INFORMATION SCIENCES **42**, 167–186 (1987)

A Practical Approach to Transforming Extended ER Diagrams into the Relational Model

DONGQING YANG

Computer Science and Technology, Peking University, Beijing, The People's Republic of China

TOBY J. TEOREY

Computing Research Laboratory, Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan 48109

and

JAMES P. FRY

Computer and Information Systems, Graduate School of Business Administration, The University of Michigan, Ann Arbor, Michigan 48109

ABSTRACT

Constructs for the extended ER model are defined and classified as a basis for transformation into the relational model. Transformation rules are developed for the different possible forms of n-ary relationships among entities including relationship connectivity, membership class, existence dependency, and generalization and subset hierarchies. Although normalization is not necessarily preserved under the transformation rules, it is feasible to implement a sequence of transformation and normalization operations without loss of information. This approach provides a practical solution to the modeling of real-world problems using first the extended ER model and then its refinement into the relational model.

1. INTRODUCTION

The emerging discipline of logical database design has been dominated by two data models: the entity-relationship (ER) model [4] and the relational model [6]. The ER model has been most successful as a tool for communication between the designer and end user during the requirements-analysis and conceptual-design phases because of its simplicity, ease of understanding, and convenience of representation. It is also being promoted as a basis for an actual

52 Vanderbilt Ave., New York, NY 10017 0020-0255/87/$03.50

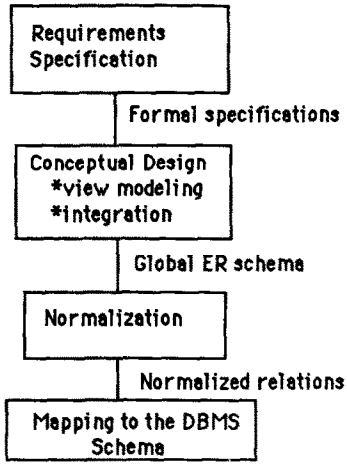


Fig. 1. Logical database design stages.

database implementation [3], competing with the more traditional hierarchical and network database management systems currently available.

The relational model, as well as being the basis for relational database systems, can also be used in the logical database design process for hierarchical and network databases (Figure 1). The concept of normalization [6] is often considered to be a vital part of logical design, regardless of the data model to be used for actual implementation, because normalization provides a higher level of data integrity and retrieval flexibility. Since some approaches to logical database design use both the ER model and the relational model in successive stages, it would be useful to develop a framework for transforming the variety of ER constructs into relations which can be normalized later.

Numerous extensions have been proposed to the original ER model in recent years. The most common extension studied has been the class of abstraction techniques such as generalization [28, 27, 1, 23, 26, 10, 19]. Other work has concentrated on existence constraints [30, 26] or more general integrity constraints [17, 24]. Ternary relationships and composite attributes are defined in [19].

There is also a large body of work devoted to the transformation of the ER model to the relational model. Most of the earlier work focused on the original ER model, consisting of entities, attributes, and binary relationships [31, 8, 13, 12]. Existence dependency was added by [30]. Later transformation algorithms included abstraction in an extended ER model [10, 19]. Transformations based on a normal form for ER models have a theoretical basis and a strong potential for future applications [5, 14, 15, 19]. We take a more pragmatic approach by

synthesizing recent research and applying it to current database model implementation.

We propose a simple taxonomy of extended ER (EER) constructs and a set of rules that will facilitate the transformation of the EER model to the relational model as a practical approach to logical database design. The EER transformations are proposed to preserve the entity definitions (to be relations) and concentrate on the interentity relationships to become new relations. An interentity relationship can be represented by a foreign key, either embedded in a "child" entity relation or combined with one or more other foreign keys in a relationship relation [22, 26, 21, 12].

Entity normalization is not necessarily preserved under such transformations. The introduction of a foreign key into an entity relation may result in additional functional dependencies in an otherwise normalized relation. For example, in Figure 2 the introduction of the foreign key DEPT-NO into the entity relation EMP creates a transitive functional dependency EMP-NO → DEPT-NO → OFFICE-BLDG. Relationship relations, consisting of two or more entity keys and possibly some intersection nonkey attributes, may also experience similar problems. Therefore the use of preliminary normalization of entities [11] will not guarantee normalized relations after the EER transformations. However, after the transformations such normalization can be accomplished using well known methods [32, 21, 20, 12]. We do, however, assume that the EER model has undergone preliminary analysis so that view integration has already been accomplished.

Section 2 reviews the EER model for different classes of objects, and Section 3 defines and categorizes the basic EER constructs that relate to each of these

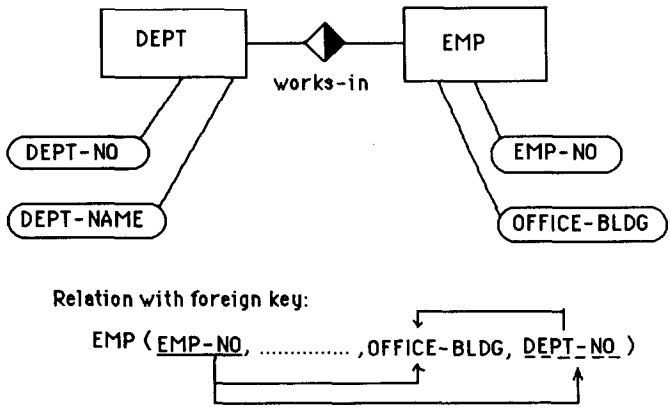


Fig. 2. Example ER-to-relation transformation losing normalization.

classes. Section 4 defines the transformation rules for each EER construct, and illustrates each rule using a common example. Integrity constraints on null values associated with each rule are also stated. The conclusion and implementation suggestions are given in Section 5.

2. EXTENDED ER (EER) MODEL

The entity-relationship approach initially proposed by Chen [4], although modified and extended, still remains the premier model for conceptual design. It is used to represent information in terms of entities, their attributes, and associations among entity occurrences called relationships.

The most important extension of the ER model has been the incorporation of abstraction techniques developed by Smith and Smith [28]. This extension has added two new objects into the model; subset hierarchies and generalization hierarchies [17, 23]. Another extension was the concept of conditional membership [1, 27].

2.1. ORIGINAL CLASSES OF OBJECTS

Initially, Chen proposed three classes of objects: entities, attributes, and relationships [Figure 3(a).] Entities were the principal objects about which information was to be collected, usually denoting a person, place, thing, or event of informational interest. Attributes were used to detail entities by giving them descriptive properties such as name, color, and weight. As indicated in

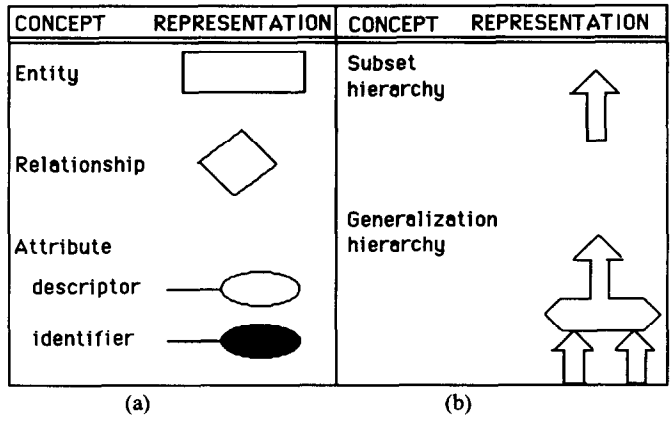


Fig. 3. Extended-ER (EER) model representations.

Figure 3(a), there are two types of attributes: identifiers and descriptors. The former is used to uniquely distinguish among the occurrences of a entity, while the latter is purely descriptive.

Entities can be distinguished by the "strength" of their identifying attributes. Strong entities have internal identifiers that uniquely determine the existence of entity occurrences. Weak entities derive their existence from the identifying attributes (sometimes called external attributes) of one or more "higher-level" entities. Relationships have semantic meaning which is indicated by the connectivity between entity occurrences (one-to-one, one-to-many, and many-to-many) and the participation in this connectivity by the member entities. This participation can be conditional or unconditional. For example, each of the objects person may or may not have a spouse. Finally, each of the objects was named, often having one or more synonyms associated with it. The diagrams for these objects are shown in Figure 3(a).

2.2. *EXTENDED CLASSES OF OBJECTS*

The original ER model has long been used effectively for communicating fundamental data and relationship definitions with the database end user. Using the ER model as a conceptual schema to be refined to a relational schema has proved quite difficult without extending the basic concepts to include various forms of abstraction or n-ary relationships of degree greater than two. The extended ER model provides simple representations for these commonly used concepts and is compatible with the simplicity of the original model.

The introduction of abstraction into the ER model resulted in two types of objects: subset hierarchies and generalization hierarchies (see Figures 3b and 5). The subset hierarchy specifies possibly overlapping subsets, while the generalization hierarchy specifies strictly nonoverlapping subsets [23, 10]. Both subset objects will transform equivalently to a relational data model scheme, but they will differ significantly in update integrity rules (see Section 4.4).

SUBSET HIERARCHY DEFINITION. An entity E_1 is a subset of another entity E_2 if every occurrence of E_1 is also an occurrence of E_2.

A subset hierarchy is the case when every occurrence of the generic entity may also be an occurrence of the other entities which are potentially overlapping subsets. For example, the entity "employee" (EMP) may include "employees attending college," "employees who hold political office," or "employees who are also shareholders" as different forms of a special classification.

GENERALIZATION HIERARCHY DEFINITION. An entity E is generalization of the entities $E_1, E_2, \ldots, E_n$ if each occurrence of E is also an occurrence of one and only one of the entities $E_1, E_2, \ldots, E_n$.

A generalization hierarchy occurs when an entity (which we call the generic entity) is partitioned by different values of a common attribute. For example, EMP is a generalization of PROFESSOR, SECRETARY, and TECHNICIAN. The generalization object (EMP) is called an IS-A exclusive hierarchy because each occurrence of the entity EMP is an occurrence of one and only one of the entities PROFESSOR, SECRETARY, TECHNICIAN.

3. FUNDAMENTAL EER CONSTRUCTS

We now define the fundamental EER constructs needed for specifying the transformations from the EER model to the relational model.

3.1. DEGREE OF A RELATIONSHIP

The degree of a relationship is the number of entities (entity sets) associated in the relationship. An n-ary relationship is of degree n. Unary, binary, and ternary relationships are special cases where the degree is 1, 2, and 3 respectively. This is indicated in part 1 of Figure 4.

3.2. CONNECTIVITY OF A RELATIONSHIP

The connectivity of a relationship specifies the mapping of the associated entity occurrences in the relationship. Values for connectivity are either "one" or "many." The actual number associated with the term "many" is called the cardinality of the connectivity. Cardinality may be represented by upper and lower bounds. Part 2 of Figure 4 shows the basic constructs for connectivity: one-to-one (unary or binary relationship), one-to-many (unary or binary relationship), and many-to-many (unary or binary relationship). A many-to-many-to-many ternary relationship is shown in part 1 of Figure 4. The shaded area in the unary or binary relationship diamond represents the "many" side, while the unshaded area represents the "one" side. We will use an n-sided figure to represent n-ary relationships for $n > 2$ in order to explicitly show each entity associated in the relationship to be either "one" or "many" related to the other entities. Each corner of the n-sided figure connects to an entity. A shaded corner denotes "many," and an unshaded corner denotes "one".

The ternary relationship illustrates a type of association that is much more complex than either a unary or binary relationship. An entity in a ternary

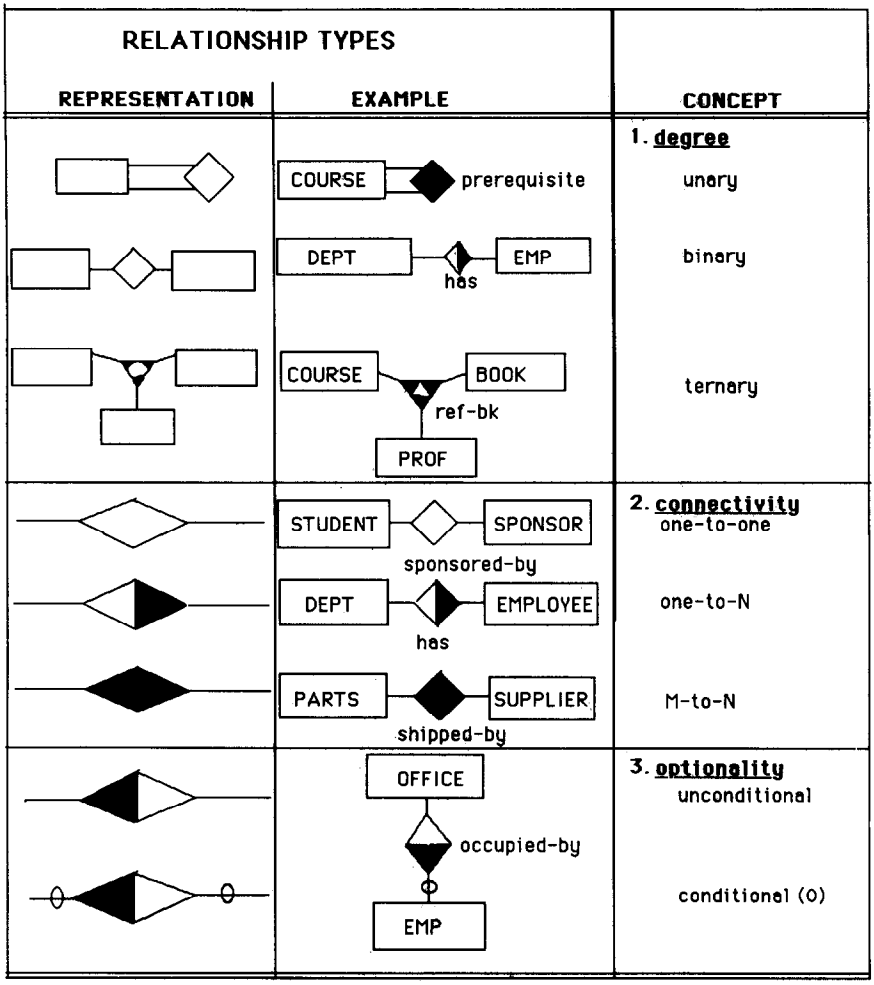


Fig. 4. Fundamental EER constructs: relationship types.

relationship is considered to be "one" if only one occurrence of it can be associated with one occurrence of each of the other two associated entities. It is "many" if more than one occurrence of it can be associated with one occurrence of each of the other associated entities. In the various parts of Figure 9 we see that each entity considered to be on the "one" side appears only on the right-hand side of a functional dependency. No entity considered "many" ever appears on the right-hand side of a functional dependency [19].

3.3. *MEMBERSHIP CLASS IN A RELATIONSHIP*

Membership class (or optionality) specifies whether either the "one" or "many" side in a relationship is unconditional or conditional. If an occurrence of the "one"-side entity must always exist to maintain the relationship, then it is unconditional. When an occurrence of that entity need not exist, it is considered conditional. The "many" side of a relationship is similarly unconditional if at least one entity occurrence must exist, and conditional otherwise. The conditional membership class, defined by "0" on the connectivity line between an entity and a relationship, is shown in Figure 4.

3.4. *EXISTENCE DEPENDENCY OF AN ENTITY IN A RELATIONSHIP*

A strong entity is shown with a single-bordered rectangle, while a weak entity is depicted with a double-bordered rectangle (Figure 5).

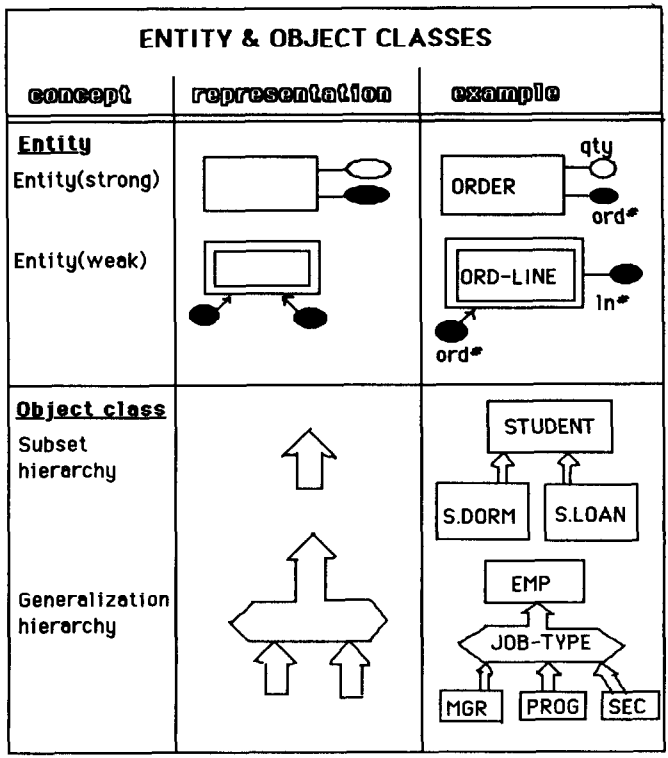


Fig. 5. Fundamental EER constructs: entity and object classes.

3.5. *OBJECT CLASS OF ENTITIES AND RELATIONSHIPS*

The basic objects are the *n*-ary relationships with their associated entities. Objects resulting from abstraction are the generalization hierarchy and the subset hierarchy (Figure 5). The generalization hierarchy implies that the subsets are a full partition, such that the subsets are disjoint and their union makes up the full set. The subset hierarchy implies that the subsets are potentially overlapping.

4. TRANSFORMATION RULES

Let us now look at each EER construct in more detail to see how each transformation rule is defined and applied. Our example is drawn from the university database EER schema illustrated in Figure 6. All types of EER constructs we must transform to relations are shown at least once in the figure.

We note that the basic transformations result in three types of relations:

(1) entity relation (with the same information content as the original entity),
(2) entity relation (with the embedded foreign key of the parent entity),
(3) relationship relation (with the foreign keys of all the entities that are thus related).

4.1. *TWO ENTITIES, ONE RELATIONSHIP*

The one-to-one relationship between entities is illustrated in Figure 7(a), (b), and (c). When both entities are unconditionally related [Figure 7(a)], each entity becomes a relation and the key of either entity can appear in the other entity's relation as a foreign key. One of the entities in a conditional relationship [see DEPT in Figure 7(b)] should contain the foreign key of the other entity in its transformed relation. The other entity (EMP) could also contain a foreign key (of DEPT), with nulls allowed, but would require more storage space because of the much greater number of EMP entity occurrences than DEPT entity occurrences. When both entities are conditionally related [Figure 7(c)], either entity could contain the embedded foreign key of the other entity, with nulls allowed.

The one-to-many relationship is shown as either unconditional or conditional on the "many" side without affecting the transformation. On the "one" side it may be either unconditional [Figure 7(d)] or conditional [Figure 7(e)]. In all cases the foreign key must appear on the "one" side, which represents the child entity, with nulls allowed for foreign keys, only in the conditional case.

The many-to-many relationship, shown here as totally conditional, requires a relationship relation with primary keys of both entities [Figure 7(f)]. The same

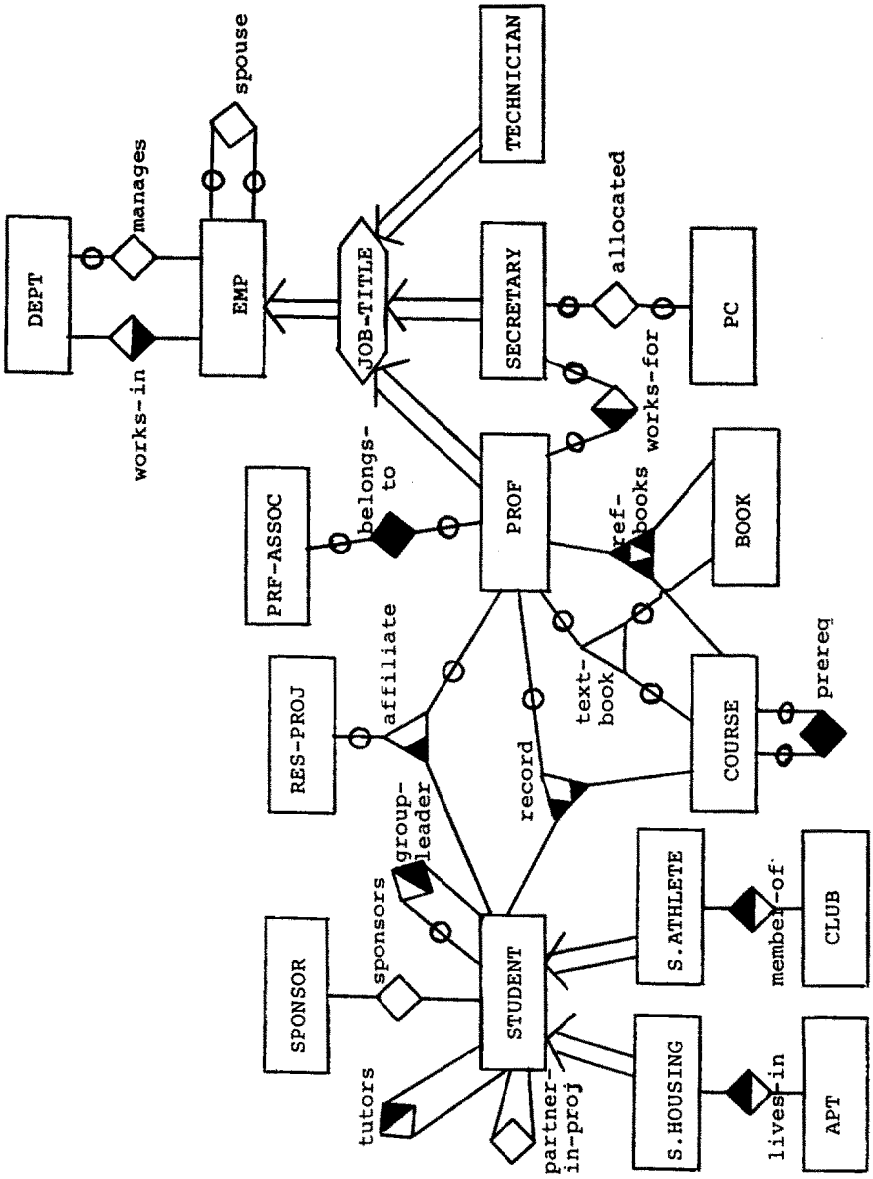


Fig. 6. University database EER diagram.

transformation applies to either the conditional or unconditional case. Embedded foreign keys are not possible because of the "many" property in both directions.

4.2. *ONE ENTITY, ONE RELATIONSHIP*

One entity with a one-to-one relationship implies some form of entity-occurrence pairing, as specified by the relationship name, and this must be either completely conditional or completely unconditional. The unconditional case [Figure 8(a)] and the conditional case [Figure 8(b)] both require a foreign key in the entity relation, without nulls. The one-to-many relationship requires a

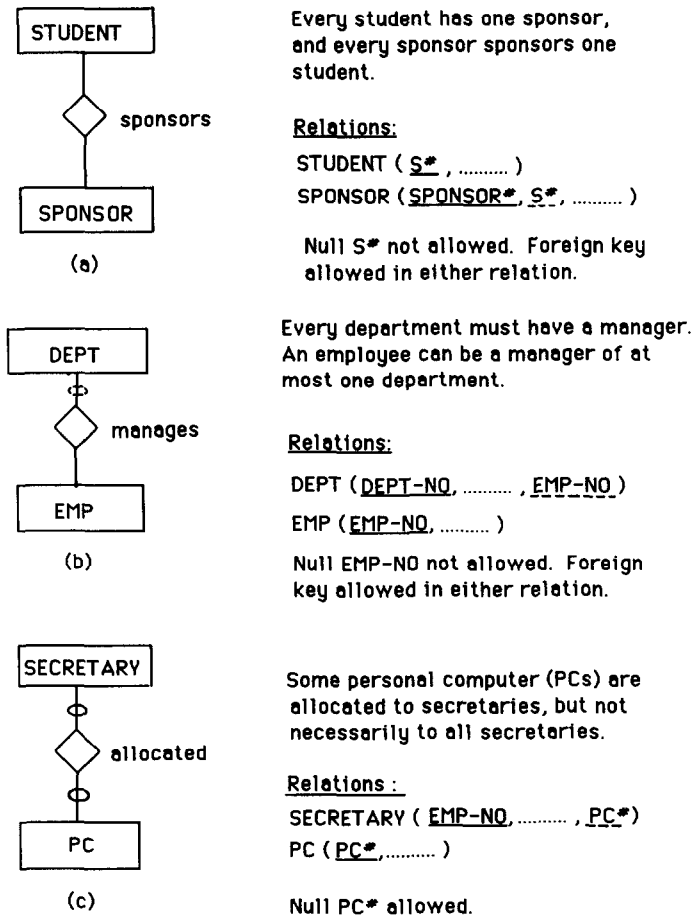
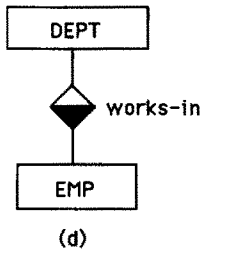


Fig. 7. Binary-relationship transformation rules.

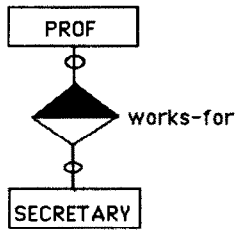


(d)

Every employee works in exactly one department. Every department could contain many employees.

Relations :
DEPARTMENT(DEPT-NO, ...)
EMPLOYEE(EMP-NO, ... , DEPT-NO)

Null DEPT-NO not allowed .

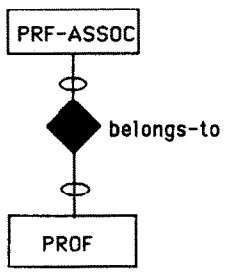


(e)

Each professor can have at most secretary. One secretary could work for several professors, or none.

Relations :
PROF (EMP-NO, ... , EMP'-NO)
SECRETARY (EMP'-NO, ...)

Null EMP'-NO allowed in PROF.



(f)

Every professional association could have many (or no) members who are professors. Every professor could belong to many professional associations (or none).

Relations :
PRF-ASSOC (PA#, ...)
PROF (EMP-NO, ...)
BELONGS-TO (PA#,EMP-NO)

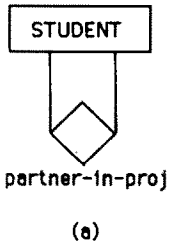
Null PA# or EMP-NO not allowed

Fig. 7. continued.

foreign key in the entity relation for both the conditional case [Figure 8(c)], with nulls allowed, and the unconditional case [Figure 8(d)], with nulls not allowed. The many-to-many relationship is shown as conditional [Figure 8(e)] and uses a relationship relation; it could also be defined as unconditional (using the word "must" instead of "may" in the narrative), but having the same transformation as the conditional case.

4.3. N ENTITIES, ONE RELATIONSHIP ($n > 2$)

The allowable varieties of an n-ary relationship are the $n+1$ possible allocations of entities with "many" connectivity (and cardinality from 0 to n).

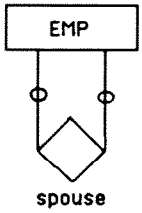


Every student has exactly one of the other students as a partner in a project.

Relation:

STUDENT (S#, ... , S'#)

Null S'# not allowed.



An employee could have one of the the other employees as his or her spouse.

Relation:

EMP (EMP-NO, ... , EMP'-NO)

Null EMP'-NO allowed in EMP

Students are divided into groups for certain projects. Each group has a leader.

Relation:

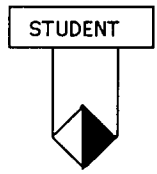
STUDENT (S#, ... , S'#)

Null S'# allowed.

Fig. 8. Unary-relationship transformation rules.

Thus, the ternary relationship in Figure 9 has four possible varieties. All varieties are transformed by creating a relationship relation containing the primary keys of all n entities; however, in each case the meaning of the keys is different. When all relationships are "one" [Figure 9(a)], the relationship relation consists of n possible distinct candidate keys, each consisting of $n-1$ entity keys. This represents the fact that there are n functional dependencies (FDs) needed to describe this relationship. The conditional "one" allows null foreign keys, but the unconditional "one" does not.

When all relationships are "many" [Figure 9(b)], the relationship relation is all key, and no FDs are present. In general the number of entities with connectivity "one" determines the number of FDs, and each determinant of an FD (or the case with all key) determines the candidate key of the relationship relation.



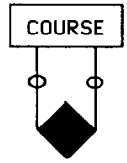
(d)

Every student tutors one of the other students. One student may be tutored by several other students.

Relation:

STUDENT (S#, ... , S'#)

Null S'# not allowed.



(e)

Each course may have many prerequisite courses, and each course could be a prerequisite for many other courses.

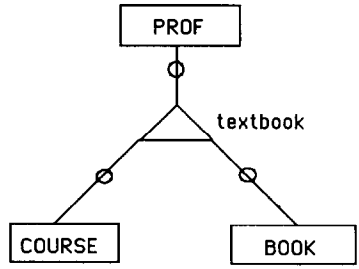
Relations:

COURSE (COURSE#, ...)

PREREQ (COURSE#, COURSE'#)

Null COURSE'# allowed.

Fig. 8. continued.



A professor will choose at most one textbook for a given course. Different professors use different textbooks for the same course. No professor will use the same textbook for different courses.

Relations:

PROF (EMP-NO, ...)

COURSE (COURSE#, ...)

BOOK (BOOK#, ...)

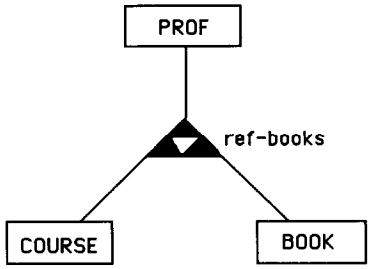
TEXTBOOK (EMP-NO,COURSE#, BOOK#)

Nulls allowed for any keys in TEXTBOOK.

FDs: EMP-NO,COURSE# --> BOOK#
EMP-NO,BOOK# --> COURSE#
BOOK#,COURSE# --> EMP-NO

(a)

Fig. 9. Ternary-relationship transformation rules.



Professors use a wide range of different reference books for each course they teach.

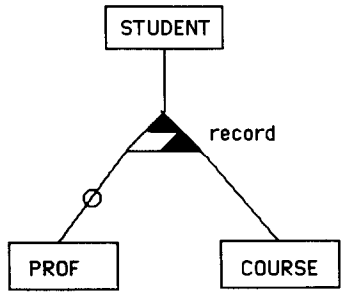
Relations:

PROF (EMP-NO, ...)
COURSE (COURSE#, ...)
BOOK (BOOK#, ...)
REF-BOOKS (EMP-NO,COURSE#,BOOK#)

Nulls allowed for any key in REF-BOOKS.

FDs: EMP-NO,COURSE#,BOOK# (all key)

(b)



Records about students taking courses are kept by the professors.

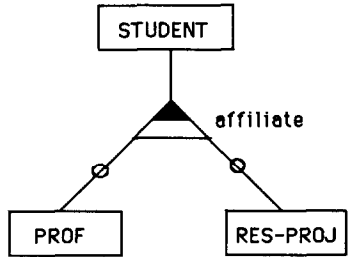
Relations:

PROF (EMP-NO, ...)
STUDENT (S#, ...)
COURSE (COURSE#, ...)
RECORD (EMP-NO, S#,COURSE#)

Nulls allowed for any foregn key in RECORD.

FDs: S#,COURSE# --> EMP-NO

(c)



A student must have a different professor for each research project.

Relations:

STUDENT (S#, ...)
PROF (EMP-NO, ...)
RES-PROJ (PROJ#, ...)
AFFILIATE (EMP-NO,S#,PROJ#)

Nulls allowed for any foreign key in AFFILIATES.

FDs: S#,PROJ# --> EMP-NO
S#,EMP-NO --> PROJ#

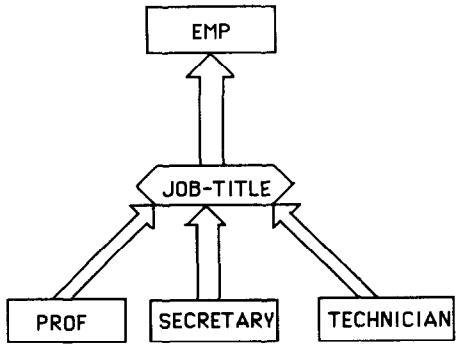
(d)

Fig. 9. continued.

4.4. *GENERALIZATION AND SUBSET HIERARCHIES*

The generalization hierarchy resulting in disjoint subsets is produced by partitioning the generic entity by different values of a common attribute, e.g. JOB-TITLE in Figure 10. The transformation of disjoint subset generalization produces a separate relation for the whole set (the generic entity) and each of the subsets. The generic entity relation contains the generic entity key and all common attributes (including the attribute used for partitioning). Each subset relation contains the generic entity key and only attributes specific to that subset. Update integrity is maintained by requiring all insertions and deletions to occur in both the set (generic entity) relation and relevant subset relation. If the change is to the key, then one subset as well as the set relation must be updated. Changes to a nonkey attribute affects either the set or one subset relation.

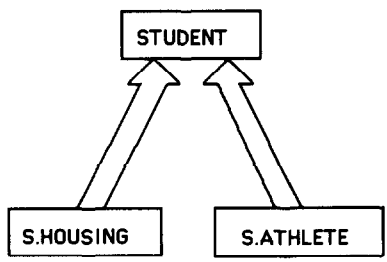
Overlapping subsets are produced by partitioning the generic entity by values of different attributes (Figure 11). The transformation of this construct produces separate relations for the generic entity and each of the subset entities. The key of each relation is the key of the generic entity; and while the generic



Different types of employees are partitioned by values of a common attribute JOB-TITLE.

Relations:
EMP (EMP-NO, JOB-TITLE, common attributes)
EMP.PROF (EMP-NO, specific attributes)
EMP.SECRETARY (EMP-NO, specific attributes)
EMP.TECHNICIAN (EMP-NO, specific attributes)

Fig. 10. Generalization hierarchy.



Students with special situations are shown as overlapping subsets based on partitions on values of different attributes.

Relations:
STUDENT (S#, common attributes)
S.HOUSING (S#, specific attributes)
S.ATHLETE (S#, specific attributes)

Fig. 11. Subset hierarchy.

entity relation contains only common attributes, the subset relations contain attributes specific to that subset entity. Thus, the transformation rules for the disjoint and overlapping subsets are the same.

The integrity rules between these two cases are different, however. With overlapping subsets, deletion from the set (generic entity) relation cascades to anywhere from none to all of the subsets. Also, before insertion to a subset relation, it is necessary to check whether a tuple with the same key value exists in the set relation. A change to a nonkey attribute affects the set or one of the subsets. A change to a key affects the set and at least one subset.

4.5. MULTIPLE RELATIONSHIPS

Multiple relationships among n entities are always considered to be completely independent. Each relationship produces entity relations that are either equivalent or differ only in the addition of a foreign key, and thus can be consolidated into a single entity relation containing all foreign keys. Relationship relations so produced are unique and cannot be consolidated.

4.6. EXISTENCE DEPENDENT (WEAK) ENTITIES

Weak entities differ from (strong) entities only in the need for keys from other entities to establish their unique identities. Otherwise they have the same transformation properties as strong entities, and no special rules are needed. When a weak entity is already derived from two or more strong entities in the ER diagram, it can be directly transformed into a relationship relation without further change.

4.7. AGGREGATION

The aggregation abstraction [28] can occur among entities, or it can relate attributes to a single entity. Aggregation among entities, defined by the PART-OF relationship, is a special case of the collection of one-to-many binary relationships and can be transformed as defined in Section 4.1. An example given in [29] describes BICYCLE as the whole entity and SEAT, PEDALS, HANDLEBARS, etc. as its parts, each part being an entity with its own distinct attributes.

5. CONCLUSIONS

We have shown how to simply transform EER diagrams into the relational model. A taxonomy of EER constructs was defined, and a transformation rule specified for each construct.

Implementation and verification of the transformation algorithm is currently under investigation. The following steps summarize the basic concepts in the formation of relations from the EER diagram:

(1) Define a data object to represent each set of n entities with a relationship among them (for $n > 0$). Every relationship is thus represented by a unique data object.

(2) Write the transformation rule of each object in terms of relations with embedded foreign keys and relations with combined keys. Account for relationship degree, connectivity, membership class, and object class (generalization, etc.).

(3) Combine relations having the same attributes (except for the foreign keys, which may be the same or different).

Later, each relation can be further normalized, and redundant relations can be eliminated.

REFERENCES

1. P. Atzeni, C. Batini, M. Lenzerini, and F. Villanelli, INCOD: A system for conceptual design of data and transactions in the entity-relationship model, in *Entity-Relationship Approach to Information Modeling and Analysis*, ER Inst., Saugus, Calif., 1981.
2. P. Bertaina, A. DiLeva, and P. Giolito, Logical design in CODASYL and relational environments, in *Methodology and Tools for Data Base Design* (S. Ceri, Ed.), North-Holland, 1983, pp. 85–117.
3. T. C. Chiang and R. F. Bergeron, A data base management system with an E-R conceptual model, in *Entity-Relationship Approach to Systems Analysis and Design* (P. Chen, Ed.), North-Holland, Amsterdam, 1980, pp. 467–476.
4. P. Chen, The entity-relationship model—toward a unified view of data, *ACM Trans. Database Systems* 1(1):9–36 (Mar. 1976).
5. I. Chung, F. Nakamura, and P. P. Chen, A decomposition of relations using the entity relationship approach, in *Entity-Relationship Approach to Information Modeling and Analysis* (P. Chen, Ed.), ER Inst., 1981.
6. E. F. Codd, A relational model for large shared data banks, *Comm. ACM* 13(6):377–387 (June 1970).
7. E. F. Codd, Recent investigations into relational data base systems, in *Proceedings of the IFIP Congress*, 1974.
8. C. J. Date, *An Introduction to Database Systems*, Vol. 1 (3rd ed.), Vol. 2, Addison-Wesley, Reading Mass., 1983.
9. C. J. Date, *A Guide to DB2*, Addison-Wesley, Reading, Mass., 1984.
10. R. Elmasri, A. Hevner, and J. Weeldreyer, The category concept: An extension to the entity-relationship model, *Data and Knowledge Engrg.* 1(1):75–116 (1985).
11. E. N. Fong, M. W. Henderson, D. K. Jefferson, and J. M. Sullivan, *Guide on Logical Database Design*, NBS Spec. Pub. 500-122, U.S. Dept. of Commerce, Feb. 1985, 115 pp.
12. I. T. Hawryszkiewycz, *Database Analysis and Design*, SRA, Chicago, 1984.
13. D. R. Howe, *Data Analysis and Data Base Design*, Arnold, London, 1983.

14. S. Jajodia and P. A. Ng, On the representation of relational structures by entity-relationship diagrams, in *The Entity-Relationship Approach to Software Engineering* (G. C. Davis et al., Eds.), Elsevier North-Holland, New York, 1983, pp. 223–248.
15. S. Jajodia and P. A. Ng, Translation of entity-relationship diagrams into relational structures, *J. of Systems and Software* 4 (1984), pp. 123–133.
16. W. Kent, Fact-Based Data Analysis and Design, *J. Systems and Software* 4:99–121 (1984).
17. M. Lenzerini and G. Santucci, Cardinality constraints in the entity-relationship model, in *The Entity-Relationship Approach to Software Engineering*, (G. C. Davis et al., Eds.), Elsevier North-Holland, New York, 1983, pp. 529–549.
18. Y. E. Lien, On the equivalence of data models, *J. Assoc. Comput. Mach.* 29 (2):333–362 (1982).
19. T. Ling, A normal form for entity-relationship diagrams, in *Proceedings of the Fourth International Conference on the E-R Approach*, Chicago, IEEE Society Press, 1985, pp. 24–35.
20. D. Maier, *Theory of Relational Databases*, Computer Science Press, Rockville, Md., 1983.
21. J. Martin, *Managing the Data-Base Environment*, Prentice-Hall, Englewood Cliffs, N.J., 1983.
22. W. C. McGee, A contribution to the study of data equivalence, in *Data Base Management* (J. W. Klimbie and K. L. Koffeman, Eds.), North-Holland, Amsterdam, 1974, pp. 123–148.
23. S. B. Navathe and A. Cheng, A methodology for database schema mapping from extended entity relationship models into the hierarchical model, in *The Entity-Relationship Approach to Software Engineering* (G. C. Davis et al., Eds.), Elsevier North-Holland, New York, 1983, pp. 223–248.
24. O. Oren, Integrity constraints in the conceptual schema language SYSDOC, in *Proceedings of the 4th International Conference on the E-R Approach*, Chicago, IEEE Society Press, 1985, pp. 288–294.
25. D. Reiner, et al. The database design and evaluation workbench (DDEW) project at CCA, *Database Engrg.* 7, No. 4 (Dec. 1984).
26. H. Sakai, Entity-relationship approach to logical database design, in *Entity-Relationship Approach to Software Engineering* (C. G. Davis, S. Jajodia, P. A. Ng, and R. T. Yeh, Eds.), Elsevier Science, New York, 1983, pp. 155–187.
27. P. Scheuermann, G. Schiffner, and H. Weber, Abstraction capabilities and invariant properties modelling within the entity-relationship approach, in *Entity-Relationship Approach to Systems Analysis and Design* (P. Chen, Ed.), North-Holland, Amsterdam, 1980, pp. 121–140.
28. J. M. Smith and D. C. P. Smith, Database abstractions: Aggregation and generalization, *ACM Trans. Database Systems* 2(2):105–133 (June 1977).
29. T. J. Teorey and J. P. Fry, *Design of Database Structures*, Prentice-Hall, Englewood Cliffs, N.J., 1982.
30. N. Webre, An extended E-R model and its use on a defense project, in *Proceedings of the 2nd International Conference on the E-R Approach*, Washington, 1981, pp. 175–194.
31. E. Wong and R. H. Katz, Logical design and schema conversion for relational and DBTG databases, in *Entity-Relationship Approach to Systems Analysis and Design* (P. Chen, Ed.), 1980, pp. 311–322.
32. P. Bernstein, Synthesizing 3NF relations from functional dependencies. *ACM Trans. Database Syst.* 1,4:272–298 (1976).

Received 15 August 1985; revised 15 June 1986, 15 February 1987