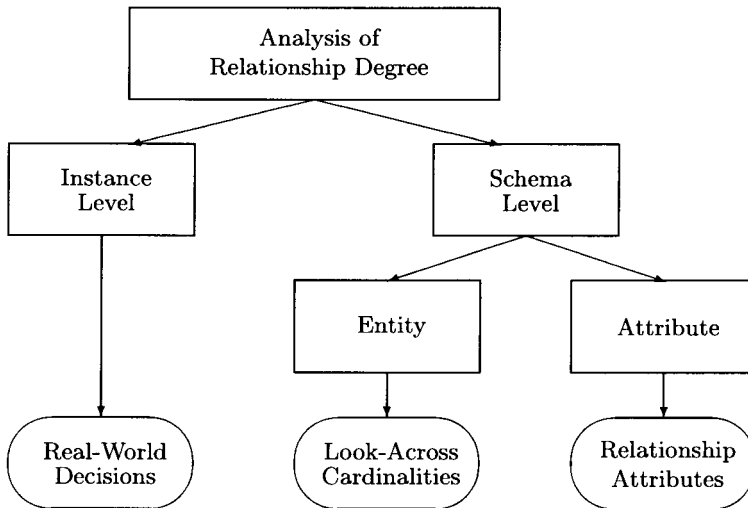


Fig. 7. A framework for assessing the degree of a relationship.

instances may be introduced when the decomposed view shown in Figure 6 is joined. As a result, the choice between an n -ary relationship and several binary ones is often “tricky,” and the appropriateness of the conceptual design depends heavily on the semantics of the application [Elmasri and Navathe 1994, pp. 60-62]. Clearly, guidelines are needed to assist the designer in identifying the correct degree of a relationship.

As shown in Figure 7, the correctness of the degree of a relationship could be analyzed using two approaches: (i) the analysis of the constructs that make up the relationship (schema-level analysis), and/or (ii) the analysis of the materialized instances of the relationship (instance-level analysis). A materialized relationship instance is a combination of several real-world entity instances; whether these entity instances should be combined together into a relationship instance can be answered by verifying whether there exists a real-world decision/event that combines them; Section 3.1 discusses this aspect.

Structurally (at the schema level), a relationship has two main components: participating entities and relationship attributes. Relationship attributes are properties of all the participating entities and can provide important information about the degree of a relationship as shown in Section 3.2. A participating entity, by itself, does not shed any light on the degree of the relationship. However, how an entity associates with another provides important consistency checks about the overall degree of the relationship McAllister [1998]. This is why it is necessary to examine pairwise look-across cardinalities for relationships; Section 3.3 discusses this issue.

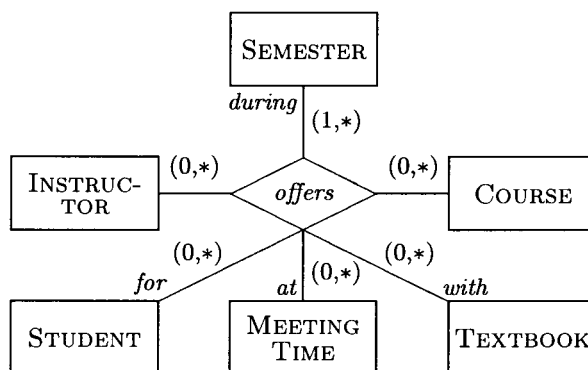


Fig. 8. A relationship of degree six.

3.1 Relationship Instances as Materialization of Real-World Decisions

The choice between a higher-degree relationship and several binary ones depends on a real-world decision and its implementation, of which we may view a relationship instance to be the result. Consider, for example, the alternate representation of Figure 5 as a relationship of degree six as shown in Figure 8. A closer examination of the relationship, and its underlying real-world decisions, however, reveals that *STUDENT* and *TEXTBOOK* should not be part of this relationship. This is because the fact that an instructor teaches a course during a particular semester is independent of who is enrolled in the course or what text is being used. Similarly, the choice of a specific textbook for a course is a separate decision. Hence, the enrollment and textbook decisions are to be represented as separate relationships, and should not be unnecessarily crowded and confused with another independent decision. Also note that the decision of a student to enroll in a course may depend on the instructor, the course, and the time of the course offering, as does the choice of a textbook. As a result, *STUDENT* and *TEXTBOOK* entities must participate in a relationship with the composite entity *SECTION*, and not only with its constituents. This is why Figure 5 represents the relationship of degree four as a composite entity.

In order to ascertain whether two decisions are similar, the designer may rely on two factors: (i) the decision maker and (ii) the time-frame of the decision. In the above example, there are three distinct decision makers. A student decides on his or her enrollment and an instructor decides on the textbook for the course that he or she is teaching. The assignment of an instructor to a course may be made by some other decision-maker — usually the chairperson of a department. Similarly, the time when a decision is made to offer a course is different than when the textbook for that course is selected (or when a student enrolls in it). This clearly indicates that there is a need for three separate relationships.

DESIGN IMPLICATION 1. *Real-world decisions are considered dissimilar if they involve different decision makers or are made during different time-*

frames. Two facts arising from two independent decisions should not be merged into a single relationship instance.

3.2 Relationship Attributes

Relationship attributes represent properties of the combination of all the participating entities. This fact can be used to identify whether irrelevant entities are connected to a relationship, or whether relevant entities have been excluded. If a relationship has an attribute that is a function of a set of entities different from the set of entities participating in the relationship, then there should be a new relationship among the former set of entities.

Consider the ternary relationship “supplies” in Figure 4. Assume that this relationship has an attribute “price.” If the price depends only on the supplier and the part, and not on the project, then we should have a binary relationship between SUPPLIER and PART. The original ternary relationship may then be broken into two binary relationships as shown in Figure 6. However, if the attribute “price” depends on the project as well, then the above decomposition is not appropriate. We state this more formally as follows:

DESIGN IMPLICATION 2. *Let R be a relationship among entities in the set \mathcal{E} . Let A be an attribute of R , with V as the value set of A , and let $\mathcal{E}' \neq \mathcal{E}$ be a set of entities. If A depends only on the entities in \mathcal{E}' , i.e., if A can be expressed as $A : \times_{E \in \mathcal{E}'} E'(R) \rightarrow V$, then a relationship R' should be created among $E \in \mathcal{E}'$, and A should become an attribute of R' . The appropriateness of R should be reexamined in the presence of R' .*

If A is a function of \mathcal{E}' , then the entities in \mathcal{E}' are clearly related with one another, and this association needs to be captured in a new relationship R' . This is analogous to the functional dependency-based arguments provided by Hainaut et al.[1993].

3.3 Look-Across Cardinalities for Higher-Degree Relationships

Consider a binary relationship, i.e., set $n = 2$ in Figure 2. The min-max cardinalities for E_1 are (c_1, d_1) ; those for E_2 are (c_2, d_2) ; they indicate the minimum and the maximum number of times an instance of an entity can participate in R . These numbers can also be interpreted as *look-across cardinalities* (LAC) [Thalheim 1992; McAllister and Sharpe 1998]:

- $R.LAmin(E_i) = c_i$ is the minimum number of instances of the other entity ($E_j, j \neq i$) that an instance of E_i can “see” when it “looks across” R .
- $R.LAmax(E_i) = d_i$ is the maximum number of instances of the other entity ($E_j, j \neq i$) that an instance of E_i can “see” when it “looks across” R .

The fundamental problem in applying this interpretation to higher-degree relationships is that an entity participating in such a relationship “sees”

more than one entity “looking across” the relationship. In this section, we describe how to interpret these numbers for higher-degree relationships and show how they can be used to identify the appropriate degree of a relationship. Our design rule is based on considering only two entities at a time. Consider the relationship R as shown in Figure 2. By $R.LAmin(E_i, E_j)$, $i \neq j$, we denote the minimum number of instances of E_j that an instance of E_i can “see” when it “looks across” R , and by $R.LAmax(E_i, E_j)$, the maximum. The pairwise LACs must be consistent in the following manner:

DESIGN IMPLICATION 3. *Let R and E_i , $i = 1, 2, \dots, n$, be as in Figure 2. Then, for R to be truly of degree n , $R.LAmin(E_i, E_j) = R.LAmin(E_i, E_k)$ and $R.LAmax(E_i, E_j) = R.LAmax(E_i, E_k)$ must hold for all distinct i, j , and k . If these conditions hold, then the min-max cardinalities for E_i in R , $i = 1, 2, \dots, n$, are given by (for any $j \neq i$):*

$$m_L(E_1, R) = R.LAmin(E_i) = R.LAmin(E_i, E_j), \text{ and}$$

$$m_U(E_1, R) = R.LAmax(E_i) = R.LAmax(E_i, E_j).$$

The above conditions require that all the minimum and the maximum pairwise LACs for an entity must match. A mismatch in the pairwise LAC indicates that the relationship incorrectly combines two or more isolated real-world decisions. Consider the ternary relationship “*supplies*” in Figure 4, and assume that the following is true:

$$\textit{supplies}.LAmin(\text{PART}, \text{SUPPLIER}) = 1 \text{ and}$$

$$\textit{supplies}.LAmin(\text{PART}, \text{PROJECT}) = 0$$

This means that, in the “*supplies*” relationship, an instance of PART must be associated with at least one instance of SUPPLIER, but need not have any association with any instance of PROJECT. Therefore, some instances of PART may not participate in the relationship at all, and the ternary relationship cannot represent the fact that every instance of PART is associated with at least one instance of SUPPLIER. The relationship between PART and SUPPLIER must then be independent of the relationship between PART and PROJECT. Similar arguments can be used for the maximum LAC as well.

It should be noted that a violation of the above two rules implies an error in the degree of a relationship. However, their compliance must not be taken as an indication that the higher-degree relationship is the appropriate representation. These rules serve as another level of verification for the consistency of the representation, and cannot be substituted for the semantics of the application. Furthermore, the use of LAC need not be limited to just verifying the consistency of a higher-degree relationship. LACs may also be used to determine how a higher-degree relationship should be

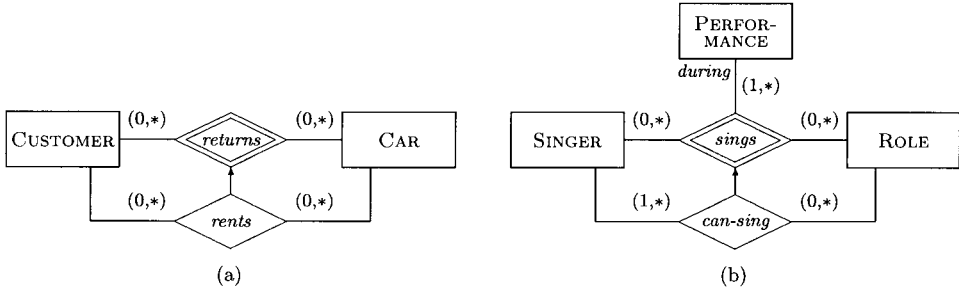


Fig. 9. Two examples of weak relationships.

broken into several lower-degree relationships if the conditions in the above design implication are not satisfied. We state this formally in the following design implication.

DESIGN IMPLICATION 4. *Let R and E_i be as above. Assume that the LACs for E_1 are inconsistent (based on Design Implication 3). Then, the set of entities $\{E_2, E_3, \dots, E_n\}$ can be partitioned into subsets \mathcal{E}_k , $k = 1, 2, \dots, n'$, for some $n' < n$, such that $R.LAmin(E_1, E_j) = R.LAmin(E_1, E_j)$ and $R.LAmax(E_1, E_j) = R.LAmax(E_1, E_j)$ must hold for all distinct $E_j, E_j \in \mathcal{E}_k$, $k = 1, 2, \dots, n'$. (The possibility that the above conditions are trivially satisfied for some k because \mathcal{E}_k is a singleton is not excluded.) Then, R could be broken into relationships Q_k involving E_1 and the entities in \mathcal{E}_k , $k = 1, 2, \dots, n'$. The min-max cardinalities of E_1 in Q_k are given by (for any $E_j \in \mathcal{E}_k$):*

$$m_L(E_1, Q_k) = R.LAmin(E_1, E_j), \text{ and}$$

$$m_U(E_1, Q_k) = R.LAmax(E_1, E_j)$$

4. WEAK RELATIONSHIPS

In this section we introduce a new construct, called the *weak* relationship, to model interrelationship dependence. A weak relationship is a relationship, the existence of whose instances depends on the instances of (one or more) other relationships. Consider the relationships: “CUSTOMER *rents* CAR” and “CUSTOMER *returns* CAR,” both of which may be important for a car rental agency. Of course, a car cannot be returned if it has not been rented. Thus, the second relationship is a weak relationship.

As shown in Figure 9, weak relationships are represented as double-lined diamonds. An arrow from a relationship to a weak relationship represents the inclusion dependency of the latter on the former. Similar interrelationship integrity constraints have also been identified by Laender and Flynn [1993] and Rochfeld and Negros [1992-93]. Weak relationships may arise in many real-world situations, such as when two relationships are connected by a time sequence, as shown in Figure 9(a), or when one relationship

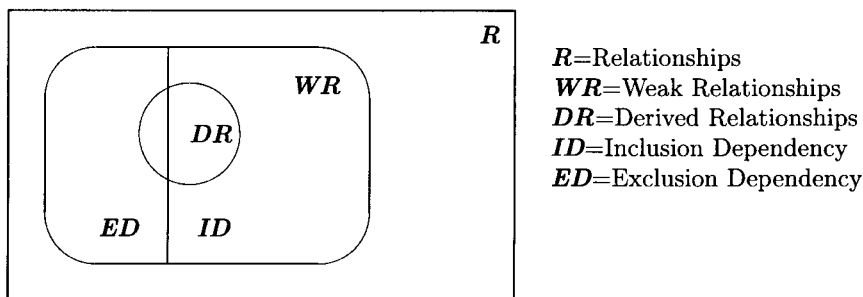


Fig. 10. A classification of weak relationships.

represents a possibility and another represents the actual realization, as shown in Figure 9(b).

In conceptual design, we often use relationships to represent the fact that one entity *can* “do” something with another entity. For example, the relationships “INSTRUCTOR *can-teach* COURSE,” “SUPPLIER *can-supply* PART,” or “SINGER *can-sing* ROLE” all show the mere possibility of an instance of one entity being associated with an instance of another; they do not depict the actual realization of the association. We call these *possibility* relationships. When the possibilities are actually realized, they must be represented as separate relationship instances. However, a realization actually depends on whether the possibility existed in the first place. For instance, a singer would be assigned to sing a role only if she or he can do it. Furthermore, there may be more than one realization of an instance of a possibility relationship. If a singer can sing a particular role, she may do so in more than one performance. As a result, the weak relationship may involve additional entities that are not part of the possibility relationship. For example, the weak relationship shown in Figure 9(b) involves an additional entity PERFORMANCE. We state these facts in the following:

DESIGN IMPLICATION 5. *Let \mathcal{E} be a set of entities. If P represents a possibility relationship involving entities in \mathcal{E} , then the materialized associations among these entities (and perhaps others) should be represented as a weak relationship, say Q , satisfying the inclusion dependency $Q[\mathcal{E}] \subset P$. Furthermore, $\delta(P) \leq \delta(Q)$, where $\delta(R)$ represents the degree of a relationship R .*

As shown in Figure 10, weak relationships must satisfy one of two types of interrelationship dependence: (i) inclusion dependency, or (ii) exclusion dependency. These dependencies are formally described in Sections 4.1 and 4.2. An important special case of these dependencies corresponds to *derived relationships*, **DR** in Figure 10, discussed in Section 4.3.

Although the examples in Figure 9 illustrate the dependence of only one relationship on another, it is straightforward to extend this notion to several relationships. In order to generalize the notion of weak relation-

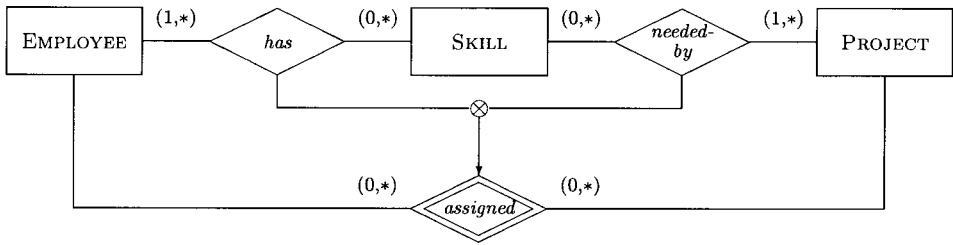


Fig. 11. A weak relationship dependent on a composite relationship.

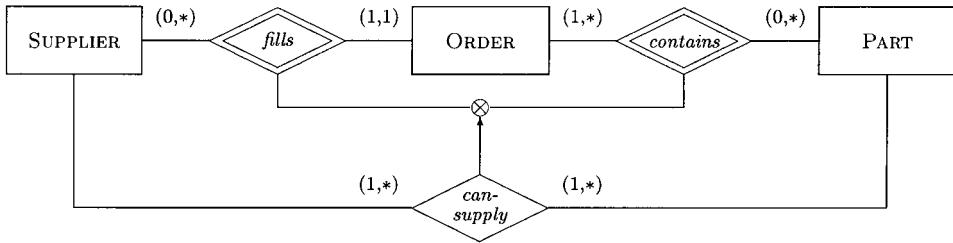


Fig. 12. A composite of two weak relationships.

ships, we first define a *composite* relationship. Let R_i be a relationship among entities in $\mathcal{E}_i, i = 1, 2, \dots, n$. A relationship Q among entities in $\cup_{i=1}^n \mathcal{E}_i$ is called a composite of R_1, R_2, \dots, R_n , written $Q = R_1 \otimes R_2 \otimes \dots \otimes R_n$, if and only if all $q \in Q$ satisfies $q(\mathcal{E}_i) \in R_i$, for all $i = 1, 2, \dots, n$. This is analogous to the *natural join* operation in the relational algebra.

4.1 Inclusion Dependency

Consider the relationships in Figure 11. This figure represents a situation where the completion of a project requires certain skills of the employees assigned to the project. Since different employees have different skills, the employee assignment for a project should be limited to only those employees who have the skills required by the project. In this case, the composite “ $has \otimes needed-by$ ” captures which employees have the required skills to contribute to a project. An inclusion dependency “ $assigned \subset has \otimes needed-by$ (EMPLOYEE,PROJECT)” ensures that an employee gets assigned to a project only if he/she has the required skills. In Figure 11, this inclusion dependency is shown as an arrow from the composite “ $has \otimes needed-by$ ” to “ $assigned$.”

It is also possible for the composite to be a weak relationship. Consider, for example, Figure 12, which represents the situation where a supplier fills an order containing different parts, and different suppliers can supply different parts. In this case, the composite “ $fills \otimes contains$ ” captures the list of parts a supplier would supply by filling a particular order. By having

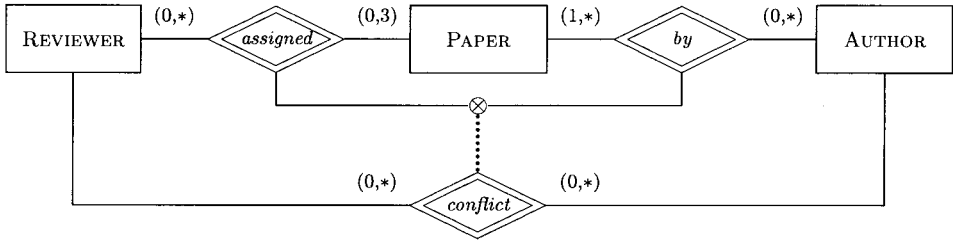


Fig. 13. Weak relationships with exclusion dependency.

an inclusion dependency “fills \otimes contains(SUPPLIER,PART) \subset can-supply,” it is possible to ensure that a supplier supplies only those parts that he/she can supply. More generally, we can say the following:

DESIGN IMPLICATION 6. *Let Q and R be two relationships among entities in \mathcal{D} and \mathcal{E} , respectively; the possibility that Q or R (or both) could be a composite relationship(s) is not excluded. Let $\mathcal{F} = \mathcal{D} \cap \mathcal{E}$. If Q is a weak relationship with an inclusion dependency on R , then Q must satisfy $Q(\mathcal{F}) \subset R(\mathcal{F})$, i.e., if $q \in Q$, then there must exist $r \in R$ such that $q(\mathcal{F}) = r(\mathcal{F})$.*

4.2 Exclusion Dependency

In order to understand the concept of exclusion dependency, consider the entities and relationships shown in Figure 13. This figure models a situation where research papers written by different authors are assigned to different reviewers. However, an assignment cannot be made if a reviewer has a “conflict of interest” with one or more authors of a paper. In this case, the composite “assigned \otimes by” represents the reviewer who is assigned an author’s paper, and the relationship “conflict” captures the reviewer who has conflicts of interest with an author. An exclusion dependency should be enforced to ensure that “conflict” and “assigned \otimes by(REVIEWER,AUTHOR)” are mutually exclusive. This is shown as a dotted line in Figure 13. It should be noted that, unlike inclusion dependency, there is no directionality in expressing an exclusion dependency. This is because in the case of an inclusion dependency, one relationship is a subset of another; the order of the relationships is important. In the case of an exclusion dependency, a relationship instance can not be part of two relationships at the same time; clearly, the order of these relationships does not matter. This is why all the relationships in Figure 13 are shown as weak relationships. We can now state this more formally as follows:

DESIGN IMPLICATION 7. *Let Q , R , and \mathcal{F} be as in Design Implication 6. Q and R are weak relationships with mutual exclusion dependency if they satisfy $Q(\mathcal{F}) \cap R(\mathcal{F}) = \emptyset$, i.e., if for all $q \in Q$ and $r \in R$, $q(\mathcal{F}) \neq r(\mathcal{F})$.*

4.3 Derived Relationships

A special type of a weak relationship is a *derived* or *redundant* relationship. A derived relationship is one whose instances can be inferred from instances of other relationships without ambiguity [Batra and Zanakis 1994; Rauh and Stickel 1993; Teorey et al. 1986]. Formally speaking:

DESIGN IMPLICATION 8. *Let R_1, R_2, \dots, R_n be relationships and let \mathcal{E} be the set of all entities that participate in at least one of these relationships. Define $R = R_1 \otimes R_2 \otimes \dots \otimes R_n$ and $\bar{R} = \{r(\mathcal{E}) \mid r(\mathcal{E}) \notin R\}$. Further, let Q be a relationship among entities in the set \mathcal{D} , where $\mathcal{D} \subset \mathcal{E}$. Q can be derived from R_1, R_2, \dots, R_n if either $Q(\mathcal{D}) = R(\mathcal{D})$ or $Q(\mathcal{D}) = \bar{R}(\mathcal{D})$.*

Since the existence of derived relationships increases redundancy in a database, it is important to identify them in the conceptual model. However, a derived relationship usually exists because it provides a natural view of the context to the user. It is, therefore, necessary that these user views are recreated from the stored data. Batra and Zanakis [1994] provide a set of three rules for detection, and subsequent elimination, of derived relationships. These rules rely on the participation constraints of the entities to assess if a relationship instance can be derived from others. We provide a generalized rule for the identification of derived relationships in the following:

DESIGN IMPLICATION 9. *Let $E_i, i = 1, 2, \dots, n$ be entities connected by a relationship R . Let $D_i, i = 1, 2, \dots, k, k \leq n$, be entities such that there is a functional relationship between E_i and D_i with $Q_i : E_i \rightarrow D_i$. The possibility that $D_i = E_i$, for some i , is not excluded. Then there is a derived relationship R' among entities $D_i, i = 1, 2, \dots, k, k \leq n$. Furthermore, the min-max cardinalities of this derived relationship are as*

$$m_L(D_i, R') = m_L(D_i, Q_i) \times m_L(E_i, R), \forall i = 1, 2, \dots, k, \text{ and}$$

$$m_U(D_i, R') = m_U(D_i, Q_i) \times m_U(E_i, R), \forall i = 1, 2, \dots, k.$$

To understand why, note that, for every entity instance $e_i \in E_i, i = 1, 2, \dots, k$, there is an instance $d_i \in D_i$. Thus, for every relationship instance $\langle e_1, e_2, \dots, e_n \rangle \in R$, we can derive a relationship instance $\langle d_1, d_2, \dots, d_k \rangle \in R'$. The min-max cardinalities follow in a straightforward manner. Consider the minimum cardinality for D_i in R' . For every $d_i \in D_i$, there is at least $m_L(D_i, Q_i)$ instances of E_i , and each of those instances participate in R at least $m_L(E_i, R)$ number of times. Since for every instance of R , there is a corresponding instance of R' , it is easy to see that there is at least $m_L(D_i, Q_i) \times m_L(E_i, R)$ number of instances of d_i in R' . Similar arguments can be applied for the maximum cardinality as well. Since R itself could be a derived relationship, it is clear that this rule can be applied repeatedly for a loop involving any number of relationships.

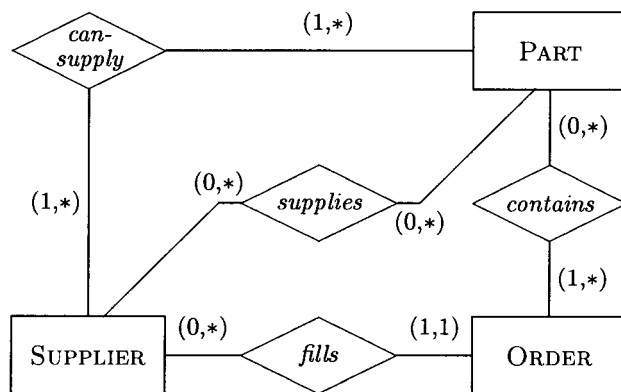


Fig. 14. Redundant relationship.

It must be noted that redundancy in relationships is fundamentally a semantic issue [Rauh and Stickel 1993]. From the structural constraints, we can only infer that a derived relationship R' exists. We cannot identify whether R' is indeed the same as another relationship already represented in the conceptual model. In other words, structural constraints can be used as one of the *necessary* conditions in detecting such relationships, but they are not *sufficient*. It should be verified that the relationship in question has the same meaning as the relationship that can be derived from others.

Consider the relationships in Figure 14. In this figure, $\text{PART}=\text{PART}$, and $\text{fills: ORDER} \rightarrow \text{SUPPLIER}$. This indicates the existence of a derived relationship between PART and SUPPLIER. Furthermore, we can infer that the min-max cardinalities for both PART and SUPPLIER in this derived relationship should be $(0, *)$. Figure 14 contains two different relationships between PART and SUPPLIER. The min-max cardinalities of the relationship “*can-supply*” do not agree with those for the derived relationship; thus, “*can-supply*” cannot be redundant. However, the relationship “*supplies*” does match with the derived one in terms of the cardinalities. The question then becomes whether this relationship has the same meaning as the derived one. If the semantics of the application dictates that a supplier supplies parts whenever he/she fills an order (and *vice versa*), then the relationship “SUPPLIER *supplies* PART” is redundant. On the other hand, if “*supplies*” represents the orders that have arrived and been processed in the last week, whereas “*fills*” pertains only to the current orders, then these two relationships are disjoint. In that case, “SUPPLIER *supplies* PART” is not redundant. The meaning of the relationship thus plays a very important role in deciding whether or not a relationship can be derived from others.

5. BINARY REPRESENTATION OF HIGHER-DEGREE RELATIONSHIPS

A higher-degree relationship may be expressed as several binary relationships, although such a conversion may not lead to a “natural” view of the

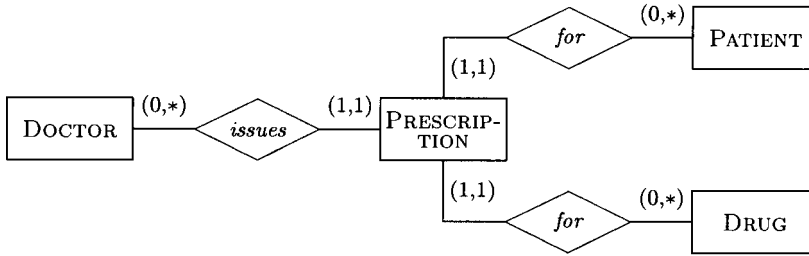


Fig. 15. Alternative binary representation of a functional relationship.

application. In this section we discuss appropriate binary representation for all the three types of higher-degree relationship: (i) functional relationships; (ii) partial functional relationships; and (iii) general relationships. We show how the min-max cardinalities should be assigned to the participating entities so that the new representation is consistent with the original representation.

5.1 Functional Relationships

Assume, without loss of generality, that the min-max cardinalities of E_1 in R (as shown in Figure 2) are (1,1); i.e., $c_1 = d_1 = 1$. Clearly, R can be expressed as a vector-valued function.

$$R = f : E_1 \rightarrow E_2 \times E_3 \times \dots \times E_n,$$

and the individual components of $f(\cdot)$ are $f_i : E_1 \rightarrow E_i$, for all $i = 2, 3, \dots, n$. In other words, $E'_i(R) = f_i(E_1)$, for all $i = 2, 3, \dots, n$. Since a vector-valued function is completely specified by its components, we can represent the relationship in Figure 1 as three separate binary relationships, as shown in Figure 15. This leads to the following:

DESIGN IMPLICATION 10. *Let R and E_i , $i = 2, 3, \dots, n$, be as shown in Figure 2. Further let $c_1 = d_1 = 1$. Then R can also be expressed as $(n - 1)$ binary relationships Q_i between E_1 and E_i , $i = 2, 3, \dots, n$. The min-max cardinalities of the new relationships can be written as*

$$m_L(E_i, Q_i) = m_L(E_i, R), \forall i = 2, 3, \dots, n,$$

$$m_U(E_i, Q_i) = m_U(E_i, R), \forall i = 2, 3, \dots, n,$$

$$m_L(E_i, Q_i) = m_U(E_i, Q_i) = 1, \forall i = 2, 3, \dots, n.$$

Although the above decomposition rule is straightforward, there may be two complications arising from such a decomposition. First, a higher-degree functional relationship may have its own attributes. When the relationship is decomposed into several binary relationships, it is not clear which entity the relationship attributes should be attached to. Second, the functional

relationship may also be a composite entity in the overall conceptual model. When the relationship is decomposed, it is not clear what transformation should be made to the relationships connected to the composite entity. We discuss each of these two situations below.

When a higher-degree functional relationship is decomposed into several binary relationships, the relationship attributes can be transferred to the entity with (1,1) cardinalities. For example, the attribute “date” of the relationship “issues” in Figure 1 can also be considered an attribute of PRESCRIPTION. To prove this, assume that A is an attribute of R , and $c_1 = d_1 = 1$ in Figure 2. Since A is a function of all the participating entities, we can write

$$\begin{aligned} A &= g(E'_1, E'_2, E'_3, \dots, E'_n) \\ &= g(E_1, f_2(E_1), f_3(E_1), \dots, f_n(E_1)) = h(E_1), \end{aligned}$$

where $g(\cdot)$ and $h(\cdot)$ are two different functions. This implies that A is an attribute of E_1 . Thus, we have the following:²

DESIGN IMPLICATION 11. *The attributes of a functional relationship can all be transferred to a participating entity with (1,1) cardinalities. Such a transfer of attributes is not necessary (but possible) if the relationship is not decomposed.*

In order to address the second concern (about the relationships of the composite entity), we note that a functional relationship need not be represented as a composite entity. A simpler representation is possible where the relationships associated with the composite entity are transferred to the entity participating in the underlying functional relationship with (1,1) cardinalities. For example, assume that the relationship in Figure 1 is represented as a composite entity ISSUANCE, and suppose it participates in a relationship: “PHARMACY fills ISSUANCE.” This relationship can always be transferred to PRESCRIPTION. In other words, we can replace this relationship by “PHARMACY fills PRESCRIPTION.” We state this formally as follows:

DESIGN IMPLICATION 12. *Let R be the underlying relationship of a composite entity C . Let R be functional and let E be a participating entity with (1,1) cardinalities. If C participates in a relationship Q , then Q could be transferred to E so that $m_L(E, Q) = m_L(C, Q)$ and $m_U(E, Q) = m_U(C, Q)$. Such a transfer of relationships is not necessary (but possible) if R is not decomposed.*

²This generalizes the rule for binary relationship attributes proposed by Storey and Goldstein [1988].

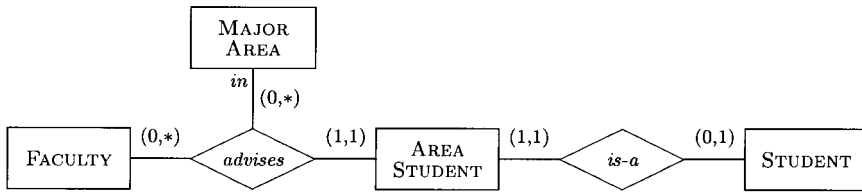


Fig. 16. A partial functional relationship represented as a functional relationship using a subtype entity AREA STUDENT.

5.2 Partial Functional Relationships

The decomposition scheme stated in Design Implication 10 may also work with partial functional relationships. The only adjustment necessary is in the minimum cardinalities for E_1 ; we should set $m_L(E_1, Q_i) = 0$ for all $i = 2, 3, \dots, n$. However, there are two problems with such a decomposition of a partial functional relationship. First, if the relationship has attributes, then these attributes cannot be transferred to a participating entity. Second, if the relationship is also expressed as a composite entity, then the relationships of the composite entity cannot be transferred to another entity without losing useful information on participation constraints. In other words, Design Implications 11 and 12 cannot be applied to partial functional relationships. Therefore, partial functional relationships cannot always be directly decomposed to binary relationships. However, two options are available to the designer. The first is to introduce a subtype of the entity with (0,1) cardinalities such that the participation of all instances of the subtype entity is mandatory. Then, the relationship becomes a functional relationship, which can be easily decomposed. This is illustrated in Figure 16 (an alternative representation of the relationship in Figure 3). In Figure 3, STUDENT has (0,1) cardinalities, whereas, in Figure 16, we introduce a subtype entity AREA STUDENT to represent those students who have already decided on their major area and faculty advisor. When the original partial functional relationship is transferred to the subtype, it becomes a functional relationship and can now be decomposed as stated in Design Implication 10. Stated formally:

DESIGN IMPLICATION 13. *Let R be a partial functional relationship involving entities in \mathcal{E} , and let $E \in \mathcal{E}$ be the entity with (0,1) cardinalities. Then, R can be converted to a functional relationship by replacing E with the entity $F = E'(R) \subset E$. Min-max cardinalities of F in R are (1,1).*

In some situations, the subtype entity may not be a natural entity in the application context, and the designer may not be willing to introduce the subtype. In that case, a second alternative binary representation is possible, where the relationship is expressed as an entity; this is discussed in the next section, where we show the representation of a general relationship as an entity.

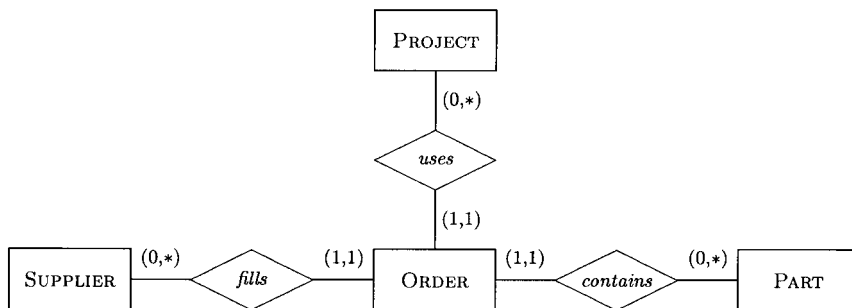


Fig. 17. A general relationship represented as an entity.

5.3 General Relationships

In situations involving general relationships, an equivalent binary representation could be obtained by converting the relationship to an entity. As mentioned in the previous section, this approach can also be used for converting partial functional relationships, especially if the relationship has its own attributes or if it is represented as a composite entity participating in other relationships. Note that the entity representation approach discussed here is different from the decomposition approach in Section 5.1 in two ways. First, in the entity representation approach, a new entity must be created; the creation of such a new entity is not required in the decomposition approach. Second, when a relationship is converted to an entity, each of the participating entities should be connected to the new entity by a binary relationship. Therefore, if the original relationship has a degree of n , then n new binary relationships would be created, as opposed to $(n - 1)$ binary relationships in the decomposition approach. An example of the entity representation approach is shown in Figure 17 where the “supplies” relationship from Figure 4 is represented as an entity ORDER. Each of the participating entities in Figure 4 is connected to ORDER using a binary relationship. All attributes of “supplies” can be transferred to ORDER. The formal conversion rules are stated below:

DESIGN IMPLICATION 14. *Let R be a relationship among entities E_1, E_2, \dots, E_n (see Figure 2). If R is converted to an entity ER , then E_i must be connected to ER using a binary relationship Q_i , $i = 1, 2, \dots, n$. The min-max cardinalities of the new relationships are*

$$m_L(E_i, Q_i) = m_L(E_i, R), \forall i = 1, 2, \dots, n,$$

$$m_U(E_i, Q_i) = m_U(E_i, R), \forall i = 1, 2, \dots, n,$$

$$m_L(ER, Q_i) = m_U(ER, Q_i) = 1, \forall i = 1, 2, \dots, n.$$

When a relationship is converted to an entity, the transfer of attributes and relationships is straightforward, as shown below:

Table II. A Historical Relationship Between SUPPLIER AND PART

SUPPLIER	PART	Date	Quantity	Order_no
s1	p1	01/12/93	500	o1
s1	p2	01/14/93	400	o2
s1	p1	02/17/93	500	o3
s2	p1	03/21/93	200	o5
...

DESIGN IMPLICATION 15. *Let R be a relationship which is converted to an entity ER . Then, all the attributes of R can be transferred to ER . Further, if C is a composite entity for R and if C participates in a relationship Q , then Q should be moved from C to ER , so that $m_L(ER, Q) = m_L(C, Q)$ and $m_U(ER, Q) = m_U(C, Q)$.*

Since a relationship can be converted to an entity, given a real-world concept, the designer needs to decide whether to model it as a relationship or as an entity. The designer's objective is to develop a conceptual model that is close to the users' perception. In real-world situations, the users' perception is primarily driven by the existence (or lack thereof) of a unique identifier for the concept.

DESIGN IMPLICATION 16. *If there is a unique identifier for a real-world concept, the user is likely to view the concept as an entity.*

To understand why, recall that only entities have primary keys in the ER model; the concept of a primary key as the unique identifier of a relationship does not exist. However, in many real-world applications, there may be an attribute of the relationship that appears to be a natural unique identifier for the relationship instances. For example, the attribute "Order no" may be used as an identifier of "SUPPLIER *supplies* PART"; we call such an attribute an *enforced key* of the relationship. Fundamentally, a key is a surrogate for a real-world object [Parsons and Wand 1997]. Thus, the existence of a key indicates the existence of an entity. Naturally, the relationship is perceived as an entity by the user, e.g., "SUPPLIER *fills* ORDER *for* PART." The attribute "Order no" can be used as an identifier of the entity "ORDER."

There is another practical reason for representing relationships as entities. Note that, in the above situations, the participating entity instances may not uniquely identify each relationship instance. For example, a specific supplier may supply a part several times. Therefore, the supplier and part identifiers together are not sufficient to identify the relationship instances uniquely as shown in Table II. The enforced key (Order_no), however, uniquely identifies every relationship instance. Since the ER model cannot capture the relationship history, it is necessary to represent the relationship as an entity ORDER, with a primary key that monotonically increases with time and hence can be used as a surrogate for time. Dey et

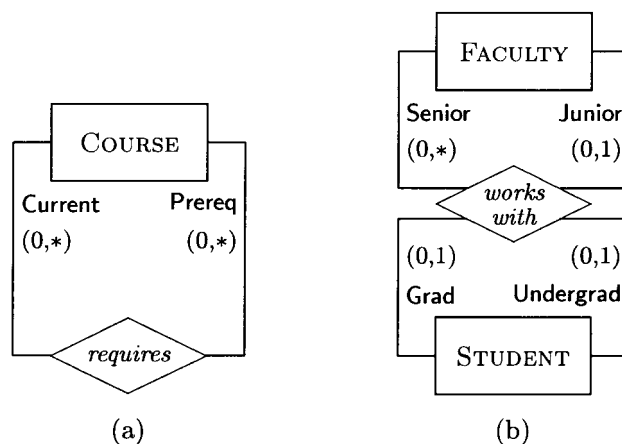


Fig. 18. Two recursive relationships.

al. [1995] discuss an alternative representation of these relationships within the context of temporal databases.

6. RECURSIVE RELATIONSHIPS

The assumption thus far has been that each of the E_i 's in Figure 2 participates in the relationship R only once. However, it is possible for an entity to participate in a relationship multiple times [Elmasri and Navathe 1994, p. 50; Teorey et al. 1986]; two such examples are shown in Figure 18. The first relationship represents a situation where the enrollment in certain courses requires the student to have passed the associated prerequisite courses. The second relationship represents a committee where a senior faculty, a junior faculty, a graduate student, and an undergraduate student work together. These relationships, where an entity participates more than once, are called *recursive* relationships [Batini et al. 1992, p. 31; Hawryszkiewicz 1990, pp. 149-151]. When specifying a recursive relationship, it is necessary to specify the different *roles* that a participating entity plays in the relationship.³ For example, in Figure 18(a), the roles associated with COURSE are “Current” and “Prerequisite.”

Note that binary recursive relationships are often referred to as “unary” relationships [Batra and Zanakis 1994; Teorey et al. 1986]. The name “unary,” however, is a misnomer, and its usage in conceptual modeling is rather unfortunate. Fundamentally, a relationship represents an association among entities. The purpose of a unary relationship is to capture the

³The *role* played by an entity instance in a relationship instance is a general concept, and can be used with recursive as well as nonrecursive relationships. However, when an entity participates more than once in a relationship, it is necessary to specify the different roles it can play in order to distinguish the meaning of each participation [Elmasri and Navathe 1994, p. 52].

association of several entity instances of the same type, not to represent the trivial association of an entity instance with itself.

To generalize our notation, we assume that E_i 's participate (possibly several times) in a relationship R , $i = 1, 2, \dots, n$. The *multiplicity* of E_i in R , denoted $\mu(E_i, R)$, is the number of roles played by E_i in R . Suppose these roles are denoted by l_{ij} , $j = 1, 2, \dots, \mu(E_i, R)$. Then each participation of an entity can be expressed as a leg of a relationship given by the pair $L_{ij} = \langle E_i, l_{ij} \rangle$. We now redefine the *degree* of a relationship as the number of legs connected to it.

DESIGN IMPLICATION 17. *If R is a relationship among entities E_i , $i = 1, 2, \dots, n$, then the degree of R , denoted $\delta(R)$, can be written as $\delta(R) = \sum_{i=1}^n \mu(E_i, R)$.*

The relationship in Figure 18(a) can then be classified as a recursive relationship of degree two, and the one in Figure 18(b) as a recursive relationship of degree four. In the second relationship, both FACULTY and STUDENT have a multiplicity of two.

The min-max cardinalities for a recursive relationship are expressed for each leg, and not for each entity. Assume that entity E_i participates in relationship R in role l_{ij} ; then $L_{ij} = \langle E_i, l_{ij} \rangle$ is a leg of R . For an instance $e \in E_i$, define $R_{(L_{ij}, e)} \subset R$ as the set of all ordered tuples containing e in the appropriate role in leg L_{ij} . Then

$$m_L(L_{ij}, R) = \min_{e \in E} |R_{(L_{ij}, e)}| \text{ and } m_U(L_{ij}, R) = \max_{e \in E} |R_{(L_{ij}, e)}|.$$

Recursive relationships can be one of two types: symmetric and asymmetric. A symmetric recursive relationship is one where the roles (two or more) played by an entity in the relationship are indistinguishable. All other recursive relationships are asymmetric. Symmetric recursive relationships have several implications in database design; we discuss these in Section 6.1. In some situations, it may be more natural for the designer to view a recursive relationship as an entity; Section 6.2 provides some examples and discusses the associated design implications.

6.1 Symmetric Recursive Relationships

If the roles played by an entity in a recursive relationship are indistinguishable, the relationship is said to be *symmetric* with respect to that entity. Consider the relationship shown in Figure 19. This relationship is symmetric with respect to TEAM if we assume that a team participating in a game is indistinguishable from its opposing team.⁴ In Figure 19, we identify the symmetric relationship by joining the symmetric legs with a

⁴In contrast, if we can distinguish the teams as the “home” and the “visitor” teams, for example, the relationship would not be considered symmetric.

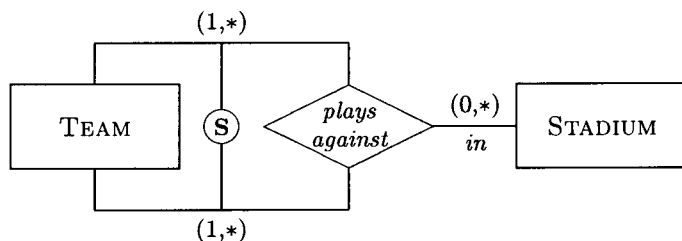


Fig. 19. A symmetric recursive relationship.

line labeled “S.” Symmetric recursive relationships have several important implications for database design:

DESIGN IMPLICATION 18. *If a recursive relationship is symmetric about an entity, the min-max cardinalities associated with the different roles of that entity are the same; in general, the converse is not true.*

DESIGN IMPLICATION 19. *If a recursive relationship is symmetric about an entity, the roles of that entity must be separated in an artificial manner, possibly by using numeric suffixes.*

For example, the roles of a TEAM can be expressed as Team1 and Team2. Now consider two instances t_1 and t_2 , of the entity TEAM who have played a game against each other. This association can be represented either as $\langle t_1, t_2 \rangle$ or as $\langle t_2, t_1 \rangle$, but to avoid redundancy, only one should be stored. We can state this more formally as follows:

DESIGN IMPLICATION 20. *Let E be an entity participating in a relationship R with $\mu(E, R) = k$; let R be symmetric about E . Then the order of the association implied by R among distinct instances $e_i \in E$, $i = 1, 2, \dots, k$, is not important, and can be expressed as $a = \{e \mid e \in E\}$ such that $|a| = k$.*

6.2 Recursive Relationship as an Entity

It may sometimes be more “natural” to represent a recursive relationship R as an entity ER , and connect the participating entities E_i to ER with the help of binary relationships. This can be formalized as follows:

DESIGN IMPLICATION 21. *Let a relationship R be expressed as an entity ER . For each leg L_{ij} of R , $i = 1, 2, \dots, n$, $j = 1, 2, \dots, \mu(E_i, R)$, a binary relationship Q_{ij} must be created between E_i and ER . The min-max cardinalities can be expressed, as before, as follows:*

$$m_L(E_i, Q_{ij}) = m_L(L_{ij}, R), \forall i = 1, 2, \dots, n, \forall j = 1, 2, \dots, \mu(E_i, R)$$

$$m_U(E_i, Q_{ij}) = m_U(L_{ij}, R), \forall i = 1, 2, \dots, n, \forall j = 1, 2, \dots, \mu(E_i, R), \text{ and}$$

$$m_L(E_i, Q_{ij}) = m_U(L_{ij}, R), \forall i = 1, 2, \dots, n, \forall j = 1, 2, \dots, \mu(E_i, R).$$

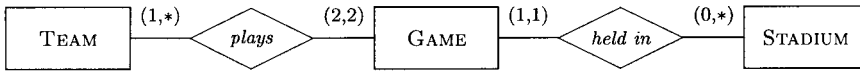


Fig. 20. A Symmetric recursive relationship represented as an entity.

There is one exception to the above rules. When a recursive relationship is symmetric with respect to an entity, the relationships resulting from the (indistinguishable) roles of that entity is exactly the same. In this case, these relationships must be combined into a single one. However, when combining these relationships, one should be careful to ensure that the min-max cardinalities are properly adjusted. Consider the relationship in Figure 19. The recursive relationship shown here may be represented by an entity *GAME*. Since the role of a team in a game cannot be distinguished from the role of the other team, we need only one relationship, as shown in Figure 20. Note that the min-max cardinalities associated with *GAME* are (2,2): there are exactly two teams participating in a game. This leads to the following:

DESIGN IMPLICATION 22. *Let R be a recursive relationship symmetric about an entity E . If R is expressed as an entity ER , then there should be only one binary relationship Q between E and ER . Further, let L be any leg of R involving E . Then min-max cardinalities r are given by*

$$m_L(E, Q) = m_L(L, R),$$

$$m_U(E, Q) = m_U(L, R),$$

$$m_L(ER, Q) = m_U(ER, Q) = \mu(E, R).$$

7. CONVERSION TO THE RELATIONAL MODEL

This section presents a scheme for transforming the conceptual model (possibly containing higher-degree relationships) into a corresponding relational structure that is natural and intuitively appealing. The associated integrity constraints are also stated. We have verified that these representations have *information capacity* [Hull 1986] equivalent to representations produced by the algorithms *Crep*, *Norm*, and *Smerge* developed by Markowitz and Shoshani [1992]. As they have shown, (i) *Crep* produces relational schemes that are *correct*, in that consistent states in the ER model are mapped into consistent states in the relational model and vice versa; and (ii) *Norm*, followed by *Smerge*, produce schemes without redundant attributes in Boyce-Codd normal form and preserving the information capacity of the schemes created by *Crep*. Thus, all of the representations shown below possess these formal properties, in addition to being natural and intuitively appealing relational schemes.

An entity is represented in the relational model by creating a relation with the primary key and attributes of the entity [Storey and Goldstein 1988; Teorey 1986]. Let E be an entity; then $\tau(E)$ denotes the relation representing E . It is possible that E is a composite entity. If E has its own primary key, then that is also the primary key of $\tau(E)$; the primary keys of the constituent entities should be added to $\tau(E)$ as foreign keys. If not, then the primary key of $\tau(E)$ is the combination of the primary keys of the constituent entities; each component of the primary key is also a foreign key referring to the relation from the appropriate entity.

Relationships can be represented either by using foreign keys or constructing new relations. The exact representation depends on the type of the relationship and its associated min-max cardinalities. In a general sense, a functional relationship can be represented using foreign keys; a new relation is not necessary. If the relationship is not a function, then a new relation should be created. Note that, whenever a relationship can be represented using foreign keys, an alternative representation of creating a new relation is always possible [Wilmot 1984]. In general, the converse is not true.

For the remainder of the discussion, let R be a relationship connecting (distinct) entities E_1, E_2, \dots, E_n . The possibility that E_i participates more than once in R is not excluded. In other words, R could be a recursive relationship; the leg L_{ij} of R represents the connection to entity E_i in role l_{ij} , $j = 1, 2, \dots, \mu(E_i, R)$. Let k_i be the primary key of E_i , $i = 1, 2, \dots, n$.

7.1 Functional Relationships

Assume that R is a functional relationship; i.e., there is a leg L_{ij} with min-max cardinalities (1,1). Let $i = j = 1$, without loss of generality. If there is more than one leg with cardinalities (1,1), the choice is up to the designer, and will often depend on the type of queries being asked in the application [Storey and Goldstein 1988]. The following steps are required to represent this case:

- (1) Create relations $\tau(E_i)$, $i = 1, 2, \dots, n$, as discussed above.
- (2) Make $l_{ij}k_i$ a foreign key of $\tau(E_1)$, for all $i = 1, 2, \dots, n$, $j = 1, 2, \dots, \mu(E_i, R)$, and either $i \neq 1$ or $j \neq 1$.
- (3) Append all attributes of R (if present) to $\tau(E_1)$ as additional attributes.

Note that the role-based prefixes l_{ij} are necessary in Step 2 only when E_i participates in R more than once, so that each role can be distinguished from the others. If $\mu(E_i, R) = 1$, the prefix for k_i may be dropped. The following integrity constraints should be enforced:

Existence integrity constraint: $\tau(E_i)$ cannot have null or duplicate values for k_i , for all $i = 1, 2, \dots, n$.

Table III. Relational Representation of Figure 1

Relational Structure	
$\tau(\text{DOCTOR})$	[DEA#, <DOCTOR attributes>]
$\tau(\text{PATIENT})$	[PAT ID, <PATIENT attributes>]
$\tau(\text{DRUG})$	[NDC#, <DRUG attributes >]
$\tau(\text{PRESCRIPTION})$	[RX#, <PRESCRIPTIONattributes >, dea#, pat_id, ndc#]
Referential Integrity Constraints	
$\tau(\text{PRESCRIPTION}).\text{dea\#}$	$\rightarrow \tau(\text{DOCTOR}).\text{DEA\#}$
$\tau(\text{PRESCRIPTION}).\text{patient id}$	$\rightarrow \tau(\text{PATIENT}).\text{PATIENT ID}$
$\tau(\text{PRESCRIPTION}).\text{ndc\#}$	$\rightarrow \tau(\text{DRUG}).\text{NDC\#}$

Referential integrity constraint: For each leg L_{ij} (except L_{11}), there should be a referential constraint from $\tau(E_1)$ to $\tau(E_i)$, with $l_{ij_k_i}$ being the foreign key of $\tau(E_1)$. Symbolically, $\tau(E_1).l_{ij_k_i} \rightarrow (E_i).k_i$, for all i and j and either $i \neq 1$ or $j \neq 1$.

Consider the relationship in Figure 1. Assume that the primary keys of DOCTOR, PATIENT, DRUG, and PRESCRIPTION are “DEA#,” “PAT ID,” “NDC#,” and “RX#,” respectively. Since the min-max cardinalities for PRESCRIPTION are (1,1), we need to include the primary keys of all other entities as foreign keys in the scheme that represents PRESCRIPTION. The relational structure and the referential integrity constraints are shown in Table III. Note that it does not matter if we use the alternative ER representation in Figure 15; the resulting structure would be exactly the same for both the representations; this further illustrates the structural equivalence of the two representations.

Conversion of ternary and other higher-degree relationships into the relational model has been a consistent source of confusion. It is generally assumed that all higher-degree relationships are automatically converted to new relations [Batini et al. 1992; Elmasri and Navathe 1994; Teorey et al. 1986]. If this rule is used on the two ER representations in Figures 1 and 15, two different relational representations would result. Therefore, this is not the only, or necessarily the most appropriate, representation. If a higher-degree relationship is a functional relationship, it can also be represented using foreign keys. Situations where a new relation scheme is necessary are discussed below.

7.2 Partial Functional Relationship

If R is a partial functional relationship, then there is a leg L_{ij} with cardinalities (0,1). As before, we can let $i = j = 1$, without any loss of generality. In this case the following steps are required:

- (1) Create relations $\tau(E_i)$, $i = 1, 2, \dots, n$.
- (2) Create a relation $\tau(R)$ with $l_{11_k_1}$ as the primary key; append all attributes of R as attributes of $\tau(R)$.

Table IV. Relational Representation of Figure 3

Relational Structure	
$\tau(\text{FACULTY})$	[<u>FAC_ID</u> , <FACULTY attributes>]
$\tau(\text{MAJOR})$	[<u>AREA_ID</u> , <MAJOR AREA attributes>]
$\tau(\text{STUDENT})$	[<u>ST_NO</u> , <STUDENT attributes >]
$\tau(\text{advises})$	[<u>ST_NO</u> , <advises attributes>, fac_id, area_id]
Referential Integrity Constraints	
$\tau(\text{advises}).\text{ST_NO}$	$\rightarrow \tau(\text{STUDENT}).\text{ST_NO}$
$\tau(\text{advises}).\text{fac_id}$	$\rightarrow \tau(\text{FACULTY}).\text{FAC_ID}$
$\tau(\text{advises}).\text{area_id}$	$\rightarrow \tau(\text{MAJOR}).\text{AREA_ID}$

- (3) Make $l_{ij_k_i}$ a foreign key of $\tau(R)$, for all $i = 1, 2, \dots, n, j = 1, 2, \dots, \mu(E_i, R)$, and either $i \neq 1$ or $j \neq 1$.

Again, the role-based prefixes may be omitted if they are not necessary. The following integrity constraints should be enforced:

Existence integrity constraint: $\tau(E_i)$ cannot have null or duplicate values for k_i , for all $i = 1, 2, \dots, n$. $\tau(R)$ cannot have null and duplicate values for $l_{11_k_1}$.

Referential integrity constraint: For each leg L_{ij} , there should be a referential constraint from $\tau(R)$ to $\tau(E_i)$, with $l_{ij_k_i}$ being the foreign key of $\tau(R)$. Symbolically, $\tau(R).l_{ij_k_i} \rightarrow \tau(E_i).k_i$, for all $i = 1, 2, \dots, n, j = 1, 2, \dots, \mu(E_i, R)$.

Consider the relationship in Figure 3. Assume that the primary keys of FACULTY, STUDENT, and MAJOR AREA are “FAC ID,” “ST NO,” and “AREA ID,” respectively. Since the min-max cardinalities for STUDENT are (0,1), we need to create a new relation with the primary key of Student. The primary keys of FACULTY and MAJOR AREA should be added to this new relation as foreign keys. The resulting structure and the referential integrity constraints are shown in Table IV.

An important clarification is necessary. It is possible to represent a partial functional relationship using foreign keys only, as suggested by Elmasri and Navathe [1994] and Teorey et al. [1986], as well as several others. However, this is not a good practice in database schema design. Recall that a partial functional relationship implies that there must be at least one entity with (0,1) cardinalities. Not all instances of this entity participate in the relationship, so some of the tuples in the relation have null values for the foreign key (and associated relationship attributes). Although, null values cannot be completely avoided in some situations, we discourage database design practices that primarily rely on the existence, and promote the use, of these null values.

Table V. Relational Representation of Figure 4

Relational Structure	
$\tau(\text{SUPPLIER})$	[SUP#, <SUPPLIER attributes>]
$\tau(\text{PROJECT})$	[PROJ#, <PROJECT attributes>]
$\tau(\text{PART})$	[PART#, <PART attributes>]
$\tau(\text{supplies})$	[SUP#, PROJ#, PART#, quantity, unit price, date]
Referential Integrity Constraints	
$\tau(\text{supplies}).\text{SUP\#}$	$\rightarrow \tau(\text{SUPPLIER}).\text{SUP\#}$
$\tau(\text{supplies}).\text{PROJ\#}$	$\rightarrow \tau(\text{PROJECT}).\text{PROJ\#}$
$\tau(\text{supplies}).\text{PART\#}$	$\rightarrow \tau(\text{PART}).\text{PART\#}$

7.3 General Relationships

Assume that R is neither a functional nor a partial functional relationship; i.e., there is no entity with min-max cardinalities of either (1,1) or (0,1). The following steps are required to represent this case:

- (1) Create relations $\tau(E_i)$, $i = 1, 2, \dots, n$.
- (2) Create a relation $\tau(R)$ with the combination of all $l_{ij}k_i$, $i = 1, 2, \dots, n$, $j = 1, 2, \dots, \mu(E_i, R)$ as the primary key; append all attributes of R as attributes of $\tau(R)$.

The role-based prefixes may be dropped if they are not necessary. The following integrity constraints need to be enforced:

Existence integrity constraint: $\tau(E_i)$, $i = 1, 2, \dots, n$, cannot have null or duplicate values for k_i . $\tau(R)$ cannot have null and duplicate values for any of $l_{ij}k_i$, $i = 1, 2, \dots, n$, $j = 1, 2, \dots, \mu(E_i, R)$.

Referential integrity constraint: For each leg L_{ij} , there should be a referential constraint from $\tau(R)$ to $\tau(E_i)$, with $l_{ij}k_i$ being the foreign key of $\tau(R)$. Symbolically, $\tau(R).l_{ij}k_i \rightarrow \tau(E_i).k_i$, for all $i = 1, 2, \dots, n$, $j = 1, 2, \dots, \mu(E_i, R)$.

As an example, consider the relationship in Figure 4. Assume the primary keys of SUPPLIER, PROJECT, and PART are “SUP#,” “PROJ#,” and “PART#,” respectively. Also assume that the relationship has as its attributes “quantity,” “unit price,” and “date.” A new relation must be created; the resulting structure and the referential integrity constraints are shown in Table V.

8. CONCLUSION

Conceptual modeling is the first and most crucial phase in database design. In this phase, users’ information requirements are represented in the form of a conceptual model. Current approaches to conceptual modeling focus mainly on modeling the entities; not enough importance is given to relationships, especially ternary and other higher-degree relationships. In this

paper we present a generalized framework for analyzing relationships during conceptual modeling of real-world applications. Four aspects of relationship modeling are examined in this framework: (i) min-max cardinalities, (ii) degree, (iii) the recursive nature of relationships, and (iv) interrelationship constraints. Generalized notation for representing relationships is presented and guidelines provided that identify what modeling constructs to use in different situations. Explicit rules are provided for assessing the degree of a relationship and understanding recursive relationships. Special semantics associated with *weak* relationships and interrelationship dependence are also analyzed. Finally, design rules for transforming these ER constructs into a relational model are presented, along with the associated integrity rules that can be derived from the conceptual design.

The results of this research have been used effectively in modeling various real-world applications. The framework provides a natural, consistent, and semantically rich conceptual design, even in situations that cannot be easily captured in a conceptual model. Most of the design implications were generated from the inadequacy of current modeling practice when applied to certain real-world situations. The results of this research are formalized as a set of design implications that are explicit enough to be implemented in a database design tool. We developed a prototype design system that incorporates most of these results.

There are several directions for future research. For example, the effectiveness of the results of this analysis can be assessed further by conducting an experiment with different user groups; important issues related to usability, expressiveness of the model, and correctness of the representation can be resolved. Another direction is to develop a learning database design system that not only uses the results of this work, but also learns from previous design sessions to develop better designs in the future. We are currently working on developing such a system.

ACKNOWLEDGMENTS

We wish to thank Won Kim, the editor-in-chief, Marianne Winslett, an associate editor, and the anonymous reviewers of the *ACM Transactions on Database Systems*; this paper has greatly benefited from their comments and suggestions

REFERENCES

- BATINI, C., CERI, S., AND NAVATHE, S. B. 1992. *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings, Redwood City, CA.
- BATRA, D. AND ANTONY, S. R. 1994. Novice errors in conceptual database design. *Eur. J. Inf. Syst.* 3, 1, 57–69.
- BATRA, D., HOFFLER, J. A., AND BOSTROM, R. P. 1990. Comparing representations with relational and EER models. *Commun. ACM* 33, 2 (Feb. 1990), 126–139.
- BATRA, D. AND ZANAKIS, S. 1994. A conceptual database design approach based on rules and heuristics. *Eur. J. Inf. Syst.* 3, 3, 228–239.

- BISKUP, J. 1995. Database schema design theory: Achievements and challenges. In *Proceedings of the 6th International Conference on Information Systems and Management of Data* (Bombay, India, Nov.), 14–44.
- CHEN, P. P. 1976. The entity-relationship model: Toward a unified view of data. *ACM Trans. Database Syst.* 1, 1, 9–36.
- DEY, D., BARRON, T. M., AND STOREY, V. C. 1995. A conceptual model for the logical design of temporal databases. *Decis. Support Syst.* 15, 4 (Dec. 1995), 305–321.
- ELMASRI, R. AND NAVATHE, S. B. 1994. *Fundamentals of Database Systems*. 2nd ed. Benjamin-Cummings Publ. Co., Inc., Redwood City, CA.
- GOLDSTEIN, R. AND STOREY, V. 1989. Some findings on the intuitiveness of entity-relationship constructs. In *Proceeding of the 8th International Conference on Entity-Relationship Approach to Database Design and Querying* (Toronto, Canada), 9–23.
- HAINAUT, J.-L., TONNEAU, C., JORIS, M., AND CHANDELON, M. 1993. Transformation-based database reverse engineering. In *Proceedings of the 12th International Conference on Entity-Relationship Approach* (Arlington, TX, Dec.), 364–375.
- HAWRYSZKIEWYCZ, I. T. 1990. *Relational Database Design: An Introduction*. Prentice-Hall, Inc., Upper Saddle River, NJ.
- HULL, R. 1986. Relative information capacity of simple relational database schemata. *SIAM J. Comput.* 15, 3, 856–886.
- HULL, R. AND KING, R. 1987. Semantic database modeling: Survey, applications, and research issues. *ACM Comput. Surv.* 19, 3 (Sept. 1987), 201–260.
- LAENDER, A. H. F. AND FLYNN, D. J. 1993. A semantic comparison of the modeling capabilities of the ER and NIAM models. In *Proceedings of the 12th International Conference on Entity-Relationship Approach* (Arlington, TX, Dec.), 237–252.
- LENZERINI, M. AND NOBILL, P. 1990. On the satisfiability of dependency constraints in entity-relationship schemata. *Inf. Syst.* 15, 4 (1990), 453–461.
- LLOYD-WILLIAMS, M. 1993. Expert system support for object-oriented database design. *Int. J. Appl. Exper. Syst.* 1, 4, 197–212.
- MARKOWITZ, V. M. AND SHOSHANI, A. 1992. Representing extended entity-relationship structures in relational databases: A modular approach. *ACM Trans. Database Syst.* 17, 3 (Sept. 1992), 423–464.
- MATTOS, N. M., MEYER-WEGENER, K., AND MITSCHANG, B. 1993. Grand tour of concepts for object-orientation from a database point of view. *Data Knowl. Eng.* 9, 4, 321–352.
- MCALLISTER, A. 1998. Complete rules for n -ary relationship cardinality constraints. *Data Knowl. Eng.* 27, 3, 255–288.
- MCALLISTER, A. J. AND SHARPE, D. 1998. An approach for decomposing N -ary data relationships. *Softw. Pract. Exper.* 28, 2, 125–154.
- NIJSSSEN, G. M. AND HALPIN, T. A., Eds. 1989. *Conceptual Schema and Relational Database Design: A Fact Oriented Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ.
- PARSONS, J. AND WAND, Y. 1997. Using objects for systems analysis. *Commun. ACM* 40, 12, 104–110.
- RAMESH, V. AND BROWN, G. J. 1999. Expressing casual relationships in conceptual database schemas. *J. Syst. Softw.* 45, 225–232.
- RAUH, O. AND STICKEL, E. 1993. Searching for compositions in ER schemes. In *Proceedings of the 12th International Conference on Entity-Relationship Approach* (Arlington, TX, Dec.), 75–86.
- ROCHFELD, A. AND NEGROS, P. 1992. Relationship of relationships and other inter-relationship links in E-R model. *Data Knowl. Eng.* 9, 205–221.
- SIAU, K., WAND, Y., AND BENBASAT, I. 1995. A psychological study on the use of relationship concept--Some preliminary findings. In *Proceedings of the 7th International Conference on Advanced Information Systems Engineering* (CAiSE '95, Jyväskylä, Finland), Springer-Verlag, Vienna, Austria, 341–354.
- SMITH, J. M. AND SMITH, C. P. S. 1977. Database abstractions: Aggregation and generalization. *ACM Trans. Database Syst.* 2, 2 (June), 105–133.
- SONG, I.-Y., EVANS, M., AND PARK, E. K. 1995. A comparative analysis of entity-relationship diagrams. *J. Comput. Softw. Eng.* 3, 4, 427–459.

- STOREY, V. C. AND GOLDSTEIN, R. C. 1988. A methodology for creating user views in database design. *ACM Trans. Database Syst.* 13, 3 (Sept. 1988), 305–338.
- STOREY, V. C. AND GOLDSTEIN, R. C. 1993. Knowledge-based approaches to database design. *Manage. Inf. Syst. Q.* 17, 1 (Mar.), 25–46.
- STOREY, V. C., CHIANG, R. H. L., DEY, D., GOLDSTEIN, R. C., AND SUDARESAN, S. 1997. Database design with common sense business reasoning and learning. *ACM Trans. Database Syst.* 22, 4, 471–512.
- TEOREY, T. J., YANG, D., AND FRY, J. P. 1986. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Comput. Surv.* 18, 2 (June 1986), 197–222.
- THALHEIM, B. 1992. Fundamentals of cardinality constraints. In *Proceedings of the 11th International Conference on Entity-Relationship Approach* (Karlsruhe, Germany, Oct. 7–23), 7–23.
- TSICHRITZIS, D. C. AND LOCHOVSKY, F. H. 1982. *Data Models*. Prentice-Hall, New York, NY.
- WILMOT, R. B. 1984. Foreign keys decrease adaptability of database designs. *Commun. ACM* 27, 12 (Dec. 1984), 1237–1243.

Received: July 1996; revised: June 1999; accepted: July 1999