LBL-28482
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

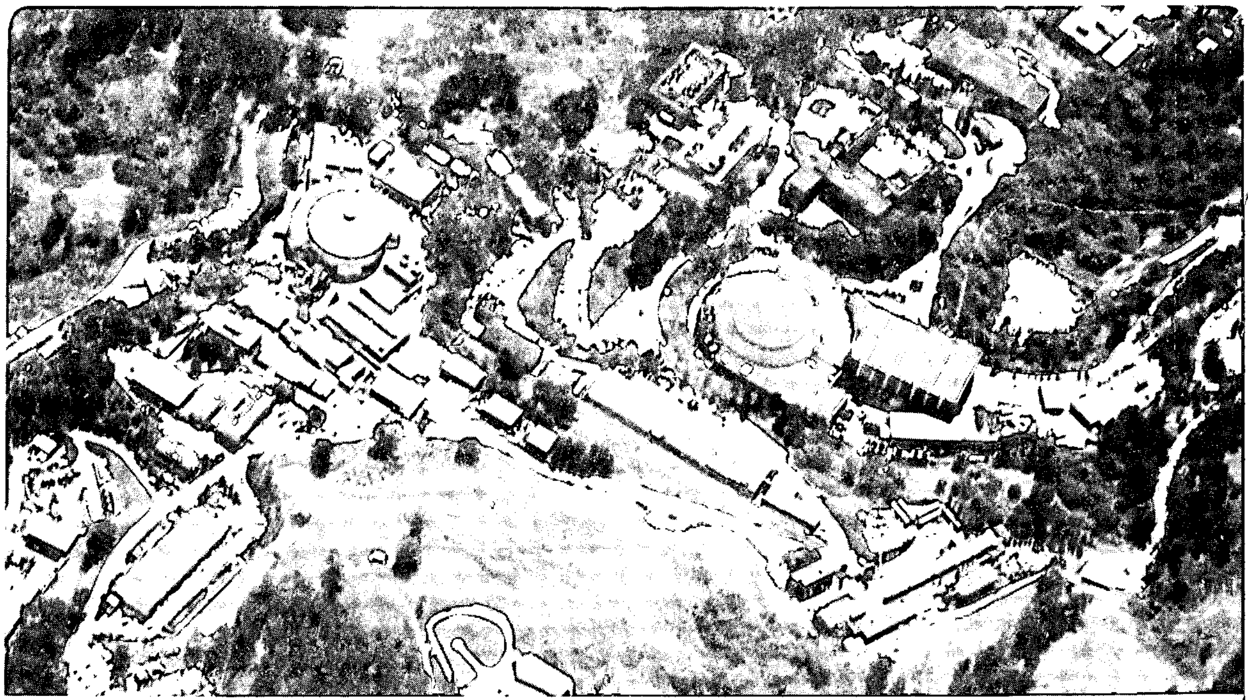
Information and Computing Sciences Division

Submitted to ACM Transactions on Database Systems

Representing Extended Entity-Relationship Structures in Relational Databases: A Modular Approach

V.M. Markowitz and A. Shoshani

August 1991



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

LBL-28482

REPRESENTING EXTENDED ENTITY-RELATIONSHIP STRUCTURES IN RELATIONAL DATABASES : A MODULAR APPROACH*

Victor M. Markowitz†
Arie Shoshani‡

Data Management Group
Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

Revised

August 1991

To be published in the *ACM Transactions on Database Systems*

* This work was supported by the Applied Mathematical Sciences Research Program and the Office of Health and Environmental Research Program, of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC03-76SF00098.
† Author's E-mail address: *V_Markowitz@lbl.gov* Office phone number:(415) 486-6835
‡ Author's E-mail address: *A_Shoshani@lbl.gov* Office phone number:(415) 486-5171

CONTENTS

ABSTRACT

A common approach to database design is to describe the structures and constraints of the database application in terms of a semantic data model, and then represent the resulting schema using the data model of a commercial database management system. Often in practice, *Extended Entity-Relationship* (EER) schemas are translated into equivalent relational schemas. This translation involves different aspects: representing the EER schema using relational constructs, assigning names to relational attributes, normalization, and merging relations. Considering these aspects together, as usually done in the design methodologies proposed in the literature, is confusing and leads to inaccurate results. We propose to treat separately these aspects, and split the translation into four stages (modules) corresponding to the four aspects mentioned above. We define criteria for both evaluating the correctness of and characterizing the relationship between alternative relational representations of EER schemas.

Categories and Subject Descriptors: H.2.1[**Database Management**] Logical Design - *data models*; *normal forms*; H.2.3[**Database Management**] Languages - *data description language (DDL).*

General Terms: Algorithms, Design.

Additional Key Words and Phrases: Database design, extended entity-relationship model, relational data model, schema translation, semantic data model.

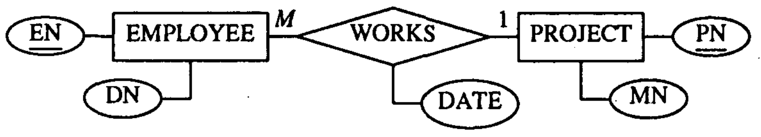
1. INTRODUCTION

1.1 Background

Many database design methodologies involve describing the database application in terms of a semantic data model, and then translating the resulting schema into a relational schema [10]. Semantic data models support the specification of (classes of) objects and provide various abstraction mechanisms, such as generalization and aggregation, and thus have more semantic intuition than the relational data model. We consider in this paper a version of the *Extended Entity-Relationship* (EER) model [24], which extends one of the most popular semantic data models, the *Entity –Relationship* (ER) model [6]; in addition to the basic constructs of the ER model, the EER model includes the generalization and full aggregation constructs. We selected the EER model because of its widespread use in designing relational databases, and because extensive work has already been done on translating EER schemas into relational schemas. In this paper we show that EER schemas can be accurately represented using relation-schemes, functional dependencies, key-based inclusion dependencies (also called *referential integrity constraints* [7]), and null constraints. However, the translation process includes several aspects that can be confusing when considered together, as discussed in the next section.

1.2 Problems with Translating ER and EER Schemas into Relational Schemas

Several methods have been proposed for translating both ER and EER schemas into relational schemas (e.g. [6], [11], [24], [25]). The informal nature of most of these proposals does not allow a precise evaluation of the correctness of the various translations. Consider, for example, the simple ER schema of figure 1.1(i) represented by an ER diagram in the usual way (entity-identifiers are underlined). Translations such as that presented in [24] generate for this ER schema the relational schema of figure 1.1(ii) (relational keys are underlined); yet another possible representation (e.g. following [11]) of the same ER

(i)


(ii) EMPLOYEE (EN, DN, PN, DATE)
PROJECT (PN, MN)

(iii) EMPLOYEE (EN, DN)
WORKS (EN, PN, DATE)
PROJECT (PN, MN)

(iv) WORKS [EN] $\subseteq$ EMPLOYEE [EN]
WORKS [PN] $\subseteq$ PROJECT [PN]

Abbreviations : EN=EMPLOYEE NAME, DN=DEPARTMENT NUMBER, MN=MANAGER NAME, PN=PROJECT NUMBER

Figure 1.1 Relational Representations for an Entity-Relationship Schema.

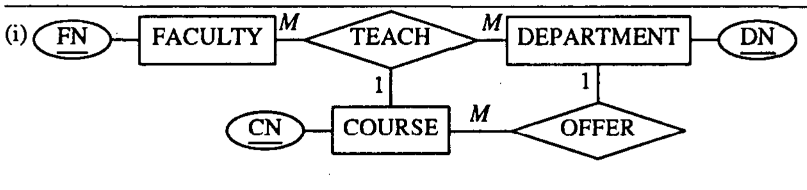
schema is the relational schema of figure 1.1(iii). The natural question to ask is: which representation is the *correct* one, and, if both are correct, how is the *equivalence* of such representations defined? We show in this paper that relational representations such as those of figures 1.1(ii) and 1.1(iii) are inaccurate because they allow database states which are inconsistent with respect to the ER semantics.

The following simple example illustrates the violation of an integrity constraint implied by an ER schema. The relational schema of figure 1.1(ii) can be associated with the following state: $s = \{r_{\text{EMPLOYEE}} = [\text{e1 d1 p1 y1}], r_{\text{PROJECT}} = [\text{p2 e2}]\}$. This state is inconsistent with respect to the semantics of the ER schema of figure 1.1(i), because the tuple representing a relationship of relationship-set WORKS includes data on a PROJECT entity that is not represented in *s*. This can be fixed by adding to the relational schema of figure 1.1(ii) the inclusion dependency $\text{EMPLOYEE[PN]} \subseteq \text{PROJECT[PN]}$ which will not allow states like *s*, but will also not allow the insertion of employees which are not associated with some project, contrary to the semantics of the corresponding ER schema. This, in turn, can be fixed by allowing *null* values for attribute PN in relation EMPLOYEE, but then additional constraints must be enforced in order to allow states that are consistent with the semantics of the ER schema. In the schema of figure 1.1(ii), for example, since DATE characterizes relationship-set WORKS, the corresponding relational attribute should be constrained to have a null value whenever PN has a null value in relation EMPLOYEE.

Following the same reasoning as above, we can conclude that the relational schema of figure 1.1(iii) must include the inclusion dependencies of figure 1.1(iv). Indeed, it can be shown that the schema of figure 1.1(iii) together with the inclusion dependencies of figure 1.1(iv) provide a correct representation for the ER schema of figure 1.1(i), provided that every relational attribute is allowed to have null values iff the corresponding ER attribute is allowed to have null values. We still could ask whether there are other equivalent representations for this ER schema. For example, following [11] the relational schemas of figures 1.1(ii) and 1.1(iii) can both represent the ER schema of figure 1.1(i) and are equivalent. The problem is that equivalence is defined in [11] without taking inclusion dependencies and null constraints into account. In this paper our definitions of correctness and equivalence for relational representations of EER schemas take into account the inclusion dependencies and null constraints implied by the EER schema.

Following the translation of an ER or EER schema into a relational schema, one may want to analyze the relational schema in order to establish its normal form, and to further normalize it if needed ([24], [25]). The combination of these two design methodologies (i.e. EER to relational schema translation and normalization) can be problematic because normalization is underlied by complex assumptions regarding the assignment of names to relational attributes [15]. Thus, different authors reach contradictory results concerning the possibility of such a combination, namely that it can be achieved (i) by first restructuring the ER schema [4]; (ii) without restricting the ER schema allowed [11]; and (iii) by applying certain restrictions on the ER schema allowed [2]. We refer below to the last two conclusions. The translation of [11] allows the representation of multiple relationship-sets involving common entity-sets. Thus, following [11], the ER schema of figure 1.2(i) is represented by the relational schema of figure 1.2(ii). The attribute names in this schema, however, do not satisfy the *Unique Role Assumption*† (URA) [17] and therefore normalization cannot be applied. Unlike [11], [2] contends that only ER schemas represented by ER diagrams without cycles can be represented by relational schemas that satisfy URA. This means, for instance, that schemas such as that shown in figure 1.2(i) cannot be represented by relational schemas that satisfy URA. In fact, this restriction can be avoided. We show in this paper that translating EER schemas into relational schemas and normalization can be combined without either restricting or restructuring the original EER schemas, but only by using an appropriate name assignment for relational attributes.

The assumptions that underlie normalization should not be taken lightly since even small inaccuracies lead to erroneous conclusions. For example, based on the translation proposed in [11], [12] concludes that in order to guarantee *Boyce-Codd Normal Form* (BCNF) for a relational schema translation of an ER schema, multiple relationship-sets that involve the same entity-sets cannot be allowed. Thus, the relational schema of figure 1.2(ii) does indeed contain a relation (TEACH) which is not in BCNF apparently because in the ER schema of figure 1.2(i) entity-sets COURSE and DEPARTMENT are involved together in two relationship-sets. However, since the translation presented in [11] does not satisfy URA, this result is based on an incorrect analysis. In this paper we show that every EER schema has an equivalent BCNF relational schema translation, and therefore there is no need to restrict the EER schemas

(i)


(ii)

FACULTY	($\underline{\text{FN}}$)
DEPARTMENT	($\underline{\text{DN}}$)
COURSE	($\underline{\text{CN}}$)
TEACH	($\underline{\text{FN, DN}}$, CN)
OFFER	($\underline{\text{CN}}$, DN)

Figure 1.2 Relational Representation for an Entity-Relationship Schema.

† URA constrains a set of attributes to represent at most one set of objects (i.e. entities or relationships). In the relational schema of figure 1.2(ii) the involvement of attributes CN and DN in both the OFFER and TEACH relations implies under URA that TEACH relationships associate FACULTY entities with OFFER relationships, e.g. as shown in figure 3.1, while in the ER schema of figure 1.2(i) relationship-sets TEACH and OFFER are independent.

in order to guarantee BCNF.

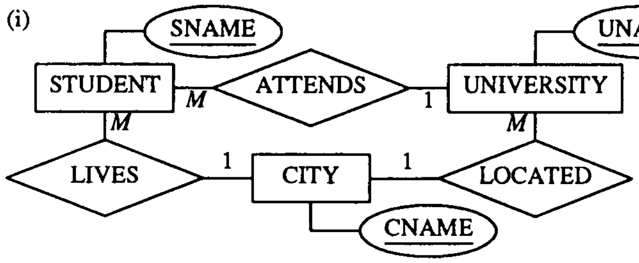
Disregarding the UR assumptions when EER schemas are translated into relational schemas can lead to erroneous analyses. For example, suppose that the relational schema of figure 1.3(ii) is the result of translating the ER schema of figure 1.3(i). Under the UR assumptions, the key dependencies in this schema imply that there are two ways of deriving the city associated with some student and both ways must result in the *same* value. In other words, under these assumptions the relational schema of figure 1.3(ii) carries the implied constraint that a student can attend only a university located in the city where he lives! The same conclusion is reached in [24] for a similar ER schema, but without the mention of the assumptions on which such a conclusion must be based. Furthermore, this conclusion clearly misinterprets the ER schema in which all relationship-sets are considered independent. A way of preventing this problem is discussed in section 5.

1.3 A Modular Approach to Translating EER Schemas into Relational Schemas

Our approach to translating EER schemas into relational schemas is to separate the translation process into four stages: (i) translate EER schemas into canonical relational schemas, (ii) assign names to relational attributes, (iii) normalize relational schemas representing EER schemas, and (iv) merge relation-schemes in relational schemas representing EER schemas. The relationship between these stages is illustrated by the diagram shown in figure 1.4.

First, we examine what constitutes a *correct* relational schema representation for an EER schema. The correctness criteria and a translation of EER schemas into canonical relational schemas that satisfies these criteria are then developed. This translation generates a class of relational schemas that are identical up to a renaming of attribute names, and consisting of relation-schemes, functional dependencies, inclusion dependencies, and null constraints.

(i)



(ii)

STUDENT	(SNAME)
CITY	(CNAME)
UNIVERSITY	(UNAME)
ATTENDS	(SNAME, UNAME)
LOCATED	(UNAME, CNAME)
LIVES	(SNAME, CNAME)

Figure 1.3 Relational Representation for an Entity-Relationship Schema.

Next, we examine alternative *equivalent* relational representations for EER schemas. We explore different ways of obtaining such alternative representations by using various name assignment algorithms, normalization, and relation merging techniques. The examination of these alternative representations requires a framework for analyzing relational dependencies. Such a framework is provided by the *Universal Relation* assumptions [17] which constrain the assignment of names to relational attributes. By applying these assumptions to canonical relational schema translations of EER schemas, we develop the conditions that must be satisfied by attribute names in such schemas.

Various strategies for assigning names to attributes of relational schemas representing EER schemas are examined. As an example of a name assignment algorithm, we present an algorithm that assigns *global* names to attributes so that the overall number of attribute names in the schema is minimal. It is worth noting that applying this algorithm to canonical relational schema translations of EER schemas results in *Universal Relation* schemas; such schemas underly *Universal Relation* that allow users to query databases by mentioning only attribute names [17].

The framework of the assumptions underlying relational normalization allows us to examine the normal form of canonical relational schemas representing EER schemas. For such schemas we specify a normalization procedure into *Boyce-Codd Normal Form* (BCNF), and prove that achieving BCNF does not depend on the structure of the EER schema. This normalization procedure can either precede or follow the assignment of names to relational attributes.

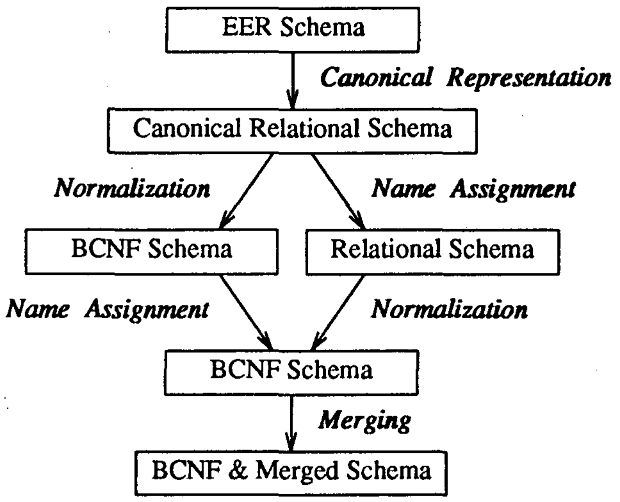


Figure 1.4 A Modular Translation of an EER Schema into a Relational Schema.

It may be desirable to decrease the number of relations in a database by *merging* relations in order to reduce the number of joins that need to be performed, and thus achieve in general a better access performance. While merging usually implies the enforcement of general *null constraints* [15], in certain cases only restricted null constraints, that allow or disallow attributes to have null values, are needed. We present in this paper a simple merging procedure that requires only such restricted null constraints, and that preserves the normal form of relational schemas.

In practice, many relational databases already exist, therefore the reverse problem of identifying EER structures in relational schemas is also important. This problem was addressed in [19], where a modular technique for translating relational schemas into EER schemas is presented.

The rest of the paper is organized as follows. In section 2 we review the graph-theoretic and relational concepts used in this paper. In section 3 we review briefly the version of the EER model considered in this paper, and define the assumptions and semantics underlying its structures and constraints. In section 4 we propose a canonical relational representation for EER schemas and prove its correctness. In section 5 we define equivalence criteria for alternative relational schema representations of EER schemas, and develop the conditions for assigning attribute names in such relational schemas. In section 6 we present a global name assignment algorithm. In section 7 we specify a procedure for transforming canonical relational schema translations of EER schemas into equivalent BCNF schemas. Unlike the standard normalization based on functional dependencies, the normalization procedure presented in this section considers inclusion dependencies as well. A merging procedure is proposed in section 8. Section 9 contains concluding remarks.

2. Preliminary Definitions and Notations

We use in this paper some graph-theoretical concepts. Any textbook on graph theory, such as [8], can provide the necessary reference. We denote by $G = (V, H)$ a directed graph (*digraph*) with set of vertices V and set of edges H, and by $v_i \rightarrow v_j$ a directed edge, h, from vertex v_i to vertex v_j; h is said to be *incident* from v_i to v_j. An undirected *path* from (*start*) vertex v_{i_0} to (*end*) vertex v_{i_m} is a sequence of alternating vertices and edges, $v_{i_0} h_{j_1} v_{i_1} \ldots h_{j_m} v_{i_m}$, such that h_{j_k} is incident from (to) $v_{i_{k-1}}$ to (from) v_{i_k}, $1 \le k \le m$. A *cycle* is a path whose start vertex is also its end vertex. A path is called *simple* if a vertex appears on it at most once. A path (cycle) is said to be directed if all the edges on the path have the same direction and the first edge is incident *from* the start vertex. If there exists a directed path from vertex v_i to vertex v_j then v_j is said to be *reachable* from v_i. The *underlying* (undirected) graph of a digraph results from the digraph by ignoring the edge directions. An undirected graph is called a *tree* iff it has no cycles and any edge added to it forms a cycle.

A digraph $G' = (V', H')$ is a *subgraph* of $G = (V, H)$ if $V' \subseteq V$ and $H' \subseteq H$. The subgraph *induced* by a subset V' of V is denoted $G(V')$, and is defined as follows: the set of vertices is V', and the set of edges is $H' = \{v_i \rightarrow v_j \mid v_i \in V', v_j \in V' \text{ and } v_i \rightarrow v_j \in H\}$. A *(directed) spanning tree* of a (directed) graph G, is a (directed) subgraph of G that contains all the vertices of G, and is a tree. If the edges of G are associated with *lengths* (*weights*) then the *maximum* (*minimum*) spanning tree of G is the spanning tree with the maximum (minimum) sum of edge lengths.

Next, we briefly review below the basic relational concepts used in this paper. Details can be found in any textbook, such as [15], for the basic concepts, and in [5] for inclusion dependencies. We use letters from the beginning of the alphabet to denote attributes and letters from the end of the alphabet to denote sets of attributes. A sequence of attributes (e.g. ABC) denotes the set containing these attributes, and a sequence of sets (e.g. XY) denotes the union of these sets. If X denotes a set of attributes, then $\overline{X}$ denotes a sequence consisting of the attributes of X. We denote by t a tuple, and by $t[W]$ the subtuple of t corresponding to the attribute-set W. A tuple is said to be *total* if it has only non-null values. A null value is denoted ω.

A *relational schema* is a pair (R, Δ) where R is a set of relation-schemes and Δ is a set of dependencies and constraints over R. We consider relational schemas which are associated with functional dependencies, inclusion dependencies, and null constraints. A *relation−scheme* is a named set of attributes, $R_i(X_i)$, where R_i is the relation-scheme name and X_i denotes the associated set of attributes. On the semantic level, every attribute is assigned a *domain*, and a *relation* (value) r_i, is assigned to every relation-scheme, $R_i(X_i)$. A *database state* associated with (R, Δ) is defined as $r = < r_1 ... r_k >$, where r_i is equal to a subset of the cross-product of the domains corresponding to the attributes of $R_i(X_i)$. Two attributes are said to be *compatible* if they are associated with the same domain, and two sequences of distinct attributes $\bar{X}$ and $\bar{Y}$ are said to be *compatible* iff $|X| = |Y| = m$ and for every $1 \le j \le m$, the j'th attribute of $\bar{X}$ and the j'th attribute of $\bar{Y}$ are compatible.

Given two compatible sequences of distinct attributes $\bar{X}$ and $\bar{Y}$, where $|X| = |Y| = m$, we denote by $t[\bar{X}] = t[\bar{Y}]$ the following set of equalities: $\{t[A_1] = t[B_1], \cdots, t[A_j] = t[B_j], \cdots, t[A_m] = t[B_m]\}$, where for every $1 \le j \le m$, A_j is the j'th attribute of $\bar{X}$ and B_j is the j'th attribute of $\bar{Y}$.

Let $R_i(X_i)$ be a relation-scheme associated with relation r_i, and let W be a subset of X_i. The *projection* of r_i on W is denoted $\pi_W(r_i)$, and generates a relation associated with attribute-set W, that is equal to $\{ t[W] \mid t \in r_i \}$. The *total projection* of r_i on W is denoted $\pi{\downarrow}_W(r_i)$, and generates a relation that is equal to the subset of total tuples of $\pi_W(r_i)$.

Let $R_i(X_i)$ be a relation-scheme associated with relation r_i, let W be a subset of X_i, and let $\bar{Y}$ be compatible with $\bar{W}$, where $|Y| = |W| = m$. *Renaming* $\bar{W}$ to $\bar{Y}$ in r_i is denoted *rename* $(r_i; \bar{W} \leftarrow \bar{Y})$, and generates a relation associated with attribute-set $(X_i - W)Y$, that is equal to $\{t' \mid t \in r_i, t'[X_i - W] = t[X_i - W]$, and for every $1 \le j \le m$, $t'[B_j] = t[A_j]$, where B_j is the j'th attribute of $\bar{Y}$ corresponding to the j'th attribute of $\bar{W}, A_j\}$.

Let $R_i(X_i)$ and $R_j(X_j)$ be two relation-schemes associated with relations r_i and r_j, respectively. The *natural join* of r_i and r_j is denoted $r_i \bowtie r_j$, and generates a relation associated with attribute-set $X_i X_j$, that is equal to $\{t \mid t[X_i] \in r_i, t[X_j] \in r_j\}$. Let Y and Z be two disjoint subsets of X_i and X_j, respectively, so that $\bar{Y}$ and $\bar{Z}$ are compatible and $|Y| = |Z| = m$. Let k_i and k_j denote the number of attributes in X_i and X_j, respectively. The *equi−join* of r_i and r_j on $(\bar{Y} = \bar{Z})$ is denoted $r_i \underset{\bar{Y} = \bar{Z}}{\bowtie} r_j$, and is equal to $\{t \mid t[X_i] \in r_i, t[X_j] \in r_j$, and $t[\bar{Y}] = t[\bar{Z}]\}$. The *outer-equi-join* of r_i and r_j on $(\bar{Y} = \bar{Z})$ is denoted $r_i \underset{\bar{Y} = \bar{Z}}{\overset{o}{\bowtie}} r_j$, and is equal to the union of three relations, r_1, r_2, and r_3, where: $r_1 = r_i \underset{\bar{Y} = \bar{Z}}{\bowtie} r_j$, $r_2 = \{t \mid t[X_i] = \omega^{k_i}, t[X_j] \in r_j$, and $\nexists\, t' \in r_i$ s.t. $t'[\bar{Y}] = t[\bar{Z}]\}$, and $r_3 = \{t \mid t[X_i] \in r_i, t[X_j] = \omega^{k_j}$, and $\nexists\, t'' \in r_j$ s.t. $t[\bar{Y}] = t''[\bar{Z}]\}$.

Let $R_i(X_i)$ be a relation-scheme associated with relation r_i. A *functional dependency* over R_i is a statement of the form $R_i\colon Y \rightarrow Z$ where Y and Z are subsets of X_i, and R_i is called a *tag*; $R_i\colon Y \rightarrow Z$ is *satisfied* by r_i iff for every two tuples of r_i, t and t', $t[Y] = t'[Y]$ implies $t[Z] = t'[Z]$.

Let $R_i(X_i)$ and $R_j(X_j)$ be two relation-schemes associated with relations r_i and r_j, respectively. An *inclusion dependency* is a statement of the form $R_i[\bar{Y}] \subseteq R_j[\bar{Z}]$, where Y and Z are subsets of X_i and X_j, respectively, and $\bar{Y}$ and $\bar{Z}$ are compatible; $R_i[\bar{Y}] \subseteq R_j[\bar{Z}]$ is *satisfied* by r_i and r_j iff $\pi\downarrow_Y(r_i) \subseteq \pi\downarrow_Z(r_j)$. The attributes involved in the left-hand side of an inclusion dependency are called *foreign* attributes. The set of inclusion dependencies I over the relation-schemes of R can be represented graphically by the following *inclusion dependency digraph*: $G_I = (V, H)$, where $V = R$, and $R_i \rightarrow R_j \in H$ iff $R_i[\bar{Y}] \subseteq R_j[\bar{Z}] \in I$. A set of inclusion dependencies I is said to be *acyclic* iff G_I is acyclic.

A *null constraint* is a single-tuple restriction on how nulls should appear in a relation [15]. A *null-existence constraint* is a null constraint of the form $R_i : Y \overset{\text{EX}}{\rightarrow} Z$, where $R_i(X_i)$ is a relation-scheme, and Y and Z are subsets of X_i; $R_i : Y \overset{\text{EX}}{\rightarrow} Z$ is *satisfied* by a relation r_i associated with R_i iff for every tuple t of r_i, $t[Y]$ is total only if $t[Z]$ is total. A *nulls-not-allowed* constraint is a null-existence constraint of the form $R_i : \varnothing \overset{\text{EX}}{\rightarrow} Z$, where $R_i(X_i)$ and Z are defined as above, that is is satisfied by a relation r_i associated with R_i iff every subtuple $t[Z]$ of r_i is total.

Let r be a database state associated with schema (R, Δ). Database state r is said to be Δ*–consistent* if it satisfies the dependencies and constraints of Δ. Given Δ, a dependency or constraint δ is said to be *implied* by Δ if every state that satisfies Δ also satisfies δ. The set of dependencies and constraints implied by Δ is called the *closure* of Δ and is denoted Δ^+. Given a set of functional dependencies F and a set of attributes X, the closure of X with respect to F, denoted X^+, consists of the set of attributes $\{Y \mid X \rightarrow Y \in F^+\}$.

A *superkey* associated with R_i is a subset of X_i, K_i, such that $R_i : K_i \rightarrow X_i$ is implied by Δ. Superkey K_i is called a *key* iff there does not exist any proper subset of K_i which is also a superkey of R_i. A relation-scheme can be associated with several *candidate keys* from which one *primary key* is chosen. An attribute which belongs to a candidate key is called a *prime attribute*. A functional dependency of the form $R_i : Y \rightarrow Z$ where $Z \not\subseteq Y$ is called (i) a *key dependency* if Y is a superkey of R_i; (ii) a *partial dependency* if Y is a proper subset of a candidate key of R_i; (iii) a *transitive dependency*, if Y is not the subset of any candidate key of R_i.

Relational *normal forms* (cf. [15]) are properties defined for relation-schemes which guarantee decreased *data redundancy*. For relation-schemes associated with functional dependencies the highest normal form is *Boyce-Codd Normal Form (BCNF)* which requires all functional dependencies to be key dependencies. Partial dependencies of prime attributes are not allowed in BCNF, but are allowed in the (weaker) *Third Normal Form (3NF)*. Transitive dependencies are not allowed in 3NF and BCNF, but are allowed in the *Second Normal Form (2NF)*. Finally, partial dependencies of non-prime attributes are not allowed in 2NF, 3NF, and BCNF.

If $R_i[\bar{Y}] \subseteq R_j[\bar{Z}]$ is an inclusion dependency and Z is the primary key of R_j then $R_i[\bar{Y}] \subseteq R_j[\bar{Z}]$ is said to be *key–based*, and Y is called a *foreign key* of R_i *referencing* R_j. Key-based inclusion dependencies are usually called *referential integrity* constraints [7].

Let r_i and r_j be associated with relation-schemes R_i and R_j, respectively. An inclusion dependency $R_i[\bar{Y}] \subseteq R_j[\bar{Z}]$ must be preserved by insertions, deletions, or updates affecting r_i and r_j. Usually, the insertion of a tuple t into r_i can be performed only if the tuple of r_j referenced by t already exists, and the deletion or update of a tuple t' of r_j can be performed only if there are no tuples in r_i referencing t'; then $R_i[\bar{Y}] \subseteq R_j[\bar{Z}]$ is said to be associated with *restricted* insert, delete, and update rules. We are interested in one additional rule, the *cascades* update-rule, which asserts that if a tuple t' of r_j is changed (updated) into $\tilde{t}'$, then every tuple t of r_i that references t' must be changed into $\tilde{t}$ so that $\tilde{t}[\bar{Y}] = \tilde{t}'[\bar{Z}]$.

3. The Extended Entity–Relationship Model

The basic *Entity-Relationship* (ER) model was originally defined in [6]. We consider in this paper a version of the *extended ER* (EER) model that has two additional abstraction capabilities, generalization and full aggregation. For the sake of brevity, we only sketch the definitions of the basic ER and EER constructs; detailed definitions can be found in reviews such as [10] or [24]. Then we discuss some assumptions concerning the specification of EER constructs and define their semantics.

3.1 Extended Entity–Relationship Constructs

In the basic ER model [6] the *atomic* objects are called *entities*. Associations of entities are represented by *relationships*. We refer commonly to entities and relationships as *objects*. Objects are qualified by *attributes* and are classified into *object–sets*. An attribute is associated with a unique object-set. Attributes take values from underlying domains called *value–sets*. A subset of the attributes associated with an entity-set is specified as the *entity–identifier*. Entity-identifiers are used to distinguish among the instances of an entity-set; when an entity-identifier is not enough to uniquely distinguish among the instances of an entity-set, that entity-set depends for identification (is *ID–dependent*) on other entity-sets and is called a *weak* entity-set. An entity-set associated by a relationship-set is said to have a *role* in that relationship-set. Roles (e.g. HUSBAND, WIFE) are essential in distinguishing the multiple involvements of an entity-set (e.g. PERSON) in a relationship-set (e.g. MARRIAGE, see figure 3.2).

Cardinality is a restriction placed on an entity-set with respect to a relationship-set and can be either *one* or *many*: if R_k is a relationship-set involving entity-set E_i, then a cardinality of *one* for E_i in R_k means that, given any element of the cross-product of all the entity-sets involved in R_k except E_i, there is *at most one* instance of E_i that can be associated by R_k with that element.

The *extended ER* (EER) model considered in this paper has two additional constructs, generalization and full aggregation. *Generalization* is an abstraction mechanism that allows viewing a set of entity-sets (e.g. SECRETARY, FACULTY) as a single *generic* entity-set (e.g. EMPLOYEE). The attributes and associations which are common to the entity-sets that are generalized (e.g. NAME) are then associated only with the generic entity-set. The inverse of generalization is called *specialization*. An entity-set that is neither weak nor the specialization of other entity-sets, is called an *independent* entity-set. We do not distinguish in this paper between different kinds of generalization such as those described in [10].

Generalization defines a transitive relationship between entity-sets. Thus, if entity-set E_i is a direct generalization of entity-set E_j and E_j is a direct generalization of entity-set E_k, then E_i is a transitive generalization of E_k [3]. A specialization entity-set *inherits* the attributes of all its (direct and transitive) generic entity-sets, including the entity-identifier, and these attributes are called its *inherited* attributes. An entity-set that is not specified as the specialization of any other entity-set is called a *generalization–source.*

In the basic ER model the *aggregation* construct takes three forms: (i) the aggregation of a collection (tuple) of attributes into an entity-set; (ii) the aggregation of a collection of attributes and several existing entity-sets into a weak entity-set; and (iii) the aggregation of two or more entity-sets into a relationship-set. The basic ER model falls short of providing the full capability of aggregation by disallowing relationship-sets to be aggregated further. What is needed in order to provide this capability is to allow relationship-sets to associate both entity-sets and relationship-sets, rather than only entity-sets.

3.2 Extended Entity–Relationship Diagram

EER-schemas are expressible in a diagrammatic form called an *EER diagram.* In an EER diagram entity-sets, relationship-sets, and attributes, are represented by rectangle, diamond, and ellipse shaped vertices, respectively, and every vertex is labeled by the name of the object-set or attribute it represents. Since the implied directionality of the ER diagrams of [6] (from vertices representing entity-sets and relationship-sets to vertices representing attributes and entity-sets) is not enough for representing EER schemas, we define an EER diagram as a *directed graph.* The directionality of edges in such a diagram allows the explicit representation not only of the interaction of EER elements, but also of their mutual *existence dependencies.* Thus, in an EER diagram there are directed edges (i) from relationship-sets to the object-sets they associate, labeled by the corresponding cardinality (1 (*one*) or M (*many*)), and possibly by *roles*; (ii) from weak entity-sets to the entity-sets on which they depend for identification, labeled *ID*; (iii) from specialization entity-sets to their direct generic entity-sets, labeled *ISA*; and (iv) from object-sets to their associated attributes.

An example of an EER diagram is shown in figure 3.1, where EMPLOYEE, COURSE, and DEPARTMENT are independent entity-sets, DEPENDENT is a weak entity-set, FACULTY is a specialization of EMPLOYEE, OFFER represents the courses offered by departments, such that each course is offered by at most one department, SUPERVISE represents the assignment of faculty members to supervise courses, such that each faculty member can supervise at most one course, and TEACH represents the assignment of faculty members to teach courses which are offered by departments, such that some course is taught by at most one faculty member. Note that without the full aggregation capability TEACH could not be associated with relationship-set OFFER, and that without the generalization capability, all the attributes and relationship-sets involving only FACULTY could not be represented accurately.

3.3 Assumptions Regarding Extended Entity–Relationship Schemas

For the sake of simplicity, we assume that attributes are allowed to have only atomic (i.e. single, rather than multiple) and non-null values, that a *unique* identifier is specified for every object-set, and that every specialization entity-set has a unique generalization-source. A consequence of the last two restrictions is that no identifiers are defined (locally) for specialization entity-sets, since such entity-sets inherit their identifiers from their generalization-sources.

We disallow two EER structures that are unnecessary and may be confusing: specialization entity-sets which are ID-dependent on other entity-sets, and EER structures involving directed cycles. A specialization entity-set does not need to be specified as ID-dependent on another entity-set because it is sufficient to specify an ID-dependency only with respect to its generalization-source. Thus, an entity-set (e.g. INFANTS) which is the specialization of a weak entity-set (e.g. CHILDREN, ID-dependent on PARENTS) is also weak (thus, there is no need to represent INFANTS as ID-dependent on PARENTS). EER structures represented by EER diagrams that include *directed cycles* have unclear and sometimes problematic semantics. In particular weak entity-sets involved in a directed cycle of an EER diagram imply the obviously erroneous semantics that a weak entity-set depends for identification on itself; and specialization entity-sets involved in a directed cycle of an EER diagram imply that these entity-sets consist of the same entities, that is, are *redundant* [3]. Consequently, we consider in this paper only EER schemas associated with EER diagrams that are acyclic directed graphs.

The last assumption concerns the *names* used for the specification of EER schemas. For a given EER schema, object-sets and roles must have unique *global* names, while the attributes (including

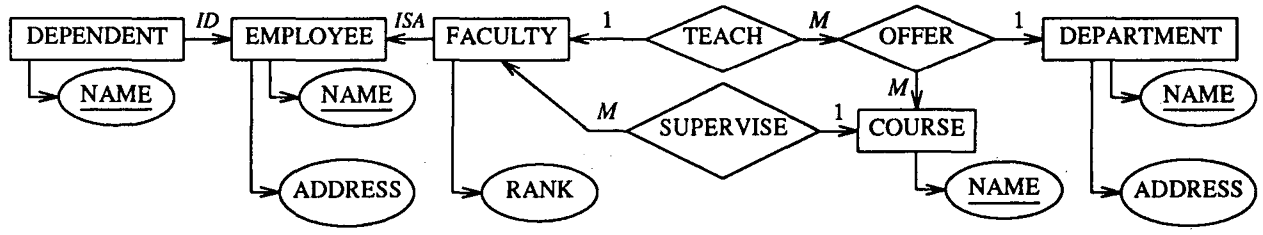


Figure 3.1 An Extended Entity-Relationship Diagram Example (*identifiers are underlined*).

inherited attributes) of an object-set must have unique *local* names among the attributes associated with that object-set. Finally, we assume that multiple involvements of an object-set in a relationship-set are characterized by distinct roles.

3.4 Semantics of Extended Entity–Relationship Constructs and Updates

In this section we define the semantics of the EER constructs and updates. The definition below refers to the basic EER constructs used in this paper, and is used later for a precise characterization of the correctness of relational representations of EER schemas. A discussion of the semantics of related constructs is provided by [10].

If O_i denotes an object-set of an EER schema and x is an object of O_i, then x consists of a collection (tuple) of non-null attribute values. Given an object x we denote by: (i) $\dot{x}$ the collection of values of the attributes associated with (i.e. local to) O_i; (ii) $\ddot{x}$ the collection of values of the identifier attributes associated with O_i; and (iii) $\tilde{x}$ the collection of values of attributes not associated with (i.e. foreign to) O_i. Every object-set O_i in an EER schema can contain objects of one of the following forms:

EER1: If O_i is an independent entity-set then every object x of O_i consists only of $\dot{x}$.

EER2: If O_i is the aggregation of object-sets O_j, $1 \le j \le m$, then every object x of O_i consists of the concatenation of $\dot{x}$ with $x_1, \cdots, x_m$, where x_j is an object of O_j, $1 \le j \le m$ (denoted $x = \dot{x}\, x_1 \cdots x_m$).

EER3: If O_i is a specialization entity-set whose (direct or transitive) generic entity-sets are entity-sets O_j, $0 \le j \le m$, where O_0 is the generalization-source, then every object x of O_i consists of the concatenation of $\dot{x}$ with x_0, $\dot{x}_1$, $\cdots$, $\dot{x}_m$, where x_j is an object of O_j, $0 \le j \le m$ (denoted $x = \dot{x}\, x_0\, \dot{x}_1 \cdots \dot{x}_m$).

Note that given an object x, if x is an independent entity then $\tilde{x} = \varnothing$; if x is an aggregation object then $\tilde{x} = x_1 \cdots x_m$; and if x is a specialization entity then $\tilde{x} = x_0\, \dot{x}_1 \cdots \dot{x}_m$ and $\ddot{x} = \varnothing$. Additionally, the objects of an object-set O_i must satisfy the following two conditions:

EER4: The identification of an object x of O_i does not depend on the values of the non identifier attributes associated with (local to) O_i, that is, for any two distinct objects of O_i, x and x': $\ddot{x}\,\tilde{x} \ne \ddot{x}'\,\tilde{x}'$.

EER5: If O_i is a relationship-set associating object-sets O_j, $1 \le j \le m$, and O_k has cardinality *one* in O_i then for any two distinct objects of O_i, x and x': $x_1 \cdots x_{k-1} x_{k+1} x_m \ne x'_1 \cdots x'_{k-1} x'_{k+1} x'_m$.

Definition 3.1. A collection of objects associated with an EER schema *EERS* is said to constitute a *consistent* state of *EERS* if the objects satisfy conditions **EER1** through **EER5** above. ■

In an EER environment an elementary update consists of modifying an attribute value, removing an object from an object-set, or adding a new object to an object-set. EER schemas imply certain existence dependencies that must be satisfied by updates. In order to satisfy the existence dependencies, an object x that is existence dependent on another object y cannot be added before y is added, and y cannot be removed before x is removed. Intuitively, y *blocks* the addition of x, and x *blocks* the removal of y. Attribute modifications, however, are local to the objects, therefore existence dependencies between objects do not affect such modifications. Consequently, the update of any local attribute value in an object x is unrestricted, provided it satisfies conditions **EER4** and **EER5**.

3.5 Navigational Semantics for Extended Entity–Relationship Schemas

We conclude this section by discussing the *navigational* semantics of an EER schema. Clarifying the navigational semantics of EER schemas is essential for investigating the assumptions underlying the definitions of equivalence between relational representations of EER schemas (see section 5).

Let G_{ER} be an EER diagram. A *navigation–path* in G_{ER} consists of a simple path between two vertices of the underlying (undirected) graph of G_{ER}. In the EER diagram of figure 3.1, for example, there are two navigation-paths between FACULTY and DEPARTMENT: via TEACH and OFFER, and via SUPERVISE, COURSE, and OFFER. A navigation-path between two object-set vertices, O_i and O_j, represents a meaningful association between the objects of O_i and O_j [13]. Navigation-paths representing associations that have distinct meaning in the basic ER model have been examined in [14]. The two navigation-paths above, for example, represent associations with (clearly) different meanings. In an EER schema, however, distinct navigation-paths that lie entirely within the same generalization structure represent equivalent (i.e. having the same meaning) associations; such navigation-paths have the same start and end vertices in the underlying EER diagram. In the EER diagram of figure 3.2, for instance, the paths from PERSON to TEACHING ASSISTANT via STUDENT and GRADUATE STUDENT, respectively via EMPLOYEE and FACULTY, are equivalent because they both refer to the set of PERSON entities that are also TEACHING ASSISTANTs.

The *Distinct Meaning Path Assumption* (DMPA) states that distinct navigation-paths between two object-sets that differ in at least one non-ISA labeled edge, represent associations with distinct meanings. Note that unlike the *Unique Path Assumption* of [13], DMPA does not restrict the EER schemas, but rather states that it is incorrect to assume that certain distinct navigation-paths represent the same association. In particular, multiple relationship-sets associating common object-sets have distinct meanings. In the EER diagram of figure 3.2, for instance, there are three distinct paths from PERSON to COURSE: (1) via STUDENT, GRADUATE STUDENT, TEACHING ASSISTANT, and ASSIGNED; (2) via EMPLOYEE, FACULTY, TEACHING ASSISTANT, and ASSIGNED; and (3) via EMPLOYEE, FACULTY, and ASSIGNED. The first two paths are equivalent because they represent associations that have the same meaning, namely PERSON assigned as TEACHING ASSISTANT to a COURSE; the meaning of the association represented by the third path is different, namely PERSON assigned as FACULTY to a COURSE.

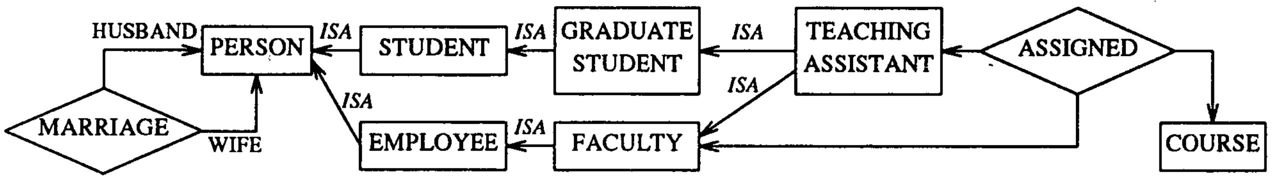


Figure 3.2 An Extended Entity-Relationship Diagram with Multiple Navigation-Paths.

4. Translating EER Schemas into Canonical Relational Schemas

Translating EER schemas into relational schemas involves representing the structural semantics of EER schemas using relational constructs and assigning names to relational attributes. We define below correctness criteria for relational representations of EER schemas. Then we propose a translation called ***Crep*** (*Canonical representation*) that satisfies these criteria. ***Crep*** generates a class of relational schemas that are identical up to a renaming of attributes. Since ***Crep*** is not dependent on a particular attribute naming mechanism we refer to the resulting relational schema as *canonical.* Using canonical schemas as a basis, we explore in the next section equivalent relational representations for EER schemas.

4.1 Correct Relational Representations for EER Schemas

We characterize below correct relational representations for EER schemas. Informally, a relational schema *RS* is said to represent correctly an EER schema if EER objects can be represented in a database state associated with *RS* without loss of information, and if a database state associated with *RS* can contain only data on EER objects. We are interested only in relational representations that preserve the EER attribute values; this requirement is ensured by condition (4) below.

Definition 4.1. Let *EERS* be an EER schema and let $RS = (R, \Delta)$ be a relational schema. *RS* is said to *represent correctly EERS* if there exist *total* functions ρ and ρ' such that:

1. ρ maps consistent states of *EERS* into consistent states of *RS*;
2. ρ' maps consistent states of *RS* into consistent states of *EERS*;
3. the composition of ρ followed by ρ' is the identity on the set of all consistent states of *EERS*; and the composition of ρ' followed by ρ is the identity on the set of all consistent states of *RS*.
4. For any consistent state s of *EERS*, ρ preserves the attribute values of s [†]; similarly, for any consistent state r of *RS*, ρ' preserves the attribute values of r. ■

While the correctness criteria above must be satisfied by relational schemas representing EER schemas, there is another very desirable property for such schemas. This property regards update translations and requires EER updates (i.e. object insertions, object deletions, and attribute modifications) to be correctly translatable into relational tuple updates (i.e. tuple insertions, tuple deletions, and attribute modifications).

[†] A state mapping ρ is said to preserve the attribute values of a state s iff the values of $\rho(s)$ are included in s.

Definition 4.2. Let *EERS* be an EER schema and let *RS* be a relational schema representing correctly *EERS*. *RS* is said to allow *faithful* translations of EER updates iff there exists a mapping ψ of EER object updates into relational tuple updates, such that given an EER object update *upd*, $\psi\,(upd)$ consists of tuple updates and $r' = \psi\,(upd)\,(r)$, where r is the database state associated with *RS* representing the objects associated with *EERS* before *upd*, and r' is the database state associated with *RS* representing the objects associated with *EERS* after *upd*. ■

4.2 A Canonical Relational Representation for EER Schemas

We define ***Crep*** in the following three subsections, where the independent entity-set, aggregation, and generalization constructs are represented using relational constructs. While it is sufficient to represent these constructs using relation-schemes and inclusion dependencies, we also use functional dependencies for representing the identifiers and relationship cardinalities in EER schemas, and nulls-not-allowed constraints for representing restrictions on the allowed (non-null) values for EER attributes. In order to avoid a specific name assignment for relational attributes, we use in our examples (internal) *symbols* for referring to attributes, such as A_{4_2} (representing the second attribute of relation-scheme R_4). Note that all EER attributes of the object-sets that are involved in an aggregation or specialization have correspondents in the relation-scheme representing the aggregate or specialization object-set, even though some of these attributes may be redundant. This is caused by the lack of any information on attribute redundancy when ***Crep*** is applied. In a later stage, redundant attributes can be removed using a normalization process.

4.2.1 Independent Entity–Sets

EER value-sets are represented straightforwardly by relational domains. The relational model has an aggregation mechanism for aggregating attributes to form relation-schemes, similar to the aggregation of EER attributes that defines independent entity-sets. Thus, an independent entity-set, E_i, is straightforwardly represented by a relation-scheme, $R_i\ (X_i)$, such that the following condition holds:

E1: X_i is in a one-to-one correspondence with the EER attributes of E_i, and the domain associated with an attribute A of X_i represents the value-set of the E_i attribute corresponding to A.

Relation-scheme R_i is associated with functional dependency $R_i : Z_i \rightarrow (X_i - Z_i)$, where Z_i is the subset of X_i corresponding to the identifier of E_i, and with nulls-not-allowed constraint $R_i : \varnothing \overset{\text{EX}}{\rightarrow} X_i$.

For example, the three independent entity-sets of the EER schema of figure 3.1 are represented by relation-schemes R_1, R_2, and R_3 of the relational schema of figure 4.1.

4.2.2 Object–Set Aggregation

Unlike the aggregation of EER attributes, the object-set aggregation has no analogous relational construct. Let object-set O_i be the aggregation of object-sets O_j, $1 \leq j \leq m$, and let each object-set O_j be represented by relation-scheme $R_j\,(Y_j)$, $1 \leq j \leq m$, respectively. Then object-set O_i is represented by relation-scheme $R_i\,(X_i)$, together with inclusion dependencies $R_i[\bar{X}_{i_j}] \subseteq R_j[\bar{Y}_j]$, $1 \leq j \leq m$, where X_i is the union of two disjoint sets of attributes, X'_i and X''_i, defined below:

A1: X'_i is in a one-to-one correspondence with the (local) EER attributes of O_i, where the correspondence is specified as in condition **E1** above;

A2: X''_i is a set of foreign attributes, such that $X''_i = \bigcup_{j=1}^{m} X_{i_j}$, where every attribute-set X_{i_j}, $1 \leq j \leq m$, is defined so that $\bar{X}_{i_j}$ is compatible with $\bar{Y}_j$, and so that attribute-sets $\bar{X}_{i_j}$, $1 \leq j \leq m$, are pairwise disjoint.

Relation-scheme R_i is associated with functional dependency $R_i : Z_i X''_i \rightarrow (X_i - Z_i\, X''_i)$, where Z_i is the subset of X_i corresponding to the identifier of O_i, and with nulls-not-allowed constraint $R_i : \varnothing \overset{\text{EX}}{\rightarrow} X_i$. If O_i is a relationship-set, then for every object-set O_j that is involved in O_i with cardinality *one*, R_i is

Relation : Object-Set	**Attribute : ER Attribute**		**Foreign Attribute : Attribute**			
$R_1\ (X_1)$: EMPLOYEE	A_{1_1} : NAME	A_{1_2} : ADDRESS				
$R_2\ (X_2)$: DEPARTMENT	A_{2_1} : NAME	A_{2_2} : ADDRESS				
$R_3\ (X_3)$: COURSE	A_{3_1} : NAME					
$R_4\ (X_4)$: FACULTY	A_{4_1} : RANK		$A_{4_2} : A_{1_1}$	$A_{4_3} : A_{1_2}$		
$R_5\ (X_5)$: DEPENDENT	A_{5_1} : NAME		$A_{5_2} : A_{1_1}$	$A_{5_3} : A_{1_2}$		
$R_6\ (X_6)$: OFFER			$A_{6_1} : A_{2_1}$	$A_{6_2} : A_{2_2}$	$A_{6_3} : A_{3_1}$	
$R_7\ (X_7)$: TEACH			$A_{7_1} : A_{4_1}$	$A_{7_2} : A_{4_2}$	$A_{7_3} : A_{4_3}$	
			$A_{7_4} : A_{6_1}$	$A_{7_5} : A_{6_2}$	$A_{7_6} : A_{6_3}$	
$R_8\ (X_8)$: SUPERVISE			$A_{8_1} : A_{4_1}$	$A_{8_2} : A_{4_2}$	$A_{8_3} : A_{4_3}$	$A_{8_4} : A_{3_1}$

Functional Dependencies	**Inclusion Dependencies**		**Null Constraints**
$R_1 : A_{1_1} \rightarrow A_{1_2}$	$R_4[A_{4_2} A_{4_3}]$	$\subseteq R_1[A_{1_1} A_{1_2}]$	$R_1 : \varnothing \overset{\text{EX}}{\rightarrow} X_1$
$R_2 : A_{2_1} \rightarrow A_{2_2}$	$R_5[A_{5_2} A_{5_3}]$	$\subseteq R_1[A_{1_1} A_{1_2}]$	$R_2 : \varnothing \overset{\text{EX}}{\rightarrow} X_2$
$R_3 : A_{3_1} \rightarrow \varnothing$	$R_6[A_{6_1} A_{6_2}]$	$\subseteq R_2[A_{2_1} A_{2_1}]$	$R_3 : \varnothing \overset{\text{EX}}{\rightarrow} X_3$
$R_4 : A_{4_2} A_{4_3} \rightarrow A_{4_1}$	$R_6[A_{6_3}]$	$\subseteq R_3[A_{3_1}]$	$R_4 : \varnothing \overset{\text{EX}}{\rightarrow} X_4$
$R_5 : A_{5_1} A_{5_2} A_{5_3} \rightarrow \varnothing$	$R_7[A_{7_1} A_{7_2} A_{7_3}]$	$\subseteq R_4[A_{4_1} A_{4_2} A_{4_3}]$	$R_5 : \varnothing \overset{\text{EX}}{\rightarrow} X_5$
$R_6 : A_{6_3} \rightarrow A_{6_1} A_{6_2}$	$R_7[A_{7_4} A_{7_5} A_{7_6}]$	$\subseteq R_6[A_{6_1} A_{6_2} A_{6_3}]$	$R_6 : \varnothing \overset{\text{EX}}{\rightarrow} X_6$
$R_7 : A_{7_4} A_{7_5} A_{7_6} \rightarrow A_{7_1} A_{7_2} A_{7_3}$	$R_8[A_{8_1} A_{8_2} A_{8_3}]$	$\subseteq R_4[A_{4_1} A_{4_2} A_{4_3}]$	$R_7 : \varnothing \overset{\text{EX}}{\rightarrow} X_7$
$R_8 : A_{8_1} A_{8_2} A_{8_3} \rightarrow A_{8_4}$	$R_8[A_{8_4}]$	$\subseteq R_3[A_{3_1}]$	$R_8 : \varnothing \overset{\text{EX}}{\rightarrow} X_8$

Figure 4.1 Canonical Relational Schema Translation of the EER Schema of Figure 3.1.

associated with the additional functional dependency $R_i : (X''_i - X_{i_j}) \rightarrow X_{i_j}$, where X_{i_j} is defined as in **A2** above.

For example, the four aggregations of the EER schema of figure 3.1 are represented by relation-schemes R_5, R_6, R_7, and R_8 of the relational schema of figure 4.1. Weak entity-set DEPENDENT, for instance, is represented by relation-scheme R_5 (X_5), where $X'_5 = A_{5_1}$ and $X''_5 = A_{5_2}A_{5_3}$; relationship-set OFFER is represented by relation-scheme R_6 (X_6), where X'_6 is empty and X''_6 consists of two disjoint subsets, $A_{6_1}A_{6_2}$ and A_{6_3}.

4.2.3 Generalization

Generalization also lacks an analogous relational construct. Let entity-set E_i be the direct specialization of entity-sets E_j, $1 \le j \le m$, and let E_0 be the generalization-source of E_i †. Let E_0 be represented by relation-scheme $R_0(Y_0)$ and each entity-set E_j be represented by relation-scheme $R_j(Y_j)$, $1 \le j \le m$, respectively. Then entity-set E_i is represented by relation-scheme R_i (X_i) together with inclusion dependencies $R_i[\overline{X}_{i_j}] \subseteq R_j[\overline{Y}_j]$, $1 \le j \le m$, where X_i is the union of two disjoint sets of attributes, X'_i and X''_i, defined below:

G1: X'_i is in a one-to-one correspondence with the set of all (local and inherited†) attributes of E_i, except the attributes inherited from E_0, where the correspondence is specified as as in condition **E1** above;

G2: X''_i is a set of foreign attributes defined so that $\overline{X}''_i$ is compatible with $\overline{Y}_0$;

G3: every set X_{i_j} of foreign attributes, $1 \le j \le m$, is defined so that X_{i_j} includes X''_i, and $\overline{X}_{i_j}$ is compatible with $\overline{Y}_j$, where corresponding attributes of $\overline{X}_{i_j}$ and $\overline{Y}_j$ result from the mapping of the same EER attribute or the same attribute of Y_0.

Relation-scheme R_i is associated with functional dependency $R_i : X''_i \rightarrow (X_i - X''_i)$, and with nulls-not-allowed constraint $R_i : \varnothing \xrightarrow{EX} X_i$.

For example, entity-set FACULTY of the EER schema of figure 3.1 is represented by relation-scheme $R_4(X_4)$ of the relational schema of figure 4.1, where $X'_4 = A_{4_1}$ and $X''_4 = A_{4_2}A_{4_3}$.

† Recall that specialization entity-sets have unique generalization-sources, and that the set of inherited attributes of a specialization entity-set consists of the attributes associated with all its generic entity-sets.

4.2.4 State Mappings

Crep is associated with two *state mappings*, ρ and ρ', where ρ maps a state of *EERS* into a state of *RS*, and ρ' maps a state of *RS* into a state of *EERS*, as follows:

ρ : For every object-set O_i that corresponds to relation-scheme $R_i(X_i)$, ρ maps every object x of O_i into a tuple t of a relation associated with R_i, such that every value of an EER attribute A in x is mapped into a t value of the relational attribute corresponding to A.

ρ' : For every relation-scheme $R_i(X_i)$ that corresponds to object-set O_i, ρ' maps every tuple t of the relation associated with R_i into an object x of O_i, such that every t value of a relational attribute B is mapped into an x value of the EER attribute corresponding to B.

To summarize, ***Crep*** generates relational schemas of the form $(R, F \cup I \cup N)$, where F denotes a set of functional dependencies, I denotes a set of inclusion dependencies, and N denotes a set of nulls-not-allowed constraints; every inclusion dependency of I is associated with a *restricted* insert-rule, a *restricted* delete-rule, and a *cascades* update-rule. The relational-schemes of R represent EER object-sets, the inclusion dependencies of I represent the interaction of EER object-sets, the functional dependencies of F represent object identifiers and relationship cardinalities, and the nulls-not-allowed constraints represent restrictions on the allowed values for EER attributes.

We prove below that ***Crep*** generates correct relational schema representations for EER schemas, that allow *faithful* translations of EER updates into relational tuple updates.

Proposition 4.1. Let $RS = (R, F \cup I \cup N)$ be the canonical relational schema generated by ***Crep*** for an EER schema, *EERS*. Then (i) *RS* represents correctly *EERS*, and (ii) *RS* allows faithful translations of EER updates.

Proof Sketch : We use below a claim whose proof follows directly from the specification of ***Crep***.

Claim. Let G_I be the inclusion dependency digraph associated with *RS*, and let G_{ER} be the EER diagram underlying *EERS*. Then G_I and the subgraph of G_{ER} induced by the object-set vertices are isomorphic.

(i) We prove that ρ and ρ' satisfy the conditions of definition 4.1. First note that every tuple of a relation associated with a relation-scheme $R_i(X_i)$ of R is of the form $t = \dot{t}\,\tilde{t}$, where $\dot{t}$ and $\tilde{t}$ contain the values of non-foreign, resp. foreign, attributes of X_i. It can be easily verified that ρ maps an object $x = \dot{x}\,\tilde{x}$ of an object-set O_i into a tuple $t = \dot{t}\,\tilde{t}$ of a relation associated with relation-scheme $R_i(X_i)$ corresponding to O_i such that x and t are in a one-to-one correspondence of identical attribute values (ρ preserves the attribute values), and $\dot{t}$ (resp. $\tilde{t}$) contains the relational attribute values corresponding to the EER attribute values of $\dot{x}$ (resp. $\tilde{x}$). Similarly, ρ' maps a tuple t of a relation associated with a relation-scheme $R_i(X_i)$ into an object x of an object-set O_i corresponding to R_i such that t and x are in a one-to-one correspondence of identical attribute values (ρ' preserves the attribute values), and $\dot{x}$ (resp. $\tilde{x}$) contain the EER attribute values corresponding to the relational attribute values of $\dot{t}$ (resp. $\tilde{t}$).

Condition 1 *of Definition* 4.1: The proof is by induction on the number of objects mapped by ρ. Initially the state associated with *RS* is empty. The inductive hypothesis is that before an object is mapped by ρ, the state associated with *RS* satisfies $(F \cup I \cup N)$. The objects are mapped by ρ in the (partial) order defined by G_{ER}. It is straightforward to verify that the relational state resulting after every object mapping also satisfies $(F \cup I \cup N)$.

Condition 2 *of Definition* 4.1: The proof is by induction on the number of tuples mapped by ρ'. Initially no objects are associated with *EERS*. The inductive hypothesis is that before a tuple is mapped by ρ', the objects associated with *EERS* satisfy conditions **EER1** through **EER5**. Following the claim above, G_I is an acyclic directed graph. The tuples are mapped by ρ' in the (partial) order defined by G_I. It is straightforward to verify that every tuple is mapped by ρ' into an object that satisfies conditions **EER1** through **EER5**.

Condition 3 *of Definition* 4.1: We must show that (a) ρ' applied on a state generated by ρ from a collection of objects associated with *EERS* returns the same collection of objects, and that (b) $\rho(\rho'(r)) = r$. Since ρ (resp. ρ') maps an object x (resp. tuple t) into a tuple t (resp. object x), such that x and t are in a one-to-one correspondence of identical attribute values, it follows that ρ (resp. ρ') maps distinct objects (resp. tuples) into distinct tuples (resp. objects). Consequently, no objects (resp. tuples) are 'lost' by ρ (resp. ρ').

Condition 4 *of Definition* 4.1: is satisfied by ρ and ρ' by definition.

(ii) Let ψ be defined as follows: ψ translates (a) an insertion of an object x into the insertion of tuple $\rho(x)$; (b) a deletion of an object x into the deletion of tuple $\rho(x)$; (c) an attribute modification affecting local attributes $A_1, \ldots, A_k$ of an object x into the attribute modification affecting the attributes of tuple $\rho(x)$ corresponding to $A_1, \ldots, A_k$. It can be verified that when an object insertion or deletion can be carried out (i.e. the existence dependencies are satisfied by the update), then the corresponding relational tuple insertion or deletion can be also carried out, and that condition $r' = \psi(upd)(r)$ of definition 4.2 is satisfied. Conversely, when an object insertion or deletion cannot be carried out then the corresponding relational tuple insertion or deletion also cannot be carried out because of the *restricted* insert and delete rules associated with the inclusion dependencies; then condition $r' = \psi\,(upd)\,(r)$ is trivially satisfied. Unlike object insertions and deletions, attribute modifications always can be carried out, and the *cascades* update-rules associated with the inclusion dependencies ensure that condition $r' = \psi\,(upd)\,(r)$ is satisfied. ■

5. Equivalent Relational Representations for EER Schemas

In this section we define the criteria characterizing alternative relational representations of EER schemas. In the next sections we explore different ways of obtaining such alternative representations, namely using various name assignment algorithms, via normalization processes, and using relation merging techniques. The examination of these alternative representations requires a framework for analyzing relational dependencies. Such a framework is provided by the *Universal Relation* assumptions that constrain the assignment of names to relational attributes. We apply these assumptions to relational schemas generated by ***Crep*** and thus develop the conditions that must be satisfied by the attribute names in a relational schema representing an EER schema.

5.1 Equivalent Information–Capacity

It is well known in database design that the same data can be structured in different ways, that is, represented by different schemas provided these schemas have *equivalent information-capacities* [9]. Since ***Crep*** generates provably correct (canonical) relational representations for EER schemas, any relational representation of an EER schema should have *equivalent* information-capacity with the canonical representation of that EER schema. As already mentioned in section 4, we are interested only in relational representations that preserve the EER attribute values. The information-capacity equivalence of two relational schemas defined below follows [9], and consists of conditions analogous to the conditions of definition 4.1.

Definition 5.1. Let $RS = (R, \Delta)$ and $RS' = (R', \Delta')$ be two relational schemas. RS' and RS are said to have *equivalent information-capacity* iff there exist *total* functions ϕ and ϕ' such that:

1. ϕ maps consistent database states of RS into consistent database states of RS' ;
2. ϕ' maps consistent database states of RS' into consistent database states of RS;
3. the composition of ϕ followed by ϕ' is the identity on the set of all consistent database states of RS; the composition of ϕ' followed by ϕ is the identity on the set of all consistent database states of RS;
4. For any database state r of RS, ϕ preserves the data values of r †; similarly, for any database state r' of RS' , ϕ' preserves the data values of r' . ■

† A database state mapping ϕ is said to preserve the data values of a database state r iff the values of $\phi(r)$ are included in r.

Informally, a schema *RS* has equivalent information-capacity with another schema *RS'* if *RS'* can be associated with the same number of database states as *RS*; that is, not only every legal database state associated with *RS* must be exactly reconstructed from its mapping into a database state of *RS'*, but every database state associated with *RS'* must be mappable into a database state of *RS*. Condition (4) above ensures that attribute values are preserved by the database state mappings (in [9] such mappings are called *internal*), thus reflecting our requirement for relational schemas representing EER schemas. The following definition extends definition 4.1 regarding the correctness of relational schema representations for EER schemas.

Definition 5.2. If a relational schema *RS represent correctly* an EER schema *EERS*, then any relational schema *RS'* that has an *equivalent* information-capacity with *RS*, also *represents correctly EERS*. ■

5.2 Conditions for Attribute Names

Crep generates a class of relational schemas that are identical up to a renaming of attributes. In this section we examine what names can be assigned to attributes in schemas generated by ***Crep***, using the *Weak Instance Model* approach to the relational model [18].

In the framework provided by the *Weak Instance Model* data dependencies can be specified and compared over the entire database rather than single relations; in particular, functional dependencies can be specified in such a framework over a universal set of attributes rather than over single relation-schemes, so that they do not need to be associated with a relation-scheme tag, that is, they are *untagged*. Given a database state *r* associated with schema (R, Δ), its *weak instance, u*, is a relation associated with the universal [†] set of attributes, such that *u* satisfies Δ and every relation r_i of *r* is a subset of $\pi\downarrow_{X_i}(u)$, where r_i is associated with relation-scheme $R_i(X_i)$. The *Weak Instance Model* is based on several assumptions. We briefly review below these assumptions and examine their impact on the assignment of names to attributes in relational schemas generated by ***Crep***; these assumptions are surveyed in detail in [17].

An attribute *A* in a relational schema generated by ***Crep*** represents an EER attribute to which *A* corresponds directly, or which is represented by the relational attribute to which *A* corresponds as a foreign attribute. The *Universal-Relation Scheme Assumption* requires every attribute to represent a property of the same class of objects in every relation-scheme in which it appears. Since two attributes are

[†] The universal set of attributes consists of the union of the attribute-sets associated with the relation-schemes of *R*.

considered to be identical iff they are assigned the same *global* name, the attributes in a relational schema generated by ***Crep*** must be assigned names that satisfy the following condition:

At1: Attributes representing distinct EER attributes must be assigned distinct global names.

In the relational schema of figure 4.1, for example, the attributes representing EER attribute NAME of entity-set DEPARTMENT (e.g. A_{2_1}, A_{6_1}) must be assigned different names than the attributes that represent EER attribute NAME of entity-set EMPLOYEE (e.g. A_{1_1}, A_{4_2}, A_{7_2}).

The *Unique Role Assumption* (URA) requires every subset W of the universal set of attributes to represent *at most one basic association* among the attributes of W. The first aspect of URA refers to W as part of attribute-sets of relation-schemes: if W appears in more than one relation-scheme then W must represent the same class of objects in all the relation-schemes in which it appears; in particular, the attribute-set of a relation-scheme must represent a unique class of objects. Consequently, the attributes in a relational schema RS generated by ***Crep*** must be assigned names that satisfy the following conditions:

At2: For any two relation-schemes $R_i(X_i)$ and $R_j(X_j)$ of RS, (i) if $X_i \cap X_j$ is not empty then there exists a relation-scheme $R_k(X_k)$ (that corresponds to an EER object-set), such that $X_i \cap X_j = X_k$; and (ii) if $X_i \subseteq X_j$, then there exists a relation-scheme $R_k(X_k)$, $k \neq j$, such that $X_i \subseteq X_k \subseteq X_j$ and $R_j(X_j)$ corresponds to an EER object-set that is either the aggregation or specialization of the EER object-set corresponding to $R_k(X_k)$.

Note that conditions **At2**(i) and **At2**(ii) above correspond to the *association integrity* and *containment condition* of [17], respectively. Consider, for example, the relational schema of figure 4.1; if attributes A_{8_1}, A_{8_2}, A_{8_3}, and A_{8_4} are assigned the same names as attributes A_{7_1}, A_{7_2}, A_{7_3}, and A_{7_6}, respectively, then condition **At2** is not satisfied because under this name assignment attribute-set X_8 is included in X_7 although the corresponding relationship-sets are independent. Similarly, the relational schema of figure 1.2(ii) (section 1) does not satisfy condition **At2** because the attribute-set of relation-scheme OFFER is

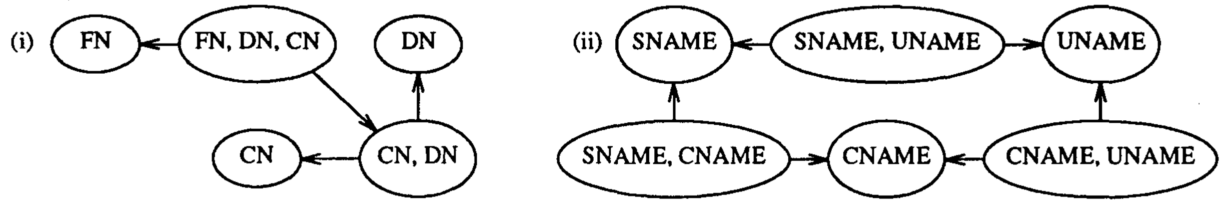


Figure 5.1 The Attribute Digraphs for the Relational Schemas of Figures 1.2(ii) and 1.3(ii).

included in the attribute-set of relation-scheme TEACH (see figure 5.1(i)), although the corresponding relationship-sets are independent; condition **At2** is satisfied, if, for instance, attribute CN of OFFER is assigned another name (i.e. is renamed), such as O.CN.

In general, the *basic association* represented by an attribute-set W refers to the projection on W of the natural-join of a set of relations. The corresponding join expression is based on a *join-path* which consists of a sequence of relation-schemes of the form $R_{i_1}(X_{i_1}),..., R_{i_m}(X_{i_m})$, where W is included in the union of attribute-sets X_{i_j}, $1 \le j \le m$, every relation-scheme appears only once in the sequence, and the intersection of the attribute-sets associated with any two neighbor relation-schemes in the sequence, R_{i_j} and $R_{i_{j+1}}$, $1 \le j < m$, is nonempty. As noted in [17], if multiple join-paths can be associated with a given set of attributes then URA implies the additional *One-Flavor Assumption* (OFA). OFA requires all the join-paths that can be associated with some attribute-set to represent the same *flavor* of relationship (see [17], [18]). Let $G_A = (V, H)$ be the *attribute digraph* associated with a set R of relation-schemes, where $V = R$, and $R_i \rightarrow R_j \in H$ iff $X_j \subseteq X_i$, and $\nexists\, R_k\,(X_k) \in R$, $k \ne i, j$, such that $X_j \subsetneq X_k \subsetneq X_i$. In order to comply with OFA the attributes in a relational schema RS generated by ***Crep*** must be assigned names that satisfy the following condition:

At3: The attribute digraph associated with the relation-schemes of RS is allowed to contain undirected cycles only of the following form: all the vertices on such a cycle correspond to relation-schemes that represent entity-sets belonging to the same generalization hierarchy.

For example, the relational schema of figure 1.3(ii) (section 1) does not satisfy condition **At3** because the associated attribute digraph contains an undirected cycle that is not allowed by **At3** (see figure 5.1(ii)); condition **At3** is satisfied, if, for instance, attribute UNAME in relation scheme ATTENDS is assigned another name, such as VNAME.

Proposition 5.1. Let $RS = (R,\ F \cup I \cup N)$ be a relational schema generated by ***Crep***, and let the names assigned to the attributes of R satisfy conditions **At1** and **At2**. Then the names assigned to the attributes of R satisfy OFA only if condition **At3** is satisfied.

Proof : We use the following notations: G_A and G_I denote the attribute digraph and inclusion dependency digraph associated with RS, respectively; G_{ER} denotes the EER diagram associated with the EER schema corresponding to RS, and G'_{ER} denotes the subgraph of G_{ER} induced by the object-set vertices. The proof is based on the following claims.

Claim 1. Digraph G_A is a subgraph of digraph G_I.

Proof : From the specification of *Crep* and condition At2 it follows immediately that if $R_i(X_i)$ and $R_j(X_j)$ are two relation-schemes of R such that $X_j \subseteq X_i$, and if $\nexists\, R_k(X_k) \in R$ such that $X_j \subseteq X_k \subseteq X_i$, then $R_i[\bar{Y}] \subseteq R_j[\bar{X}_j] \in I$.

Claim 2. Digraphs G_I and G'_{ER} are isomorphic (see proposition 4.1).

We now show that every join-path has an *equivalent* join-path represented by a simple path in G_A, where the equivalence means that join expressions based on equivalent join-paths evaluate to the same relation.

Claim 3. Every join-path over R is represented by a simple path in G_A.

Proof : Let $R_{i_1}(X_{i_1}),\ldots,\ R_{i_m}(X_{i_m})$ be a join-path and R_{i_j}, $R_{i_{j+1}}$, $1 \le j < m$, be a pair of adjacent relation-schemes in this join-path. Then $X_{i_j} \cap X_{i_{j+1}} \neq \emptyset$, and from condition At2 it follows that there exists a relation-scheme $R_k(X_k)$ such that either $X_{i_j} \subseteq X_k \subseteq X_{i_{j+1}}$ or $(X_{i_j} \cap X_{i_{j+1}}) = X_k$. For $R_{i_j} \neq R_k \neq R_{i_{j+1}}$ we must show that $R_{i_j} \bowtie R_{i_{j+1}} \equiv R_{i_j} \bowtie R_k \bowtie R_{i_{j+1}}$ and this follows from claim 1 above. The equivalence preserving addition of relation-schemes to the join-path described above can be repeated until the join-path corresponds to a simple path in G_A.

The elements of the association represented by a navigation-path of G_{ER} are called *traversals* [14]. By claims 1 and 2, every path of G_A is isomorphic to a navigation-path of G_{ER}. Let α be a navigation-path in G_{ER}. Following the definition of traversals of [14], it can be shown that the traversals corresponding to α are tuples in the relation generated by the join corresponding to the join-path of G_A that is isomorphic to α. By claim 3, every join-path is represented by some path of G_A, which, in turn, is isomorphic to some navigation-path of G_{ER}. Consequently, the *Distinct Meaning Path Assumption* of section 3 implies that for a given set of attributes distinct join-paths represent different relationship *flavors*, with one exception: when the join-paths correspond to navigation-paths that differ only in vertices that represent entity-sets of the same generalization hierarchy. Consequently, in order to satisfy OFA G_A must satisfy condition At3. ■

6. ATTRIBUTE NAME ASSIGNMENT ALGORITHMS

In this section we demonstrate that the translation of EER schemas into relational schemas can be coupled with various name assignment algorithms. Our approach is to generate first relational attributes that are represented by (internal) symbolic names, as done in translation ***Crep*** defined in section 4. Subsequently, a name assignment algorithm can be applied in order to replace these symbolic names by (semantically) meaningful names. Provided that this algorithm complies with the assumptions underlying relational normalization, the relational schema can then be normalized. Conversely, normalization can be applied directly on the canonical schema, and then a name assignment algorithm can be applied on the normalized schema.

6.1 Strategies for Assigning Names to Relational Attributes

We discuss briefly below strategies for assigning names to attributes of relational schemas representing EER schemas. These strategies depend on the way users interface with the database:

1. Interface at the EER level. If users access the database through an EER interface, then the symbolic attribute names generated by ***Crep*** can be used for a relational implementation of the database.
2. Interface at the relational level. If EER schemas are used only for design, then users and application programs access the corresponding database through a relational interface (e.g. using SQL). For such interfaces meaningful (rather than symbolic) names for relational attributes are appropriate.
3. Interface using attribute names only. There is a school of thought that claims that user interfaces should deal only with attribute names; then relations are invisible to users and attribute names must have a global independent meaning. This is one of the goals of *Universal Relation* (UR) interfaces [17]. For such interfaces a name assignment that generates as few as possible global names is needed.

In this section we present only a global attribute name assignment algorithm that can be used for both UR and regular relational interfaces. Another name assignment algorithm is given in [23].

There are some common sense principles that one can follow in choosing a name assignment algorithm for relational attributes. We chose a combination of the following principles:

1. *Retention of EER Names.* We assume that much thought was given to the selection of names in the EER schema, therefore relational attribute names should be as close as possible to these names. Accordingly, we derive relational attribute names from the names of EER attributes, object-sets, and roles. We concatenate these names (using the customary "." notation, such as COURSE.NAME) when necessary. Furthermore, we chose not to make up new names and to avoid any systematic abbreviation of names (such as CR.NM for COURSE.NAME) because this can obscure their meaning.
2. *Clarity.* It is important to assign names that carry the semantic clarity intended in the EER design. Thus, as a general criterion, we prefer to assign a name to a foreign attribute that includes the name of the object-set to which the corresponding EER attribute belongs. In the relational schema of figure 4.1, for example, foreign attribute A_{6_3} (which corresponds to EER attribute NAME of entity-set COURSE) can be assigned several reasonable names, such as NAME, COURSE.NAME, OFFER.NAME, or OFFER.COURSE.NAME. For the sake of clarity we prefer COURSE.NAME because this name reflects the original association in the EER schema. Using the name of the EER attribute alone (e.g. NAME) may obscure this association, even if this name happens to be unique.
3. *Brevity.* Long names are difficult to remember and can be confusing. In the example above, the attribute name OFFER.COURSE.NAME is unnecessarily long. In general it is easy to understand names that have two components, one corresponding to the object-set and one to the EER attribute, such as COURSE.NAME. While in most cases it is possible to limit the number of name components to two, for some structures more than two components are necessary.

Finally, criteria such as those discussed above should not compromise the correctness of the translation nor restrict the functional capability of EER modeling.

Following the criteria discussed above, we develop below an algorithm called ***Assign$_G$*** that assigns *global* names to relational attributes so that the assumptions underlying normalization are satisfied, and the resulting number of relational attributes is minimal[†]. This algorithm can be used for both regular relational interfaces, and for UR interfaces whose users are expected to know only attribute names.

[†] Recal that two attributes are considered identical iff they are assigned the same *global* name.

6.2 A Global Name Assignment Algorithm

A relational schema that is going to be normalized must satisfy the assumptions discussed in section 5.2; for schemas generated by ***Crep*** these assumptions imply that the global names assigned to relational attributes must satisfy conditions **At1**, **At2**, and **At3**.

Let G_{ER} be an EER diagram and let $RS = (R, F \cup I \cup N)$ be the relational schema generated by ***Crep*** from the EER schema associated with G_{ER}. The foreign attributes of a relation-scheme $R_i(X_i)$ of R are involved in left-hand sides of inclusion dependencies. When the name of a foreign attribute A is different from the name of the attribute to which A corresponds via an inclusion dependency, A is said to be *renamed*. Following proposition 5.1, the inclusion dependency digraph G_I associated with RS is isomorphic to the subgraph of G_{ER} induced by the object-set vertices, and the attribute digraph G_A associated with RS is a subgraph of G_I. Clearly, the edges of G_I which do not belong to G_A are those corresponding to renamed foreign attributes, that is, if $R_i \rightarrow R_j$ is an edge of G_I but not an edge of G_A, then some foreign attributes of R_i are renamed. Undirected cycles in the attribute digraph associated with a relational schema generated by ***Crep*** must be only of the form allowed by condition **At3**. Consequently, cycles that have a different form must be broken by renaming attribute names. However, in order to avoid the proliferation of relational attributes (every renamed attribute is an additional attribute), renaming should be done only when necessary so that the number of renamed attributes would be kept at a minimum.

We specify below a procedure that determines a strategy for renaming foreign attributes so that the corresponding attribute digraph will be free of undesired cycles, and the overall number of attributes (i.e. global names) will be minimal. Note that the foreign attributes corresponding to attributes of another relation-scheme must be either renamed together or not renamed at all in order to satisfy condition **At2**. We rename attributes using role names whenever possible. Given a generalization-source entity-set E_i, if E_i has a specialization entity-set E_j that is involved in relationship-set R_k then (i) if E_j has a role in R_k then this role is considered also as the role of E_i in R_k otherwise (ii) the name of E_j is considered as the role of E_i in R_k. For example, in the EER schema of figure 3.1 FACULTY is considered as the role of entity-set EMPLOYEE in relationship-set TEACH.

The main steps of the procedure are exemplified in figure 6.1. Starting with an EER diagram G_{ER}, (1) first we construct the corresponding inclusion dependency digraph G_I (figure 6.1(ii)); then (2) we unify in G_I the vertices corresponding to entity-sets that belong to the same generalization structure, thus obtaining a reduced digraph G'_I (figure 6.1(iii)); (3) next we associate with every edge of G'_I a *weight* that represents the number of additional relational attributes which would result by renaming the foreign attributes involved in the left-hand side of the inclusion dependency corresponding to that edge (see the edge labels in figure 6.1(iii)); additionally, edges that correspond to EER edges that are associated with roles have a star (*) label; (4) we find the edges of G'_I that must be removed from G_A in order to break the undesirable cycles mentioned above; in order to minimize the number of renamed attributes we must maximize the number of foreign attributes that are not renamed, therefore we find the subgraph of G'_I corresponding to the maximum spanning tree of the underlying graph of G'_I; if there are multiple choices in generating the maximum spanning tree, then edges without a star label are preferred (figure 6.1(iv)). The foreign attributes that are involved in the left-hand sides of the inclusion dependencies corresponding to the edges that do not belong to the spanning tree found in (4) are candidates for renaming.

Definition 6.1 – ***Mark***$_G$.

Input: An EER diagram G_{ER} describing an EER schema.

Output: Marked edges of G_I and G_{ER}.

1. Construct the inclusion dependency digraph $G_I = (V, H)$ isomorphic to the subgraph of G_{ER} induced by the object-set vertices.

2. Construct the digraph $G'_I = (V', H')$ as follows:

   V' : remove from V vertices that correspond to specialization entity-sets;

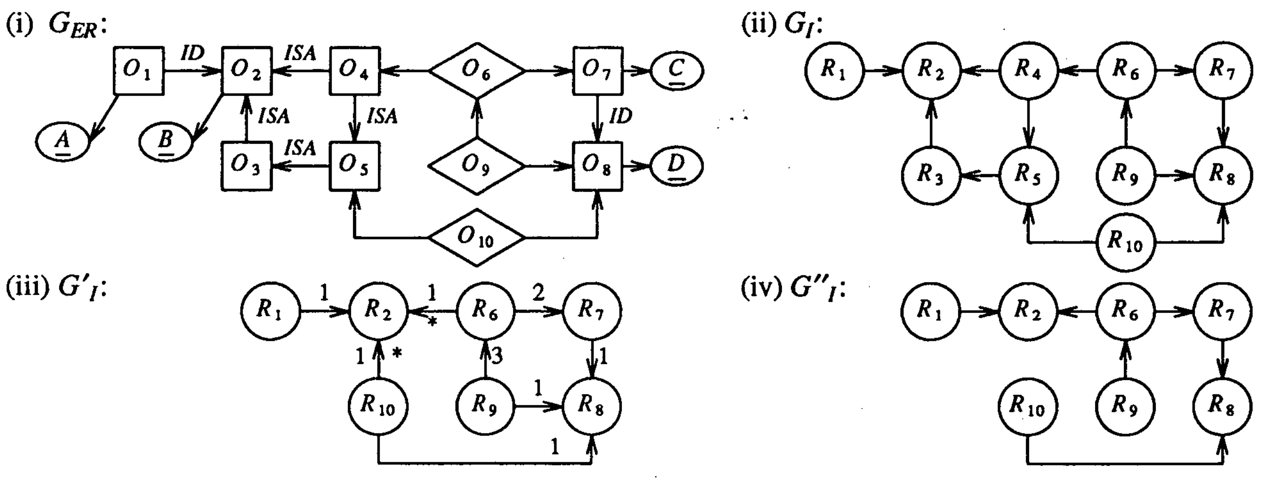


Figure 6.1 Marking Inclusion Dependency Digraph Edges for Global Name Assignment.

H' : (a) remove from H edges that correspond to ISA-labeled edges and (b) replace edges of H of the form $R_i \rightarrow R_j$ where R_j corresponds to a specialization entity-set E_j, by $R_i \rightarrow R_k$ where R_k corresponds to the generalization-source of E_j.

3. Every edge of H', $R_i \rightarrow R_j$, is associated with a *weight*, ω_{ij}, representing the number of foreign attributes in $R_i(X_i)$ that correspond to attributes in relation-scheme $R_j(X_j)$.

4. Find a connected subgraph of G'_I, $G''_I = (V'', H'')$, whose underlying graph is the *maximum* spanning tree of the underlying graph of G'_I, that is, such that $V'' = V'$ and for which the sum of edge weights is maximum; in every step of choosing an edge for the maximum spanning tree, edges without a star label are preferred over edges with a star label.

5. ***Mark*** the edges of G_I that belong to $(H' - H'')$, and the edges of G_{ER} that correspond to the marked edges of G_I. ■

The specification of ***Assign***$_G$ is given below. An example of applying ***Assign***$_G$ is given in figure 6.2, where ***Assign***$_G$ is applied to the relational schema of figure 4.1; in this example only one attribute (A_{84}) has been renamed in order to obtain an attribute digraph free of undirected cycles.

Definition 6.2 – ***Assign***$_G$.

Let $RS = (R, F \cup I \cup N)$ be the relational schema generated by ***Crep*** from EER schema *EERS*. For every relation-scheme $R_i(X_i)$ of R, where $R_i(X_i)$ corresponds to object-set O_i of *EERS*, every attribute A_{i_m} of X_i is assigned a *global name* as follows:

```
If  $A_{i_m}$ corresponds directly to an EER attribute of $O_i$
  Then  $A_{i_m}$ is assigned the name of that EER attribute prefixed by the name of $O_i$ ;
  Else  let $A_{i_m}$ correspond to relational attribute $A_{j_n}$ of relation-scheme $R_j(X_j)$,
                                                where $R_j$ corresponds to object-set $O_j$ ;
    If $O_i \rightarrow O_j$ is not marked by Mark$_G$ Then $A_{i_m}$ is assigned the global name of $A_{j_n}$ EndIf
    If $O_i \rightarrow O_j$ is marked by Mark$_G$ Then $A_{i_m}$ is assigned the global name of $A_{j_n}$ in $X_j$ in which the
                                                prefix is replaced by the role of $O_j$ in $O_i$ , or (when
                                                no such role exists) is prefixed by the name of $O_i$.
    EndIf
EndIf ■
```

Proposition 5.1. Let $RS = (R, F \cup I \cup N)$ be a relational schema generated by ***Crep***, and let the relational attribute names in RS be assigned by ***Assign***$_G$. Then these attribute names (i) are consistent with the specification of ***Crep***, (ii) satisfy conditions **At1**, **At2**, and **At3**, and (iii) are minimal in number when (i) and (ii) hold.

Proof Sketch : (i) The proof is by induction on the number of steps of ***Crep*** and is based on the assumptions regarding EER names (see section 3.3). (ii) Conditions **At1** and **At2** are satisfied initially by assigning to relational attributes the names of the corresponding EER attributes, prefixed by the names of their associated object-sets. The renaming involved in ***Assign***$_G$ does not affect **At1**; for **At2** and **At3** the proof follows the specification of ***Assign***$_G$. (iii) Suppose that an attribute that is renamed in ***Assign***$_G$ is not renamed. Then it can be verified that either condition **At2** or **At3** is not satisfied. ∎

Relation	**Non–Foreign Attributes**	**Foreign Attributes**		
R_1	A_{1_1} := E.NAME, A_{1_2} := E.ADDRESS			
R_2	A_{2_1} := D.NAME, A_{2_2} := D.ADDRESS			
R_3	A_{3_1} := C.NAME			
R_4	A_{4_1} := F.RANK	A_{4_2} := E.NAME,	A_{4_3} := E.ADDRESS	
R_5	A_{5_1} := W.NAME	A_{5_2} := E.NAME,	A_{5_3} := E.ADDRESS	
R_6		A_{6_1} := D.NAME,	A_{6_2} := D.ADDRESS,	A_{6_3} := C.NAME
R_7		A_{7_1} := F.RANK,	A_{7_2} := E.NAME,	A_{7_3} := E.ADDRESS
		A_{7_4} := D.NAME,	A_{7_5} := D.ADDRESS,	A_{7_6} := C.NAME
R_8		A_{8_1} := F.RANK,	A_{8_2} := E.NAME,	A_{8_3} := E.ADDRESS,
		A_{8_4} := S.C.NAME		

Abbreviations: C=COURSE, D=DEPARTMENT, E=EMPLOYEE, F=FACULTY, S=SUPERVISE, W=DEPENDENT

Figure 6.2 Global Names Assigned Under ***Assign***$_G$ to the Relational Attributes of Figure 4.1.

7. Normalization of Canonical Relational Schemas

The framework discussed in section 5.2 allows us to analyze the normal form of canonical relational schemas generated by ***Crep***. First we examine below how keys are computed for these relational schemas. Then we propose a normalization procedure transforming canonical relational schemas into BCNF schemas; this procedure can be applied either before or after assigning names to relational attributes.

7.1 Computing Keys in Canonical Relational Schemas Representing EER Schemas

We show first that the keys of a relation-scheme in a canonical relational schema generated by ***Crep*** can be computed using only a subset of functional dependencies.

Proposition 7.1. Let $RS = (R, F \cup I \cup N)$ be a canonical relational schema generated by ***Crep***, and let $\tilde{F}$ be the closure of F under the following derivation rule: from $R_i[\overline{X}\ \overline{Y}\ \overline{Z}] \subseteq R_j[\overline{X'}\overline{Y'}\overline{Z'}]$ and $R_j : X' \rightarrow Y'$ derive $R_i : X \rightarrow Y$. Then a functional dependency σ belongs to $\tilde{F}^+$ iff σ belongs to $(F \cup I \cup N)^+$.

Proof: The nulls-not-allowed constraints have obviously no effect on the derivation of functional and inclusion dependencies, therefore we must show that $\sigma \in \tilde{F}^+$ iff $\sigma \in (F \cup I)^+$.

(only-if) see proposition 4.1 of [5].

(if) Clearly, $(F \cup I)^+ = (\tilde{F} \cup I)^+$. Let $\sigma \in (\tilde{F} \cup I)^+$. We show that then $\sigma \in \tilde{F}^+$. Assume that $\sigma = R_i : Y \rightarrow A$ and that $\sigma \notin \tilde{F}^+$. We show that then $\sigma \notin (\tilde{F} \cup I)^+$ by constructing a database state that satisfies $(\tilde{F} \cup I)$ but does not satisfy σ.

Let r be a database state associated with RS whose relations are all empty except r_i which contains two tuples t_1 and t_2 such that $t_1[Y] = t_2[Y]$ and $t_1[A] \neq t_2[A]$ for each attribute A of $(X_i - Y^+)$. Clearly, r satisfies F. The following rule is applied as long as there exists an inclusion dependency $R_j[\overline{X}_{j_k}] \subseteq R_k[\overline{X}_k] \in I$ which is violated by r: for each tuple $t_j \in r_j$ such that $\nexists\ t_k \in r_k$ with $t_k = t_j[X_{j_k}]$ add to r_k a tuple t, so that $t[\overline{X}_k] = t_j[\overline{X}_{j_k}]$. Since I is acyclic this process does not add any new tuple to r_i and terminates by generating a state r' that satisfies I but does not satisfy σ.

The proof is completed by showing that r' satisfies $\tilde{F}$ and thus satisfies $(\tilde{F} \cup I)$. First note that a tuple is added to r_k, where r_k is associated with $R_k(X_k) \in R$, iff there is directed path from R_i to R_k in the inclusion dependency digraph associated with RS. Moreover, any such tuple is of the form $t_i[X_k]$ where t_i is a tuple of r_i. Suppose that a tuple t_k is added to r_k and that the new state does not satisfy $\tilde{F}$, that is, there exists a functional dependency $R_k : Y' \rightarrow Z'$ in $\tilde{F}$ and a tuple t'_k in r_k such that $t_k[Y'] = t'_k[Y']$ and $t_k[Z'] \neq t'_k[Z']$. Then there exist two tuples in r_i, t_i and t'_i, such that $t_k = t_i[X_k]$ and $t'_k = t'_i[X_k]$. It can be shown, by induction on the number of steps in the process that generates $\tilde{F}$, that then r_i also does not satisfy a functional dependency of $\tilde{F}$, which contradicts the conclusion above concerning r_i. ■

We show now that the keys of a relation-scheme R_i can be computed using only functional dependencies of $\tilde{F}$ associated with R_i. We denote by $\tilde{F}_i$ the subset of $\tilde{F}$ associated with (having *tag*) R_i, and by $\tilde{F}_U$ the set of *untagged* functional dependencies corresponding to $\tilde{F}$.

Proposition 7.2. Let $RS = (R,\ F \cup I \cup N)$ be a canonical relational schema generated by ***Crep***. Let $R_i\ (X_i) \in R$ and $YZ \subseteq X_i$. Then $Y \rightarrow Z$ belongs to $\tilde{F}_U^+$ iff $R_i : Y \rightarrow Z$ belongs to $\tilde{F}_i^+$.

Proof Sketch : (only-if) obvious. (if) The proof is based on the following claim.

Claim. Let $R_i(X_i) \in R$ and $YZ \subseteq X_i$. Then $(\ Y^+ \cap X_i\ )$ is included in the closure of Y with respect to $\tilde{F}_i$.

We omit the proof of this claim and note only that it is based on condition At3 concerning the attribute digraph associated with R. Now suppose that $Y \rightarrow Z \in \tilde{F}_U^+$ but $R_i : Y \rightarrow Z \notin \tilde{F}_i^+$. Then Z belongs to Y^+, but not to the closure of Y with respect to $\tilde{F}_i$, thus contradicting the claim. ■

Now the candidate keys can be computed in the usual way and the relation-schemes can be associated with primary keys. The definition of keys and foreign-keys below is used later in the specification of the normalization procedure. Although well known, the result presented below is valid only under the conditions discussed in section 5.2.

Proposition 7.3. Let $RS = (R,\ F \cup I \cup N)$ be a canonical relational schema generated by ***Crep*** from an EER schema *EERS*. For every relation-scheme $R_i(X_i)$ of R let (a) O_i be the object-set of *EERS* corresponding to R_i; (b) Z_i be the subset of X_i corresponding to the identifier of O_i; (c) T_i denote the set of relation-schemes $\{R_j(X_j) \mid R_i[\bar{X}_{i_j}] \subseteq R_j[\bar{X}_j] \in I\}$; and (d) W_i be the union of all the *foreign−keys* of R_i, where each foreign-key W_{i_j} is in a one-to-one correspondence with the *primary key* of relation-scheme R_j of T_i. Then R_i is associated with keys as follows.

1. If O_i is an ***entity-set*** then the primary key is $Z_i W_i$.
2. If O_i is a *relationship−set* then if all the cardinalities of the object-sets involved in O_i are *many*, then (i) the primary key is W_i; otherwise (ii) for every object-set O_{i_j} that has cardinality *one* in O_i, $(W_i - W_{i_j})$ is a candidate key, where W_{i_j} is the *foreign−key* that references the relation-scheme that represents O_{i_j}, and one of the candidate keys is selected as the primary-key.

Proof : is based on propositions 7.1 and 7.2. Details are omitted. ■

Note that for independent entity-sets attribute-set W_i is empty, for specialization entity-sets attribute-set Z_i is empty and all foreign-keys are equal, and for weak entity-sets and relationship-sets all foreign-keys are pairwise disjoint.

7.2 Normalization Procedure for Canonical Relational Schemas Representing EER Schemas

Crep generates relational schemas which are not generally in a high normal form. For example, relation-scheme R_6 (X_6) in figure 4.1, whose primary key is A_{6_3}, is not in 3NF because of the transitive dependency $A_{6_1} \rightarrow A_{6_2}$. In this section we define a procedure called ***Norm*** that transforms canonical relational schemas generated by ***Crep*** into equivalent BCNF schemas. Further, we show that ***Norm*** generates schemas without ***redundant*** attributes, where an attribute is considered redundant if its removal has no effect on the information-capacity of the schema. Thus, the composition of ***Crep*** and ***Norm*** generates relational schemas that are correct, without redundant attributes, and in BCNF.

Definition 7.1 (***Norm***). Let $RS = (R,\ F \cup I \cup N)$ be a canonical relational schema generated by ***Crep***. Given relation-scheme $R_i(X_i)$ of R, let T_i denote the set of relation-schemes $\{R_j \mid R_i[\overline{X}_{i_j}] \subseteq R_j[\overline{X}_j] \in I\}$. Let $R_i(X_i)$ be a relation-scheme of R associated with relation r_i; $R_i(X_i)$ (resp. r_i) is said to be *mappable* if either T_i is empty or all the relation-schemes of T_i (resp. relations associated with relation-schemes of T_i) have been mapped already.

Norm maps $RS = (R,\ F \cup I \cup N)$ into $RS' = (R',\ F' \cup I' \cup N')$ by successively mapping *mappable* relation-schemes $R_i(X_i)$ of R as follows:

1. $R_i\ (X_i)$ is mapped into $R'_i(X'_i)$, where X'_i is generated by removing from X_i the foreign attributes that do not belong to any foreign-key of R_i ;
2. every functional dependency of F associated with R_i, $R_i : Z \rightarrow W$, is mapped into functional dependency $R'_i : Z' \rightarrow W'$ of F', such that Z' and W' are generated by removing the attributes of $(X_i - X'_i)$ from Z and W, respectively;
3. every inclusion dependency of I involving R_i in its left-hand side, $R_i[\overline{X}_{i_j}] \subseteq R_j[\overline{X}_j]$, is mapped into inclusion dependency $R'_i[\overline{Y}] \subseteq R'_j[\overline{Z}]$ of I', such that Y and Z are generated by removing from X_{i_j} the attributes of $(X_i - X'_i)$, respectively by removing from X_j the attributes corresponding to the attributes removed from X_{i_j};
4. the nulls-not-allowed constraint of N associated with R_i, $R_i : \varnothing \overset{\mathrm{EX}}{\rightarrow} X_i$, is mapped into nulls-not-allowed constraint $R'_i : \varnothing \overset{\mathrm{EX}}{\rightarrow} X'_i$ of N' .

Norm is associated with two *state mappings*, η and η', where η maps a database state r of RS into a database state r' of RS', and η' maps a state r' of RS' into a database state $\tilde{r}$ of RS as follows:

η maps every relation r_i of r into $r'_i = \pi_{X'_i}(r_i)$;

η' successively maps *mappable* relations r'_i of r' into $\tilde{r}_i = r'_i \underset{R_j \in T_i}{\bowtie} rename(\tilde{r}_j, \bar{X}_j \leftarrow \bar{X}_{i_j})$. ■

For example, ***Norm*** maps the relational schema of figure 4.1 into the normalized schema of figure 7.1(i). As already mentioned, ***Norm*** also can be applied after assigning names to relational attributes, provided that these names satisfy the conditions defined in section 5.2. For example, if the name assignment algorithm presented in section 6 is used to assign names to the attributes in the schema of figure 4.1, then the result of applying ***Norm*** on this schema is the schema shown in figure 7.1(ii).

Procedure ***Norm*** is well-defined: it terminates due to the acyclicity of I, and it generates syntactically correct schemas. The following proposition shows that ***Norm*** generates relational schemas that have equivalent information-capacity (in the sense of definition 5.1) with the corresponding schemas on which ***Norm*** is applied.

Proposition 7.4. Let $RS = (R, F \cup I \cup N)$ be a canonical relational schema generated by ***Crep***, and let $RS' = (R', F' \cup I' \cup N')$ be the result of applying ***Norm*** on RS. Then RS and RS' have equivalent information-capacities.

(i) Relation–Schemes	Inclusion Dependencies	(ii) Relation–Schemes	Inclusion Dependencies
$R_1(\underline{A_{1_1}}, A_{1_2})$	$R_4[A_{4_2}] \subseteq R_1[A_{1_1}]$	R_1 (<u>E.NAME</u>,E.ADDRESS)	R_4[E.NAME] $\subseteq R_1$[E.NAME]
$R_2(\underline{A_{2_1}}, A_{2_2})$	$R_5[A_{5_2}] \subseteq R_1[A_{1_1}]$	R_2 (<u>D.NAME</u>,D.ADDRESS)	R_5 [E.NAME] $\subseteq R_1$[E.NAME]
$R_3(\underline{A_{3_1}})$	$R_6[A_{6_1}] \subseteq R_2[A_{2_1}]$	R_3 (<u>C.NAME</u>)	R_6[D.NAME] $\subseteq R_2$[D.NAME]
$R_4(A_{4_1}, \underline{A_{4_2}})$	$R_6[A_{6_3}] \subseteq R_3[A_{3_1}]$	R_4 (F.RANK,<u>E.NAME</u>)	R_6[C.NAME] $\subseteq R_3$[C.NAME]
$R_5(\underline{A_{5_1}, A_{5_2}})$	$R_7[A_{7_2}] \subseteq R_4[A_{4_2}]$	R_5 (<u>W.NAME, E.NAME</u>)	R_7[E.NAME] $\subseteq R_4$[E.NAME]
$R_6(A_{6_1}, \underline{A_{6_3}})$	$R_7[A_{7_6}] \subseteq R_6[A_{6_3}]$	R_6 (D.NAME,<u>C.NAME</u>)	R_7[C.NAME] $\subseteq R_6$[C.NAME]
$R_7(A_{7_2}, \underline{A_{7_6}})$	$R_8[A_{8_2}] \subseteq R_4[A_{4_2}]$	R_7 (E.NAME,<u>C.NAME</u>)	R_8[E.NAME] $\subseteq R_4$[E.NAME]
$R_8(\underline{A_{8_2}}, A_{8_4})$	$R_8[A_{8_4}] \subseteq R_3[A_{3_1}]$	R_8 (<u>E.NAME</u>,S.C.NAME)	R_8[S.C.NAME] $\subseteq R_3$[C.NAME]

Abbreviations: C=COURSE, D=DEPARTMENT, E=EMPLOYEE, F=FACULTY, S=SUPERVISE, W=DEPENDENT

Notes: Nulls-not-allowed constraints are omitted. Primary keys are underlined.

Figure 7.1 Normalized Relational Schemas Corresponding to Schema of Figure 4.1.

Proof. The proof is based on the following claims.

Claim 1. Let r be a database state associated with RS and satisfying $(F \cup I \cup N)$.
Then $\eta(r)$ satisfies $(F' \cup I' \cup N')$.

Claim 2. Let r' be a database state associated with RS' and satisfying $(F' \cup I' \cup N')$.
Then $\eta'(r')$ satisfies $(F \cup I \cup N)$.

Proof : The proof of the first claim is straightforward.

The proof of the second claim is by induction on the number of steps of η'. First, observe that ***Norm*** specifies a one-to-one correspondence between RS and RS'. Let $R_i(X_i) \in R$ correspond by ***Norm*** to $R'_i(X'_i) \in R'$. Initially $\tilde{r}$ is empty. The inductive hypothesis is that before r'_i is mapped by η' into $\tilde{r}_i$, $\tilde{r}$ satisfies the dependencies of $(F \cup I \cup N)$. By definition, when r'_i is mapped by η' into $\tilde{r}_i$, all the relations associated with the relation-schemes in the set $\{R'_j \mid R'_i[\overline{Y}] \subseteq R'_j[\overline{Z}] \in I'\}$ have been already mapped. (a) The nulls-not-allowed constraints are obviously satisfied by $\tilde{r}$.

(b) Satisfying the inclusion dependencies of I (e.g. those of the form $R_i[\overline{X}_{i_j}] \subseteq R_j[\overline{X}_j]$) by $\tilde{r}$ after the mapping of r'_i, is guaranteed by the natural join in the definition of η'.

(c) Assume that $\tilde{r}_i$ does not satisfy some functional dependency $R_i : Z \rightarrow W \in F$ which corresponds by ***Norm*** to $R'_i : Z' \rightarrow W' \in F'$. Then there exist two tuples t_1 and t_2 in $\tilde{r}_i$ such that $t_1[Z] = t_2[Z]$ and $t_1[W] \neq t_2[W]$. The definition of η' implies that $t_1[X'_i] \in r'_i$ and $t_2[X'_i] \in r'_i$. Since r'_i satisfies $R'_i : Z' \rightarrow W'$, it follows that $t_1[W - W'] \neq t_2[W - W']$. By definition of ***Norm***, the attributes of $(W - W')$ can be only foreign attributes that do not belong to any foreign-key. Let some attribute A of $(W - W')$ correspond to an attribute of relation-scheme $R_j \in R$, and let $Y \subseteq X_i$ be the foreign-key of R_i referencing R_j. Then, following ***Crep***, I includes the inclusion dependency $R_i[\overline{X}_{i_j}] \subseteq R_j[\overline{X}_j]$, where $AY \subseteq X_{i_j}$. The analysis of the functional dependencies generated by ***Crep*** shows that if A appears in the right-hand side of a functional dependency, then the attributes of Y also appear in the right-hand side of that functional dependency. By definition, if A is removed by ***Norm*** from the right-hand side of some functional dependency, then the attributes of Y are not removed, that is, $Y \subseteq W'$. Consequently, $t_1[Y] = t_2[Y]$. Since $\tilde{r}_i$ satisfies $R_i[\overline{X}_{i_j}] \subseteq R_j[\overline{X}_j]$, $AY \subseteq X_{i_j}$, and Y corresponds to a primary key of R_j, it follows that $t_1[A] = t_2[A]$, which contradicts the hypothesis above.

The last condition of definition 5.1 is obviously satisfied by both η and η'. Following claims 1 and 2, the first two conditions of definition 5.1 are satisfied. We must show that condition 3 of definition 5.1

is also satisfied.

(a) $\underline{\eta' (\eta (r)) = r}$ follows from the fact that ***Norm*** preserves the primary keys of the relation-schemes of R, and that the joins involved in η' are on primary keys.

(b) $\underline{\eta (\eta' (r')) = r'}$. The proof is by induction on the number of steps of η'. Let $R_i(X_i) \in R$ correspond by ***Norm*** to $R'_i(X'_i) \in R'$. The inductive hypothesis is that before r'_i is mapped by η' into $\tilde{r}_i$, every relation of $\tilde{r}$, $\tilde{r}_k$, satisfies the condition $\eta(\tilde{r}_k) = r'_k$. By definition, when $r'_i \in r'$ is mapped by η' into $\tilde{r}_i$, all the relations associated with the relation-schemes in the set $T = \{R'_j \,|\, R'_i[\bar{Z}] \subseteq R'_j[\bar{Y}] \in I'\}$ have been already mapped. Assume that $\eta(\tilde{r}_i) \neq r'_i$. Since clearly $\eta(\tilde{r}_i) \subseteq r'_i$, it follows that there exists a tuple $t \in r'_i$ such that $t \notin \tilde{r}_i[X'_i]$. Consequently, for some relation r'_j associated with a relation-scheme of T, t does not match any tuple of $\tilde{r}_j = \eta' (r'_j)$. From the inductive hypothesis, $\eta(\tilde{r}_j) = r'_j$, it follows that t cannot match any tuple of r'_j, but then the inclusion dependency $R'_i[\bar{Z}] \subseteq R'_j[\bar{Y}]$ is not satisfied by r'. ■

Mapping ***Norm*** simplifies the relational schemas generated by ***Crep*** by removing ***redundant*** attributes from the canonical relational schema. The following proposition shows that ***Norm*** removes all redundant attributes, and generates relational schemas in BCNF.

Proposition 7.5. Let $RS = (R,\ F \cup I \cup N\ )$ be a canonical relational-schema generated by ***Crep***, and let $RS' = (R'\ ,\ F' \cup I' \cup N'\ )$ be the result of applying ***Norm*** on RS. Then every relation-scheme of R' is in BCNF and free of redundant attributes.

Proof Sketch : Let $R'_i(X'_i)$ be a relation-scheme of R' which corresponds by ***Norm*** to $R_i(X_i) \in R$. Let r_i and r'_i be associated with R_i and R'_i, respectively. By definition, $r'_i = \pi_{X'_i}(r_i)$, therefore any functional dependency that is satisfied by r'_i is also satisfied by r_i. Thus, any partial or transitive dependency, $Y \to Z$, satisfied by r'_i is also satisfied by r_i, that is, $R_i : Y \to Z \in \tilde{F}_i^+$. Suppose $R_i : Y \to Z \in \tilde{F}_i$ is a partial (transitive) dependency. Then $R_i : Y \to Z \notin F_i$. Let $R_i : Y \to Z$ be added to $\tilde{F}_i$ because of $R_i[\bar{X}_{i_j}] \subseteq R_j[\bar{X}_j] \in I$ and $R_j : W \to Q \in \tilde{F}_j$, where Y and Z correspond to W and Q, respectively. Clearly both Y and Z must belong to some foreign-key of R_i, otherwise they would be removed by ***Norm***. Since either all foreign keys are equal (when R_i corresponds to a specialization entity-set) or X_{i_j} and $(X_i - X_{i_j})$ are disjoint (when R_i corresponds to a weak entity-set or a relationship-set), it follows that Y and Z belong to the same foreign-key. But then WQ would belong to the primary key of R_j which is a contradiction.

Norm removes the foreign attributes that do not belong to foreign-keys, so it must be shown that foreign-key attributes are not redundant. Suppose that a foreign-key attribute is removed from $R_i(X_i)$ of R. Then a relation r'_i can be easily constructed so that $\eta' (r'_i)$ will not satisfy $\tilde{F}_i$. Details are omitted. ■

8. Merging Relations in Relational Schema Translations of EER Schemas

In the relational schemas generated by ***Crep*** and ***Norm*** every relation-scheme corresponds to a unique EER object-set. Sometimes it is desirable to decrease the number of relations in a database by *merging* relations, thus reducing the number of joins that need to be performed for achieving a better access performance. In this section we present a procedure for merging relation-schemes in relational schemas generated by ***Norm***.

Merging several relation-schemes into a new relation-scheme, R_m, must involve a state mapping that ensures that all the tuples of the relations associated with the merged relation-schemes have corresponding tuples in the relation associated with R_m. This requirement can be fulfilled by using *outer-joins* on primary-keys in defining such a state mapping. The result of outer-joins usually contains null values. These null values must be restricted in order to ensure that the original relations can be reconstructed from the new relation without losing or adding information, that is, in order to preserve the information-capacity of the merged relations. Such restrictions are expressed using *null constraints*. In certain cases merging can be carried out without requiring null constraints that are more complex than nulls-not-allowed constraints. We present in this section a procedure called ***Smerge*** that carries out merging in such cases. ***Smerge*** can be applied on schemas generated by ***Norm***, and preserves the information capacity and normal form of these schemas. A less restricted merging procedure, but involving more complex null constraints, is presented in [20].

Procedure ***Smerge*** specified below takes a schema generated by ***Norm***, $RS = (R, F \cup I \cup N)$, and under certain conditions maps it into a new relational schema, $RS' = (R', F' \cup I' \cup N')$; given a subset $\tilde{R}$ of R, such that the primary-keys associated with the relation-schemes of $\tilde{R}$ are pairwise compatible, R' results by replacing the relation-schemes of $\tilde{R}$ with a new relation-scheme, R_m, and F', I' and N' result by adjusting in RS the key dependencies, inclusion dependencies, and nulls-not-allowed constraints, respectively. The foreign-key attributes of R_m that are not included in the primary-key of R_m are allowed to have null values, and the new dependencies and constraints associated with RS' ensure that the relations involved in merging can be reconstructed from r_m without loss of information.

Definition 8.1. – ***Smerge.***

Input: a relational schema generated by ***Norm***, $RS = (R, F \cup I \cup N)$, and a subset of R, $\tilde{R}$, such that

(i) the primary-keys associated with the relation-schemes of $\tilde{R}$ are pairwise compatible;

(ii) there exists a relation-scheme $R_k(X_k)$ in $\tilde{R}$ that satisfies the following condition:

for every relation-scheme R_i of $\tilde{R}$, $i \neq k$, $R_i[\overline{K_i}] \subseteq R_k[\overline{K_k}] \in I$;

(iii) for every relation-scheme $R_i(X_i)$ of $\tilde{R}$, $i \neq k$, the following conditions are satisfied:

a. $|\ Y\ | = 1$, where $Y = X_i - K_i$;

b. R_i is not involved in the right-hand side of any inclusion dependency of I;

c. In addition to the inclusion dependency involving R_k, R_i can be involved in the left-hand side of at most one additional inclusion dependency, of the form $R_i[\overline{Y}] \subseteq R_j[\overline{K_j}]$.

Output: a relational schema $RS' = (R', F' \cup I' \cup N')$.

Smerge$(\tilde{R})$ applied on RS generates RS' as follows:

1. R' results by replacing in R the relation-schemes of $\tilde{R}$ with a new relation-scheme, $R_m(X_m)$, such that $K_m := K_k$, $X_m := K_m \bigcup_{R_i(X_i) \in \tilde{R}} (X_i - K_i)$;

2. F' results by replacing in F the key dependencies involving primary-keys associated with relation-schemes of $\tilde{R}$ with key dependency $R_m : K_m \rightarrow X_m$; in other (candidate) key dependencies associated with relation-schemes of $\tilde{R}$, the name of the relation-scheme (its *tag*) is replaced with R_m;

3. I' results by replacing R_i with R_m and K_i with K_m in every inclusion dependency of I that involves a relation-scheme R_i of $\tilde{R}$; subsequently, inclusion dependencies of the form $R_m[\overline{K_m}] \subseteq R_m[\overline{K_m}]$ are removed from I';

4. N' results by replacing in N the nulls-not-allowed constraints associated with relation-schemes of $\tilde{R}$ with nulls-not-allowed constraint $R_m : \varnothing \overset{\text{EX}}{\rightarrow} X_k$.

Smerge$(\tilde{R})$ is associated with *state mappings*, η and η', where η maps a database state r of RS into a database state r' of RS', and η' maps a database state r' of RS' into a database state $\tilde{r}$ of RS as follows:

η is the *identity* for the relations of r that are associated with relation-schemes of $(R - \tilde{R})$; and maps the set of relations $\{r_i \mid r_i \in r,\ r_i$ is associated with R_i of $\tilde{R}\ \}$ into r_m as follows:

(i) $r_m := r_k$; (ii) <u>for</u> each R_i of $(\tilde{R} - \{R_k\})$ <u>do</u> $r_m := \pi_{(X_m - K_i)} (r_m \underset{\bar{K}_m = \bar{K}_i}{\overset{o}{\bowtie}} r_i)$ <u>enddo</u>;

η' is the *identity* for every relation of $(r' - \{r'_m\})$; and maps relation r'_m of r' into relations $\tilde{r}_i$ as follows: $\tilde{r}_i :=$ ***rename*** $(\pi{\downarrow}_{K_m(X_i - K_i)} (r'_m), \bar{K}_m \leftarrow \bar{K}_i)$, where $R_i(X_i)$ is a relation-scheme of $\tilde{R}$.

An example of applying ***Smerge*** is shown in figure 8.1, where ***Smerge*** is applied on the relational schema of figure 7.1(ii) in order to merge relation-schemes R_4 and R_8 into R_9, and relation-schemes R_6 and R_7 into R_{10}.

We prove below that procedure ***Smerge*** preserves the information-capacity and normal form of the relational schemas on which it is applied.

Proposition 8.1. Let $RS = (R, F \cup I \cup N)$ and $\tilde{R}$ be defined as in definition 8.1. Let $RS' = (R', F' \cup I' \cup N')$ be the result of applying ***Smerge***$(\tilde{R})$ on RS. Then (i) RS and RS' have equivalent information-capacities; and (ii) the relation-schemes of R' are in BCNF.

Proof Sketch. (i) The proof refers to the conditions of definition 5.1, and concerns only the relation-schemes of R affected by merging; for the first two conditions, the proof follows the definition of η and η' and is straightforward; for the third condition, the proof follows from the fact that ***Smerge*** preserves the primary-keys associated with the relation-schemes of $\tilde{R}$, and that the outer-joins involved in η are all on primary keys; the last condition is obviously satisfied. (ii) It can be verified that all functional dependencies implied by $(F' \cup I' \cup N')$ are key dependencies. The proof is based on the fact that the closure of F' can be computed independently of I' (see [1]) and is not affected by the constraints of N'. ■

The limits of the merging carried out by ***Smerge*** are embodied by input conditions (ii) and (iii). Thus, in order to remove input condition (iii.a) from the definition of ***Smerge*** the following set of null-

Relation-Schemes (*keys are underlined*)	**Inclusion Dependencies**	**Nulls-Not-Allowed Constraints**
R_1 ($\underline{\text{E.NAME}}$, E.ADDRESS)	R_5 [E.NAME] $\subseteq R_1$ [E.NAME]	R_1 : $\varnothing \overset{\text{EX}}{\rightarrow}$ E.NAME, E.ADDRESS
R_2 ($\underline{\text{D.NAME}}$, D.ADDRESS)	R_9 [E.NAME] $\subseteq R_1$ [E.NAME]	R_2 : $\varnothing \overset{\text{EX}}{\rightarrow}$ D.NAME, D.ADDRESS
R_3 ($\underline{\text{C.NAME}}$)	R_9 [S.C.NAME] $\subseteq R_3$ [C.NAME]	R_3 : $\varnothing \overset{\text{EX}}{\rightarrow}$ C.NAME
R_5 ($\underline{\text{W.NAME, E.NAME}}$)	R_{10} [D.NAME] $\subseteq R_2$ [D.NAME]	R_5 : $\varnothing \overset{\text{EX}}{\rightarrow}$ W.NAME, E.NAME
R_9 ($\underline{\text{E.NAME}}$, F.RANK, S.C.NAME)	R_{10} [C.NAME] $\subseteq R_3$ [C.NAME]	R_9 : $\varnothing \overset{\text{EX}}{\rightarrow}$ E.NAME, F.RANK
R_{10} ($\underline{\text{C.NAME}}$, D.NAME, E.NAME)	R_{10} [E.NAME] $\subseteq R_4$ [E.NAME]	R_{10} : $\varnothing \overset{\text{EX}}{\rightarrow}$ C.NAME, D.NAME

Abbreviations: C=COURSE, D=DEPARTMENT, E=EMPLOYEE, F=FACULTY, S=SUPERVISE, W=DEPENDENT

Figure 8.1 The Relational Schema of Figure 7.1(ii) after Merging.

existence constraints must be defined for a relation-scheme $R_i(X_i)$ affected by merging [20]: $\{R_i : (Y - A_j) \overset{\text{EX}}{\rightarrow} A_j \mid A_j \in Y$, where $Y = X_i - K_i\}$; this set of null-existence constraints ensures that in every tuple of a merged relation the subtuple corresponding to the non-key attributes of X_i is either total or consists only of null values, that is, is not *partly null.* Consider, for example, a relational schema consisting of the relation-schemes EMPLOYEE, WORKS and PROJECT shown in figure 1.1(iii), together with the inclusion dependencies shown in figure 1.1(iv) and with nulls-not-allowed constraints that disallow the attributes to have null values. ***Smerge*** cannot be applied on this schema (because of condition (iii.a)). However, relation-schemes EMPLOYEE and WORKS can be merged into relation-scheme EMPLOYEE′ ($\underline{\text{EN}}$, DN, PN, DATE), where attributes PN and DATE are allowed to have null values, provided that this relation-scheme is also associated with null-existence constraints DATE $\overset{\text{EX}}{\rightarrow}$ PN and PN $\overset{\text{EX}}{\rightarrow}$ DATE which ensure that in every tuple of a relation associated with EMPLOYEE′, the values of attributes PN and DATE are either both null or both non-null. Note that without these null-existence constraints the information capacity of the original schema would not be preserved.

In terms of EER schemas, the input conditions of definition 8.1 imply that multiple object-sets can be represented by a single relation involving only nulls-not-allowed constraints if these multiple object-sets consist of:

M1. an entity-set E_i and its specialization entity-sets, provided that these specialization entity-sets (a) have no specializations of their own and are directly generalized only by E_i, (b) are not involved (by aggregation) in relationship-sets or weak entity-sets, and (c) have exactly one local (not inherited) attribute of their own (see figure 8.2(i)); or

M2. an object-set O_i and *binary many–to–one* relationship-sets in which O_i is involved with a *many* cardinality, provided that these relationship-sets (a) have no attributes, (b) are not involved (by aggregation) in any other relationship-set, and (c) O_i is associated by these relationship-sets with independent entity-sets that have single-attribute entity-identifiers (see figures 8.2(ii) and 8.2(iii)).

Consider, for example, the EER schema of figure 3.1. Entity-sets EMPLOYEE and FACULTY satisfy conditions (M1.a) and (M1.c), but do not satisfy condition (M1.b); similarly, entity-set COURSE and relationship-set OFFER satisfy conditions (M2.a) and (M2.c), but do not satisfy condition (M2.b). Consequently, the relation-schemes that correspond to these object-sets in the relational schema of figure 7.1(ii), cannot be merged by ***Smerge***. Conversely, entity-set FACULTY and relationship-set SUPERVISE, respectively relationship-set OFFER and relationship-set TEACH, satisfy conditions (M2.a), (M2.b), and (M2.c), and therefore the corresponding relation-schemes in the relational schema of figure 7.1(ii) can be merged using ***Smerge***, as shown in figure 8.1.

Merging relations in a relational database whose schema is the translation of an EER schema, usually obscures the semantics of the database application for users. While the meaning of a relation that corresponds to exactly one object-set remains clear, the semantics of a relation that corresponds to multiple object-sets are usually harder to comprehend. Ideally, relation merging should be transparent to users, and users should continue to refer to independent, rather than embedded, objects. This can be accomplished by specifying as *views* the relations that are merged, so that every view will correspond to a unique object-set; queries and updates that refer to individual objects, can be then automatically mapped into manipulations of relations in the underlying database.

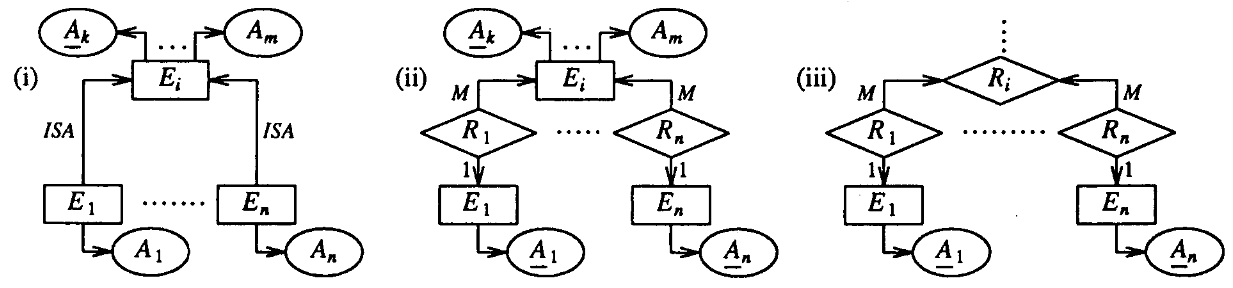


Figure 8.2 EER Structures Amenable for Representation by a Single Relation.

9. Concluding Remarks

We have proposed in this paper a modular approach to the translation of EER schemas into relational schemas, by separating the translation into four stages: (i) translate EER schemas into canonical relational schemas, (ii) assign names to relational attributes, (iii) normalize relational schemas representing EER schemas, and (iv) merge relation-schemes in relational schemas representing EER schemas.

We have defined procedures for translating EER schemas into canonical relational schemas and for normalizing canonical relational schemas. We have proved that the result of using these procedures together are relational schemas in BCNF. We have shown that the assumptions underlying normalization impose certain restrictions on the assignment of names to the attributes of relational schemas representing EER schemas, and discussed some strategies for assigning names to such attributes. In general, different name assignment algorithms can be used depending on the preferences of the users. As an example, we have presented a specific name assignment algorithm that assigns global names to attributes. It is worth noting that applying this algorithm to canonical relational schemas results in schemas that can be used in Universal-Relation interfaces [17]. A different name assignment algorithm is given in [23].

Most ER and EER-oriented design methodologies recommend representing certain multiple object-set structures by a single relation. For example, the EER-oriented methodology of [24] employs a single relation-scheme for representing a binary *many-to-one* relationship-set and the entity-set involved in that relationship-set with the *many* cardinality. General merging techniques are explored in [20], where it is shown that merging requires the enforcement of additional null constraints. In [20] we have shown that multiple object-sets can be represented by a single relation-scheme not only for the standard binary many-to-one relationship-set construct, but for more complex EER structures as well. In this paper we have presented a merging procedure that is restricted to cases that do not require null constraints more complex than nulls-not-allowed constraints.

The translation process described in this paper can be simplified by employing *surrogate* [7] attributes as primary and foreign key attributes, as shown in [19]. This approach ensures an improved object identification in relational databases, a simplified maintenance of referential integrity constraints, and allows adding, removing or renaming attributes without affecting the specification of referential integrity constraints.

Some or all the procedures presented in this paper can be easily integrated into a single, concise, translation procedure. This approach is desirable when no changes to the integrated procedures are anticipated. Alternatively, these procedures can be employed as independent steps in translating EER schemas into relational schemas. Such an approach allows easy modifications and extensions of the translation procedures. We have taken a combination of both approaches in implementing a database *Schema Definition* and *Translation* (*SDT*) tool [22], where the translation of EER schemas into canonical relational schemas and the normalization procedure have been integrated, while the name assignment and merging have been left as independent procedures. Thus, *SDT* supports several interchangeable name assignment and merging algorithms, and can be readily extended by adding new name assignment algorithms and new (more powerful) merging techniques.

SDT generates abstract (DBMS-independent) relational schemas from EER schemas, and relational DBMS schemas from abstract relational schemas. For carrying out accurately the later stage, the target relational DBMS must have mechanisms for maintaining key dependencies, referential integrity constraints, and nulls-not-allowed constraints. Note that referential integrity mechanisms are not provided by all commercial DBMSs. Moreover, when provided, these mechanisms are conceptually different, varying from support for non-procedural constraint specifications, such as in IBM's DB2, to support for procedural constraint specifications, such as in SYBASE and in INGRES (see [21] for a discussion on the referential integrity mechanisms of DB2, SYBASE, and INGRES). Currently, *SDT* targets INFORMIX, SYBASE, and INGRES DBMSs; for SYBASE and INGRES it generates the procedures required for maintaining the referential integrity constraints. *SDT* can be coupled with a graphical editor for defining EER schemas [26]. By insulating users from the technical details of the translation and from the complexities and peculiarities of specific DBMSs, *SDT* significantly simplifies and increases the productivity of the database design process[†].

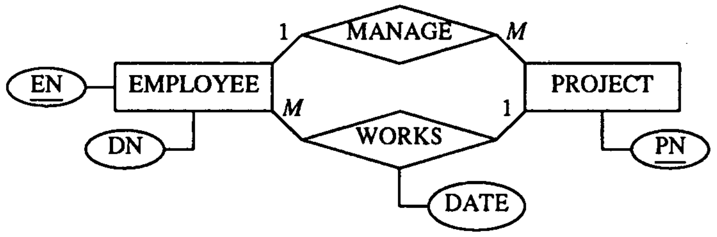
The formalism developed in this paper can be used in examining the equivalence of different EER schemas. This problem is addressed in [11] where two ER schemas are said to be equivalent if their relational schema representations are equivalent. However, the equivalence of relational schemas defined in [11] does not take into account inclusion dependencies and null constraints, and is based on the stringent *Pure Universal Instance Assumption.* Consider the following example adapted from [11]. Following [11], the ER schemas of figures 1.1(i) and 9.1(i) are equivalent because the relational schema of figure 9.1(ii) is equivalent, in the sense of [11], to the relational schema of figure 1.1(iii). According to our correctness criteria, the relational schema of figure 9.1(ii) represents incorrectly the ER schema of figure

[†]Based on experimental data on using *SDT* in developing databases for the the *Human Genome* and the *Superconducting Super Collider* projects at several DOE laboratories.

9.1(i); a correct representation for this schema is shown in figure 9.1(iii). Since the relational schema of figure 1.1(iii) is also an incorrect representation of the ER schema of figure 1.1(i), the relationship above is certainly not an equivalence relationship. Indeed, the correct representations for the ER schemas of figures 1.1(i) and 9.1(i) are not equivalent in the sense defined in this paper. The open question is whether two different EER schemas can be non trivially (i.e. without considering renamings) equivalent.

We have not considered in this paper all the constructs and constraints proposed for various versions of the EER model. Thus, we have not considered EER multi-valued attributes and the mandatory/optional constraints regarding the involvement of entity-sets in relationship-sets. We believe, however, that our approach can be applied to richer versions of the EER model as well. Finally, restrictions such as the uniqueness of entity-identifiers and not allowing null values for EER attributes, have been made for the sake of simplicity and can be removed; however, removing these restrictions require extending the procedures presented in this paper.

(i)



(ii) EMPLOYEE ($\underline{EN}$, DN)
WORKS ($\underline{\overline{EN}}$, PN, DATE)
PROJECT ($\underline{PN}$, MN)

(iii) EMPLOYEE ($\underline{EN}$, DN)
WORKS ($\underline{\overline{EN}}$, PN, DATE)
PROJECT ($\underline{PN}$, MN)
WORKS [EN] $\subseteq$ EMPLOYEE [EN]
WORKS [PN] $\subseteq$ PROJECT [PN]
PROJECT [MN] $\subseteq$ EMPLOYEE [EN]
EMPLOYEE : $\emptyset \overset{EX}{\rightarrow}$ EN, DN
WORKS : $\emptyset \overset{EX}{\rightarrow}$ EN, PN, DATE
PROJECT : $\emptyset \overset{EX}{\rightarrow}$ PN

Abbreviations : EN=EMPLOYEE NAME, DN=DEPARTMENT NUMBER, MN=MANAGER NAME, PN=PROJECT NUMBER

Figure 9.1 Relational Representations for an Entity-Relationship Schema.

ACKNOWLEDGEMENTS

The authors want to express their appreciation to the referees of *ACM Transactions on Database Systems* for their many and excellent critiques and suggestions that played an important role in restructuring this paper, as well as in improving its understandability. We also want to acknowledge the high quality implementation of *SDT* carried out by Weiping Fang, as well as his comments on the procedures underlying *SDT*.

REFERENCES

[1] P. Atzeni and E.P.F. Chan, "Independent database schemes under functional and inclusion dependencies", *Proc. of the 13th VLDB Conf.*, 1987, pp. 159-166.

[2] P. Atzeni and D.S. Parker, "Assumptions in relational database theory", *ACM Symposium on Principles of Database Systems*, 1982, pp. 1-9.

[3] P. Atzeni and D.S. Parker, "Formal properties of net-based knowledge representation schemes", *Proc. of 2nd IEEE Int. Conf. on Data Engineering*, 1986, pp. 700-706.

[4] L.I. Brady, "A universal relation assumption based on entities and relationships" *Proc. of the 4th Int. Conf. on Entity-Relationship Approach*, P.P. Chen (ed), IEEE Computer Society Press, 1985, pp. 208-215.

[5] M.A. Casanova, R. Fagin, and C.H. Papadimitriou, "Inclusion dependencies and their interaction with functional dependencies", *Journal of Computer and System Sciences* **28**,1 (Feb. 1984), pp. 29-59.

[6] P.P. Chen, "The entity-relationship model- towards a unified view of data", *ACM Trans. on Database Systems* **1**,1 (March 1976), pp. 9-36.

[7] E.F. Codd, "Extending the relational database model to capture more meaning", *ACM Trans. on Database Systems* **4**,4 (Dec 1979), pp. 397-434.

[8] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

[9] R. Hull, "Relative information capacity of simple relational database schemata", *Proc of Third ACM Symposium on Principles of Database Systems*, 1984, pp. 97-109.

[10] R. Hull and R. King, "Semantic database modeling: Survey, applications, and research issues", *Computing Surveys* **19**,3 (September 1987), pp. 201-260.

[11] S. Jajodia, P.A. Ng, and F.N. Springsteel, "The problem of equivalence for entity-relationship diagrams", *IEEE Trans. on Software Engineering* **SE-9**,5 (September 1983), pp. 617-630.

[12] S. Jajodia, P.A. Ng, and F.N. Springsteel, "Entity-relationship diagrams which are in BCNF", *International Journal of Computer and Information Sciences* **12**,4 (1983), pp. 269-283.

[13] Y.E. Lien, "On the semantics of the entity-relationship data model", *Entity-Relationship Approach to System Analysis and Design*, P.P. Chen (ed), North-Holland, 1980, pp. 155-167.

[14] Y.E. Lien, "On the equivalence of database models", *Journal of the ACM*, **29**,2 (April 1982), pp. 333-362.

[15] D. Maier, *The theory of relational databases*, Computer Science Press, 1983.

[16] D. Maier, D. Rozenshtein, and J. Stein, "Representing roles in universal scheme interfaces", *IEEE Trans. on Software Engineering* , vol SE-11,7, July 1985, pp. 644-652.

[17] D. Maier, D. Rozenshtein, and D.S. Warren, "Window functions", *Advances in Computing Research,* vol.3, JAI Press, 1986, pp. 213-246.

[18] D. Maier, J.D. Ullman, and M. Vardi, "On the foundations of the universal relation model", *ACM Trans. on Database Systems* **9**,2 (June 1984), pp. 283-308.

[19] V.M. Markowitz and J.A. Makowsky, "Identifying extended entity-relationship object structures in relational schemas", *IEEE Trans. on Software Engineering,* **16**, 8 (Aug. 1990), pp. 777-790.

[20] V.M. Markowitz, "Merging relations in relational databases", Technical Report LBL-27842, Lawrence Berkeley Laboratory, June 1991.

[21] V.M. Markowitz, "Problems underlying the use of referential integrity mechanisms in relational database management systems", *Proc. of the 7th IEEE Int. Conf. on Data Engineering*, 1991.

[22] V.M. Markowitz and W. Fang, "SDT 4.1. Reference Manual", Technical Report LBL-27843, Lawrence Berkeley Laboratory, May 1991.

[23] V.M. Markowitz and A. Shoshani, "Name assignment techniques for relational schemas representing extended entity-relationship structures", *Proc. of the 8th Int. Conf. on Entity-Relationship Approach*, Toronto, Canada, 1989.

[24] T.J. Teorey, D. Yang, and J.P. Fry, "A logical design methodology for relational databases using the extended entity-relationship model", *Computing Surveys* **18**,2 (June 1986), pp. 197-222.

[25] T.J. Teorey, *Database modeling and design. The entity-relationship approach*, Morgan Kaufmann Publishers, Inc., 1990.

[26] E. Szeto and V.M. Markowitz, "ERDRAW 2.2. Reference Manual", Technical Report LBL-PUB-3084, Lawrence Berkeley Laboratory, May 1991.