LBL-28799
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

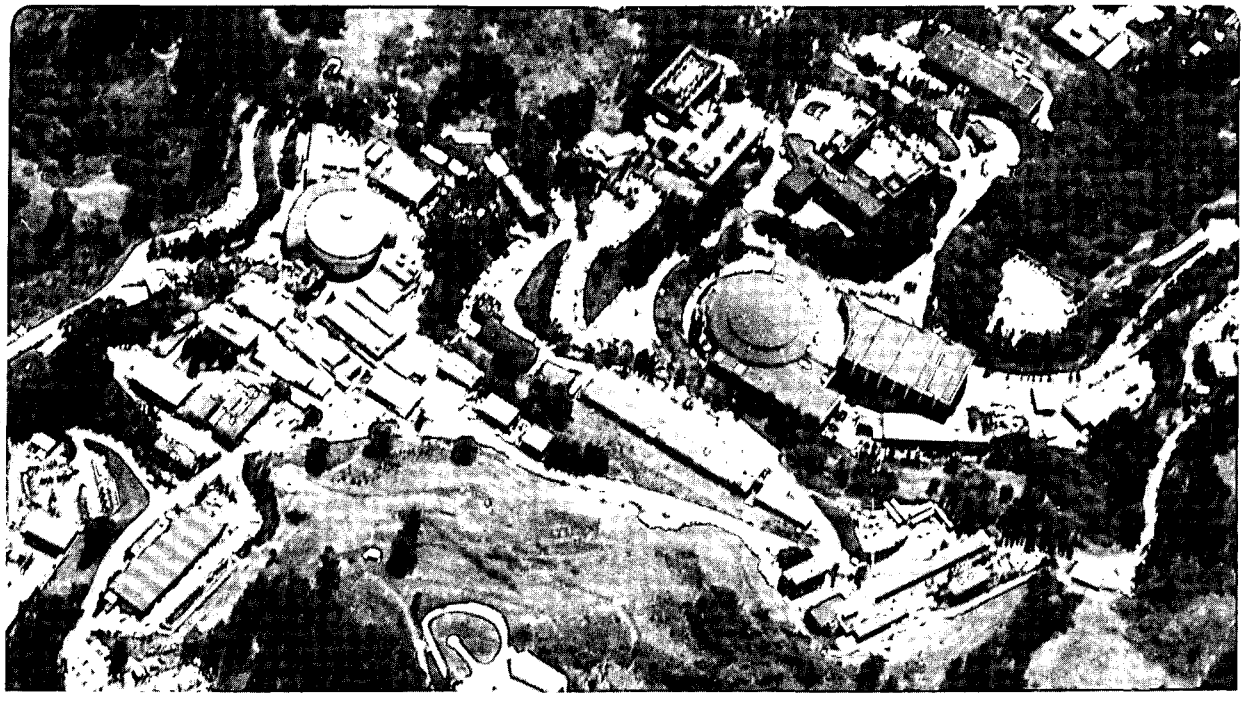
Information and Computing
Sciences Division

Submitted to IEEE Transactions on Software Engineering

Identifying Extended Entity-Relationship Object Structures in Relational Schemas

V.M. Markowitz and J.A. Makowsky

April 1990



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

DISCLAIMER

LBL-28799

IDENTIFYING EXTENDED ENTITY-RELATIONSHIP OBJECT STRUCTURES IN RELATIONAL SCHEMAS

Victor M. Markowitz

Computing Science Research & Development
Information & Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, California 94720

Johann A. Makowsky

Computer Science Department
Technion-Israel Institute of Technology
Haifa 3200, Israel

April 1990

To appear in IEEE Transactions on Software Engineering

This work was supported by the Director, Office of Energy Research, Applied Mathematical Sciences Research Program, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

IDENTIFYING EXTENDED ENTITY–RELATIONSHIP OBJECT STRUCTURES IN RELATIONAL SCHEMAS *

VICTOR M. MARKOWITZ † and JOHANN A. MAKOWSKY ‡

† Computer Science Research Department
Lawrence Berkeley Laboratory
Berkeley, CA 94720

‡ Computer Science Department
Technion- Israel Institute of Technology
Technion City, Haifa 3200

Abstract — **One of the main goals of database design is to capture the structural semantics of information systems. This goal can be achieved by using an object-oriented data model for specifying the structure of an information system in terms of objects and object connections, and then translating such a specification into a relational database definition (schema). In previous papers we have investigated the problem of representing object structures using relational constructs in the context of an Extended Entity-Relationship (EER) model. In this paper we address the reverse problem, namely of identifying EER object structures in relational schemas. We consider relational schemas consisting of relation-schemes, key dependencies, and key-based inclusion dependencies (referential integrity constraints). Schemas of this form are said to be EER-convertible if they can be associated with an EER schema. A procedure that determines whether a relational schema is EER-convertible is developed. We propose a normal form for relational schemas representing EER object structures. For EER-convertible relational schemas we present the corresponding normalization procedure. The procedures presented in this paper can be used for analyzing the semantics of existing relational databases, and for converting relational database schemas into object-oriented database schemas.**

Index Terms — **database design, extended entity-relationship model, key dependency, inclusion dependency, normal form, referential integrity, relational model, schema translation.**

I. INTRODUCTION.

Database design involves describing the structure of an information system, that is, specifying the relevant objects and object connections. The relational model was an important step in providing a simple, easy to use data model for this purpose with its easy to understand table structure, the relation. In a relational database the information on objects is kept in relations. Regarding object connections, there are two approaches to their relational representation. In the

* To appear in IEEE Transactions on Software Engineering; also issued as technical report LBL-28799.
This work was supported in part by the Applied Mathematical Sciences Research Program, of the Office of Energy Research, U.S. Department of Energy, under Contract DE-AC03-76SF00098.
† Email address: *V_Markowitz@lbl.gov*
‡ Email address: *janos@TECHSEL.BITNET*

Universal Relation (UR) approach [16], which lead to various *normalization* techniques, object connections are implied by the involvement of the same attributes in different relations. The UR approach is underlied by a set of assumptions that define the meaning of relations sharing common attributes. These assumptions make the description of object connections extremely difficult and not entirely reliable, mainly because they require an excessively complex attribute naming.

A different approach is to use *referential integrity* constraints to represent object connections[6]. A referential integrity constraint connects two relations by associating a *foreign-key* in one relation with the *primary-key* in the other relation. Referential integrity is easier to use and understand than the UR assumptions. However, the referential integrity concept is still surrounded by confusion, as illustrated by the successive modifications of the original definition of [6], and by the unclear semantics of certain referential integrity structures [23].

The difficulties encountered by the relational model in providing a suitable framework for describing information systems lead to the development of object-oriented (semantic) data models that allow users to describe their applications directly in terms of objects and object connections (cf. [11]). One model that has gained in popularity is the *Entity–Relationship* (ER) model [4] and its various extensions, such as the *Extended ER* (EER) model ([19], [24]). The ER concepts reflect a natural, although limited, view of the world: entities are qualified by their attributes and entity connections are expressed by relationships. The EER model includes, in addition to the basic ER constructs, the generalization and full aggregation abstraction capabilities usually found in object-oriented data models. The ER and EER models have been mostly accepted as an early stage database design tool. Once the design ends, the ER or EER schema is informally translated into a relational schema, and its role is therewith ended ([4], [24], [25]).

In [19] and [20] we have proposed to take this translation more seriously, and capture all the semantics of object structure specifications. In [19] we have shown that EER object structures can be represented accurately using relation-schemes, functional dependencies, and inclusion dependencies. We have shown that an EER schema can be always represented by a relational schema in *Boyce-Codd Normal Form* (BCNF), which involves only key dependencies and key-based inclusion dependencies (referential integrity constraints) [19]. The definition of such schemas is supported by many commercially available relational database management systems (RDBMS), such as IBM's DB2, SYBASE 4.0, and INGRES 6.3.

In this paper we address the problem of identifying EER object structures in relational schemas of the form mentioned above. Such schemas are said to be *EER-convertible* if they can be associated with an EER schema. Deriving the object-oriented (e.g. EER) description associated

with an existing database is essential in understanding the semantics of database applications, and in coping with the difficulty of specifying referential integrity constraints.

Our work is a continuation of [18]. Prior to [18] mappings of ER and EER schemas into relational schemas were presented in numerous papers (e.g. [4], [5], [12], [15], [24]); these mappings are mainly informal (see [19] for a detailed discussion), and do not attempt to characterize the relational schema translations of ER and EER schemas. The absence of such a characterization makes the reverse mappings presented in [3], [8], and [13], partial and mostly informal. Our approach is to first analyze the relational schema translations of EER schemas. Next, we examine the effect of merging relation-schemes in such schemas. Unlike [3] and [18], we do not rely on any convenient well behavior of relational schemas, in order to encompass an as large as possible class of relational schemas. Thus, we do not assume that relation-schemes represent single object-sets or that relational attribute names convey any structural information.

We develop in this paper a procedure for translating relational schemas into EER schemas. This procedure first transforms relational schemas into a form appropriate for identifying EER object structures. Thus, certain cyclic inclusion dependency structures are detected and removed, and certain relation mergings are reversed. Next, every relation-scheme is mapped into an object-set. The type of every object-set (e.g. weak entity, specialization entity) and the type of object-set connections (e.g. relationship-set—entity-set, specialization entity-set—generic entity-set) are derived by examining the set of inclusion dependencies and the structure of the keys in each relation-scheme. We discuss the significance of failing to derive an EER schema for a relational schema. Possible extensions of this procedure are also discussed.

We contrast the EER-oriented design of relational databases with relational normalization. We show that the assumptions underlying normalization make the normalization techniques difficult to use and unreliable. Conversely, EER-oriented design favors the realization of many normalization goals, by facilitating the task of representing separately independent object-sets. We show that the degree of normalization ensured by EER-oriented design depends on the set of dependencies considered. If only dependencies directly derivable from EER schemas are considered, then an EER schema can be represented by a BCNF relational schema [19]. Note that this result contradicts [12], and is different from similar results in [5] (for ER schemas) and in [15] (for a different version of EER schemas), where inclusion dependencies are not taken into account. Some authors (e.g. [5], [15], [24]) argue that the specification of additional functional dependencies should be allowed, and recommend using relational normalization as a complement of ER and EER-oriented design. Such a combination, however, is not simple and could lead to

erroneous designs, as shown in [19].

For EER-convertible relational schemas we propose an *EER Object Structure Normal Form* (EER/OSNF), which involves using *surrogate* attributes as primary and foreign key attributes. Our definition of EER/OSNF extends a similar definition in [3]. Note that different ER normal forms have been defined in [5] and [15]; however, the ER normal form of [5] does not take into account the inter-relation constraints which are essential for an accurate representation of ER and EER schemas, and the ER normal form of [15] regards EER schemas, rather than the relational translations of EER schemas. We discuss the desirability of EER/OSNF schemas, and present a normalization procedure for EER-convertible relational schemas. EER/OSNF 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 way relations are inter-related, that is, without affecting the referential integrity constraints.

The rest of the paper is organized as follows. In section 2 we review briefly the graph and relational concepts used in this paper. Section 3 contains a brief review of the EER model. In section 4 we examine the properties of relational schema translations of EER schemas. In section 5 we compare the EER-oriented design of relational databases with relational normalization. In section 6 we develop a procedure for mapping relational schemas into EER schemas. In section 7 we propose a normal form for relational schemas representing EER object structures, and present a normalization procedure for EER-convertible relational schemas. We conclude the paper with a summary.

II. Preliminary Definitions and Notations

We use in this paper some basic graph-theoretical concepts. Any textbook on graph theory (e.g. [9]) 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. 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, consists of the vertices of V', and of the edges of H that connect only (are incident to) vertices of V'.

We briefly review below the basic relational concepts used in this paper. Details can be found in any textbook (e.g. [25]) for the basic concepts, and in [7] for the theory of 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. We denote by t a tuple, and by $t[W]$ the sub-tuple of t corresponding to the

attribute set W. A null value is denoted ω, and a tuple consisting of k null values is denoted ω^k. A tuple is said to be *total* iff it has only non-null values.

A *relational schema* is a pair (R, Δ), where R is a set of relation-schemes and Δ is a set of dependencies over R. We consider relational schemas which are associated with $\Delta = F \cup I$, where F and I denote sets of functional and inclusion dependencies, respectively. 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. Every attribute is assigned a *domain*, and every relation-scheme, $R_i(X_i)$, is assigned a *relation* (value), r_i. Two attributes are said to be *compatible* if they are associated with the same domain, and attribute sets X and Y are said to be *compatible* iff there exists a one-to-one correspondence of compatible attributes between X and 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 is equal to $\{t[W_i] \mid t \in r_i\}$. The *total projection* of r_i on W is denoted $\pi\downarrow_W(r_i)$, and is equal to $\{t[W_i] \mid t \in r_i$ and t is total$\}$.

Let $R_i(X_i)$ be a relation-scheme associated with relation r_i, let W be a subset of X_i, and let Y be an attribute set that is compatible with W. *Renaming* W to Y in r_i is denoted *rename* $(r_i; W \leftarrow Y)$, 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 $t'[Y] = t[W] \}$.

Let $R_i(X_i)$ and $R_j(X_j)$ be two relation-schemes associated with relations r_i and r_j, respectively; let Y and Z be two compatible and disjoint subsets of X_i and X_j, respectively; 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 $(Y = Z)$ is denoted $r_i \underset{Y=Z}{\bowtie} r_j$, and is equal to $\{t \mid t[X_i] \in r_i, t[X_j] \in r_j$, and $t[Y] = t[Z]\}$. The *outer-equi-join* of r_i and r_j on $(Y = Z)$ is denoted $r_i \overset{\circ}{\underset{Y=Z}{\bowtie}} r_j$, and is equal to the union of three relations, r_1, r_2, and r_3, where: $r_1 = r_i \underset{Y=Z}{\bowtie} r_j$, $r_2 = \{t \mid t[X_i] = \omega^{k_i},\ t[X_j] \in r_j$, and $\nexists\, t' \in r_i$ s.t. $t'[Y] = t[Z]\}$, 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. $t[Y] = t''[Z]\}$.

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: $Y \rightarrow Z$, where Y and Z are subsets of X_i; R_i: $Y \rightarrow Z$ is *satisfied* by r_i iff for any two tuples of r_i, t and t', $t[Y] = t'[Y]$ implies $t[Z] = t'[Z]$. A *key* associated with R_i is a subset of X_i, K_i, such that $R_i : K_i \rightarrow X_i$ is satisfied by any r_i associated with R_i and there does not exist any proper subset of K_i having this property. A relation-scheme can be associated with several *candidate keys* from which one *primary key* is chosen. If all functional dependencies associated with a relation-scheme, R_i, involve in their left-hand sides

supersets of keys, then R_i is said to be in *Boyce-Codd Normal Form* (BCNF).

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 $R_i[Y] \subseteq R_j[Z]$, where Y and Z are compatible subsets of X_i and X_j, respectively; $R_i[Y] \subseteq R_j[Z]$ is *satisfied* by r_i and r_j iff $\pi_Y(r_i) \subseteq \pi_Z(r_j)$. 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 $R_i[Y] \subseteq R_j[Z] \in I$. A set of inclusion dependencies I is said to be *acyclic* iff G_I is acyclic. If $R_i[Y] \subseteq R_j[Z]$ is an inclusion dependency and Z is the primary key of R_j then $R_i[Y] \subseteq R_j[Z]$ is said to be *key–based*, and Y is called a *foreign key* in R_i. Key-based inclusion dependencies are called *referential integrity constraints* [6].

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* [10]. We are interested only in relational schemas that preserve the attribute values. This requirement is captured by the information-capacity equivalence defined below, which follows the definition of *generic* equivalence of [10].

Definition 2.1 (*Information–Capacity Equivalence*). Let RS and RS' be two relational schemas. RS' is said to *dominate* RS if 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 and ϕ' is the identity on the set of all consistent states of RS;

(4) For any database state r associated with RS, ϕ commutes with each permutation of the values in the domains of r; similarly, for any database state r' associated with RS', ϕ' commutes with any permutation of the values in the domains of r'.

RS and RS' are said to be *equivalent* if RS dominates RS' and RS' dominates RS. ■

Informally, a schema RS' dominates another schema RS if RS' can be associated with more database states then RS, that is, while every legal database state associated with RS can be exactly reconstructed from its mapping into a database state of RS', some database states associated with RS' cannot be exactly reconstructed from their mappings into database states of RS. Condition (4) above ensures that attribute values are preserved by the state mappings.

III. THE EXTENDED ENTITY-RELATIONSHIP MODEL

The concepts of the *Entity-Relationship* (ER) model, (*entity*, *relationship*, *entity—set*, *relationship—set*, *value—set*, *attribute*, *entity—identifier*, *weak* entity-set, relationship *cardinality*, *role*) have been repeatedly reviewed (e.g. [24]) since their original definition in [4], and are well known. We refer commonly to entities and relationships as *objects*. We review briefly in this section only the additional abstraction mechanisms of the *Extended* ER (EER) model considered in this paper. We also discuss briefly the issue of well-definedness of EER structures; a detailed discussion can be found in [21].

The EER model includes, in addition to the basic ER constructs, the generalization and full aggregation capabilities. *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 are then associated only with the generic entity-set. The inverse of generalization is called *specialization*. An entity-set which is not specified as the specialization of any other entity-set is called *generalization—source*. An entity-set that is neither weak nor the specialization of other entity-sets, is called an *independent* entity-set. For the sake of simplicity we do not distinguish in this paper between different kinds of generalizations such as those described in [11]. In the ER model the *aggregation* construct takes three forms: (i) the aggregation of a collection of attributes into an entity-set; (ii) the aggregation of a collection of attributes and the entity-identifiers of several existing entity-sets into a weak entity-set; and (iii) the aggregation of two or more entity-sets into a relationship-set. The full capability of aggregation is provided in the EER model by simply allowing relationship-sets to associate any object-set, rather than only entity-sets.

EER-schemas are expressible in a diagrammatic form called *EER diagram*. In an EER diagram entity-sets, relationship-sets, and attributes, are represented by rectangle, diamond, and ellipse shaped vertices, respectively. Every vertex is labeled by the name of the corresponding object-set or attribute. The EER diagram is a *directed graph*. The directionality of edges allows the explicit representation not only of the interaction of the 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 (either 1 (*one*) or M (*many*)), (ii) from weak entity-sets to the entity-sets on which they depend for identification, labeled *ID*; (iii) from specialization entity-sets to the corresponding generic entity-sets, labeled *ISA*; and (iv) from object-sets to their associated attributes. A self-explanatory example of an EER diagram is shown in figure 3.1.

EER structures must satisfy certain restrictions concerning the combination of different EER constructs. These restrictions refer mainly to the specification of generalization structures and to the interaction of generalization with aggregation. EER structures that satisfy these restrictions are said to be *well—defined*. A detailed discussion can be found in [21] (see also [11] for a related discussion). The first restriction disallows *directed* cycles in EER diagrams. Second, specialization entity-sets are restricted to have unique generalization-sources. The third restriction concerns the interaction of generalization and aggregation, namely that an entity-set cannot be specified by using both generalization and aggregation. Finally, in an EER structure object-sets must have unique (*global*) names, all the attributes of an object-set must have unique (*local*) names among the attributes associated with that object-set, and the roles must have unique names among the multiple roles of some object-set in another object-set. The restrictions above are summarized in the following definition.

Definition 3.1.

An EER schema is said to be *well—defined* if it satisfies the following conditions:

(1) the EER schema is associated with an acyclic EER diagram;

(2) every attribute is associated with exactly one object-set;

(3) every relationship-set associates at least two object-sets;

(4) if an entity-set is a *generalization—source* then it has a non-empty *entity—identifier*;

if an entity-set is a *specialization* then it has an empty entity-identifier, it has a unique generalization-source, and it is not directly dependent for identification on other entity-sets;

(5) *Name Uniqueness* must be satisfied for: the set of object-sets; the set of attributes associated with an object-set; and the set of roles of some object-set in another object-set. ■

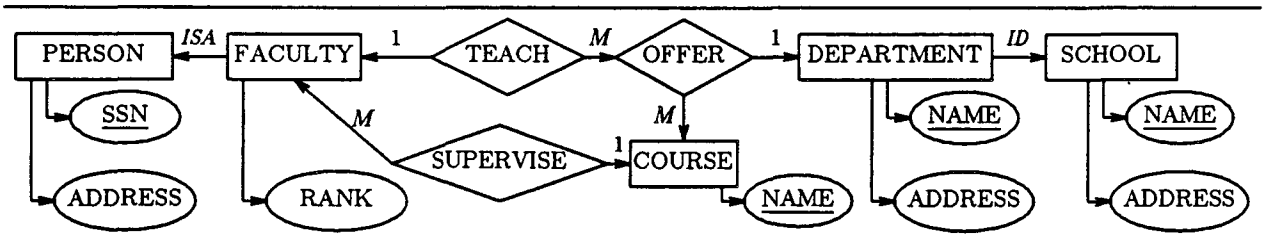


Figure 3.1 Extended Entity-Relationship Diagram Example (*identifiers are underlined*).

IV. MAPPING EER SCHEMAS INTO RELATIONAL SCHEMAS

In [21] we have shown that mapping EER schemas into relational schemas involves the following aspects: (i) representing EER object structures using relational constructs, (ii) normalizing relational schema translations of EER schemas, (iii) assigning names to relational attributes; and (iv) merging relation-schemes in relational schema translations of EER schemas. In [19] we have examined the conditions for a *correct* relational schema representation of an EER schema, and the conditions required in order to apply normalization on such schemas. In [19] we have proposed a *canonical* schema representation for EER schemas, and a normalization mapping of such schemas into *Boyce-Codd Normal Form* (BCNF) schemas, both satisfying the conditions mentioned above. In order to keep the relational schema translations of EER schemas independent of a specific name assignment for relational attributes, we employed only (internal) symbolic names for relational attributes (e.g. A_{3_2} represents the second attribute of the third relation). Techniques for assigning names to attributes in relational schema translations of EER schemas have been examined in [20].

4.1 Mapping EER Schemas into Normalized Relational Schemas.

The mapping procedure defined below, ***Rmap***, is the result of integrating the two mappings mentioned above. ***Rmap*** is exemplified in figure 4.1, which shows the result of mapping the EER schema of figure 3.1. ***Rmap*** maps EER schemas into BCNF relational schemas of the form $(R, F \cup I)$, where R, F, and I, denote sets of relation-schemes, functional dependencies, and inclusion dependencies, respectively, such that every relation-scheme of R corresponds to a unique object-set, the functional dependencies of F are key dependencies, and the inclusion dependencies of I are key-based, that is, are referential integrity constraints.

Object–Set	Relation–Scheme (keys are underlined)	Attribute : ER Attribute		Inclusion Dependencies	
PERSON	$R_1(\underline{A_{1_1}, A_{1_2}})$	A_{1_1} : SSN	A_{1_2} : ADDRESS	$R_4[A_{4_2}]$	$\subseteq R_1[A_{1_1}]$
DEPARTMENT	$R_2(\underline{A_{2_1}, A_{2_3}}, A_{2_2})$	A_{2_1} : NAME	A_{2_2} : ADDRESS	$R_5[A_{5_1}A_{5_2}]$	$\subseteq R_2[A_{2_1}A_{2_3}]$
COURSE	$R_3(\underline{A_{3_1}})$	A_{3_1} : NAME		$R_5[A_{5_3}]$	$\subseteq R_3[A_{3_1}]$
FACULTY	$R_4(\underline{A_{4_1}}, A_{4_2})$	A_{4_1} : RANK		$R_6[A_{6_1}]$	$\subseteq R_4[A_{4_2}]$
OFFER	$R_5(A_{5_1}, \underline{A_{5_2}, A_{5_3}})$			$R_6[A_{6_2}]$	$\subseteq R_5[A_{5_3}]$
TEACH	$R_6(A_{6_1}, \underline{A_{6_2}})$			$R_7[A_{7_1}]$	$\subseteq R_4[A_{4_2}]$
SUPERVISE	$R_7(\underline{A_{7_1}}, \underline{A_{7_2}})$			$R_7[A_{7_2}]$	$\subseteq R_3[A_{3_1}]$
SCHOOL	$R_8(\underline{A_{8_1}}, A_{8_2})$	A_{8_1} : NAME	A_{8_2} : ADDRESS	$R_2[A_{2_3}]$	$\subseteq R_8[A_{8_1}]$

Figure 4.1 Relational Schema Corresponding to the EER Schema of Figure 3.1.

Definition 4.1 — *Rmap* .

Input: a well-defined EER schema;

Output: a relational schema of the form (R, $F \cup I$).

1. *Value—Sets*. Every value-set is mapped into a relational domain.

2. *Independent Entity-Sets*. An independent entity-set, E_i, is mapped into a relation-scheme, $R_i(X_i)$, such that: X_i is in a one-to-one correspondence with the EER attributes of E_i; every attribute A of X_i is assigned the domain corresponding to the value-set of the EER attribute of E_i corresponding to A. The subset Z_i of X_i that is in a one-to-one correspondence with the identifier of E_i, is specified as the primary-key of R_i, and key dependency $R_i : Z_i \rightarrow X_i$ is added to F.

3. *Aggregation Object—Sets*. Let object-set O_i be the aggregation of (not necessarily distinct) object-sets O_{i_j}, $1 \leq j \leq m$, and let object-sets O_{i_j} correspond to relation-scheme $R_{i_j}(Y_{i_j})$, $1 \leq j \leq m$, respectively. Then O_i is mapped into relation-scheme $R_i(X_i)$, and inclusion dependencies $R_i[FK_{i_j}] \subseteq R_{i_j}[K_{i_j}]$, $1 \leq j \leq m$, are added to I. X_i is the union of two disjoint sets of attributes, X'_i and X''_i, such that: (i) X'_i is in a one-to-one correspondence with the EER attributes of O_i, where the correspondence is specified as in (2) above; (ii) $X''_i = \bigcup_{j=1}^{m} FK_{i_j}$, is a set of foreign-key attributes, where every foreign-key FK_{i_j} is in a one-to-one correspondence with primary-key K_{i_j}, $1 \leq j \leq m$, such that every attribute A of FK_{i_j} is assigned the domain associated with attribute B of K_{i_j} corresponding to A.

   If O_i is a *weak entity-set* and Z_i is the subset of X_i which is in a one-to-one correspondence with the identifier of E_i, then $Z_i X''_i$ is specified as the primary-key of R_i, and key dependency $R_i : Z_i X''_i \rightarrow X_i$ is added to F . If O_i is a *relationship—set* then if all the cardinalities of the object-sets involved in O_i are *many*, then X''_i is specified as the primary-key of R_i, and key dependency $R_i : X''_i \rightarrow X_i$ is added to F; else $(X''_i - FK_{i_k})$ is specified as the primary-key of R_i, where FK_{i_k} is the *foreign—key* referencing the relation-scheme corresponding to an object-set that has cardinality *one* in O_i, and for every object-set O_{i_j} which has cardinality *one* in O_i, key dependency $R_i : (X''_i - FK_{i_j}) \rightarrow FK_{i_j}$ is added to F.

4. *Specialization Entity—Sets*. Let entity-set E_i be the specialization of entity-sets E_{i_j} , $1 \leq j \leq m$, and E_k be the (unique) generalization-source of E_i. Let E_k correspond to relation-scheme $R_k(Y_k)$ and entity-sets E_{i_j} correspond to relation-schemes $R_{i_j}(Y_{i_j})$, $1 \leq j \leq m$,

respectively. Then E_i is mapped into relation-scheme $R_i(X_i)$, and inclusion dependencies $R_i[FK_{i_j}] \subseteq R_{i_j}[K_{i_j}]$ $1 \leq j \leq m$, are added to I. X_i is the union of two disjoint sets of attributes, X'_i and X''_i, such that: (i) X'_i is in a one-to-one correspondence with the EER attributes of E_i, where the correspondence is specified as in (2) above; (ii) X''_i is in a one-to-one correspondence with the primary-key of R_k, where the correspondence is specified as in (3.ii) above; and (iii) every foreign-key FK_{i_j}, $1 \leq j \leq m$, is equal to X''_i. X''_i is specified as the primary-key of R_i, and key dependency $R_i : X''_i \rightarrow X_i$ is added to F. ■

The following proposition characterizes the relational schema translation of an EER schema, generated by ***Rmap***.

Proposition 4.1. Let $RS = (R, F \cup I)$ be a relational schema generated by ***Rmap*** from an EER schema. Let relation-scheme $R_i(X_i)$ of R correspond to object-set O_i, and let FK_i denote the union of all foreign-keys, FK_{i_j}, associated with R_i. Then (i) I is acyclic; and

(ii) Every foreign-key FK_{i_j} associated with R_i satisfies the following conditions:

(a) the attributes of FK_{i_j} are not allowed to have null values;

(b) FK_{i_j} is either included in, or disjoint with, the primary-key of R_i;

(c) FK_{i_j} is either equal to, or disjoint with, any other foreign-key of R_i; and

(d) FK_{i_j} is involved in an inclusion dependency corresponding to the interaction of O_i with another object-set, O_j, such that

- O_i is a weak entity-set (*ID*) dependent on entity-set O_j iff $FK_{i_j} \subsetneq K_i$ and $K_i \not\subseteq FK_i$;
- O_i is a specialization entity-set of entity-set O_j iff $FK_{i_j} = FK_i = K_i$;
- O_i is a relationship-set involving object-set O_j with cardinality M iff $K_i \subseteq FK_i$, for every candidate key CK_i of R_i (including K_i) $FK_{i_j} \subseteq CK_i$, and at least one of these inclusions is proper;
- O_i is a relationship-set involving object-set O_j with cardinality 1 iff $K_i \subsetneq FK_i$ and there exists at least one candidate key CK_i of R_i, such that FK_{i_j} and CK_i are disjoint.

Proof Sketch. (i) Let G_{ER} be the EER diagram associated with the EER schema mapped by ***Rmap***. First, we prove that the inclusion dependency graph associated with RS, G_I, is isomorphic to the subgraph of G_{ER} induced by the vertices corresponding to the object-sets: the relation-schemes of R are in a one-to-one correspondence with the set of vertices representing

object-sets in G_{ER}, and the inclusion dependencies of I are in a one-to-one correspondence with the set of edges of G_{ER} connecting vertices representing object-sets. Consequently, G_I and the subgraph of G_{ER} induced by the object-set vertices are isomorphic. Since G_{ER} is acyclic, the isomorphism above implies that G_I, and therefore I, is acyclic. For (ii) the proof follows the specification of ***Rmap***. ■

As an illustration of the proposition above, consider the foreign-keys associated with the relation-schemes of the relational schema in figure 4.1. The translation of EER schemas into relational schemas is not unique. Thus, EER schemas can be represented by unnormalized relational schemas, or by relational schemas in which relation-schemes correspond to multiple, rather than unique, object-sets (see below). However, all the relational schema translations of an EER schema must have equivalent information-capacity [19].

4.2 Merging Relation–Schemes.

Now we shall briefly examine the issue of merging relation-schemes in relational schema translations of EER schemas. Merging brings about the need to allow certain attributes to have special purpose *null* values. The various meanings associated with nulls are generally compressed into two: *inapplicable* values and *unknown* (but applicable) values. We have shown in [21] that nulls representing *inapplicable* values are not needed in relational databases associated with EER schemas. In such databases *unknown* null values can represent either an unknown EER attribute value or an unknown relational foreign-key attribute value. For simplicity, we assume below that nulls are employed to represent only unknown foreign-key values, while unknown EER attribute values are represented by other special values, rather than nulls.

A procedure for merging relations in relational databases that preserves the information-capacity and normal form of the corresponding schemas, is presented in [22]. We have shown in [22] that merging relations requires the introduction of additional *null constraints* for restricting the way in which null values appear in merged relations. We discuss below a restricted version of merging that involves only simple null constraints disallowing attributes to have null values; such constraints have the advantage of being directly supported by all relational database management systems. We use the following notation: if a relational attribute, A, is allowed to have null values then it is denoted A^{null}, otherwise A is considered, by default, to be disallowed to have null values.

A restricted version of the procedure proposed in [22] for merging relation-schemes in relational schemas, called ***Rmerge***, is given below. Given a schema generated by ***Rmap***, $RS = (R, F \cup I)$, and a subset $\overline{R}$ of R, such that the primary-keys associated with the

relation-schemes of $\overline{R}$ are pairwise compatible, ***Rmerge*** maps RS into a new relational schema, $RS' = (R', F' \cup I')$, where R' results by replacing the relation-schemes of $\overline{R}$ with a new relation-scheme, R_m, and F' and I' consist of adjusted key and inclusion dependencies, respectively. Merging is achieved by outer-joining the relations associated with the relation-schemes of $\overline{R}$, so that the relation associated with R_m, r_m, includes tuples corresponding to every object represented in the merged relations. The dependencies associated with R_m ensure that the relations involved in merging can be always reconstructed from r_m, so that the schema generated by ***Rmerge***, RS', has equivalent information-capacity with RS. The foreign-key attributes of R_m that are not included in the primary-key associated with R_m, are allowed to have null values. An example of applying ***Rmerge*** is shown in figure 4.2, where ***Rmerge*** is applied on the relational schema of figure 4.1 in order to merge relation-schemes R_4 and R_7 into R'_4, and relation-schemes R_5 and R_6 into R'_5.

Definition 4.2 — *Rmerge* .

Input : a relational schema $RS = (R, F \cup I)$, and a subset of R, $\overline{R}$, such that

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

(ii) there exists a relation-scheme $R_p(X_p)$ in $\overline{R}$ that satisfies the following condition:
for every relation-scheme R_i of $\overline{R}$, $i \neq p$, $R_i[K_i] \subseteq R_p[K_p] \in I$;

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

a. $| X_i - K_i | = 1$;

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_p, R_i can be involved in the left-hand side of at most one additional inclusion dependency, of the form $R_i[X_i - K_i] \subseteq R_j[K_j]$.

Output: a relational schema RS' of the form $(R', F' \cup I')$.

Rmerge $(\overline{R})$ applied on RS generates $RS' = (R', I' \cup F')$ as follows:

1. R' results by replacing in R the relation-schemes of $\overline{R}$ with a new relation-scheme, $R_m(X_m)$, such that $K_m := K_p$, $X_m := K_m \bigcup_{R_i(X_i) \in \overline{R}} (X_i - K_i)$, where the attributes of $(X_m - X_p)$ are allowed to have null (unknown reference) values;

2. F' results by replacing all the key dependencies involving primary-keys associated with the relation-schemes of $\overline{R}$, with key dependency $R_m : K_m \rightarrow X_m$;

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 $\overline{R}$.

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

η is the *identity* for every relation of r associated with a relation-scheme of $(R - \overline{R})$; and maps the set of relations $\{r_i \mid r_i \in r,\ r_i$ is associated with relation-scheme R_i of $\overline{R}\}$ into r_m as follows: (a) $r_m := r_p$; (b) <u>for</u> each R_i of $(\overline{R} - \{R_o\})$ <u>do</u> $r_m := \pi_{(X_m - K_i)}(r_m \overset{\circ}{\bowtie}_{K_m = K_i} r_i)$;

η' is the *identity* for every relation of $(r' - r'_m)$; and maps relation r'_m of r' into relations $\tilde{r_i} := rename(\pi\downarrow_{K_m(X_i - K_i)}(r'_m), K_m \leftarrow K_i)$, where $R_i(X_i)$ is a relation-scheme of $\overline{R}$. ■

Intuitively, ***Rmerge*** merges relation-schemes corresponding to multiple object-sets structures consisting of:

M1. 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 entity-sets that are not weak and have single-attribute identifiers; and

M2. an entity-set E_i and its specialization entity-sets, provided that these specialization entity-sets (a) have no specializations and are directly generalized only by E_i, (b) are not involved in relationship-sets or weak entity-sets, (c) have exactly one attribute.

Consider, for example, the EER schema of figure 3.1. Entity-set COURSE and relationship-set OFFER satisfy conditions (M1.a) and (M1.c), but do not satisfy condition (M1.b). Similarly, entity-sets PERSON and FACULTY satisfy conditions (M2.a) and (M2.c), but do not satisfy condition (M1.b). Conversely, entity-set FACULTY and relationship-set SUPERVISE, respectively relationship-set OFFER and relationship-set TEACH, satisfy conditions (M1) and (M2). In [22] we have shown that only the multiple object-sets structures defined above can be represented by a

Relation—Scheme (keys are underlined)	**Inclusion Dependencies**
$R_1(\underline{A_{1_1}, A_{1_2}})$	$R_4[A_{4_2}] \subseteq R_1[A_{1_1}]$
$R_2(\underline{A_{2_1}, A_{2_3}}, A_{2_2})$	$R_5[A_{5_1}A_{5_2}] \subseteq R_2[A_{2_1}A_{2_3}]$
$R_3(\underline{A_{3_1}})$	$R_4[A_{7_2}] \subseteq R_3[A_{3_1}]$
$R'_4(\underline{A_{4_2}}, A_{4_1}, A_{7_2}^{null})$	$R_5[A_{6_1}] \subseteq R_4[A_{4_2}]$
$R'_5(\underline{A_{5_3}}, A_{5_1}, A_{5_2}, A_{6_1}^{null})$	$R_5[A_{5_3}] \subseteq R_3[A_{3_1}]$
$R_8(\underline{A_{8_1}}, A_{8_2})$	$R_2[A_{2_3}] \subseteq R_8[A_{8_1}]$

Figure 4.2 The Relational Schema of Figure 4.1 after Merging.

single relation involving the simple null constraints mentioned above.

Merging affects the foreign-key structures in some relation-schemes. This effect of merging is captured by the following proposition.

Proposition 4.2. Let $RS = (R, F \cup I)$ be a relational schema generated by ***Rmap***, and let $RS' = (R', F' \cup I')$ be the result of applying ***Rmerge*** on RS. Then in every relation-scheme $R'_i(X'_i)$ of R', every foreign-key of R'_i, FK'_{i_j}, satisfies either condition (ii) of proposition 4.1, or FK'_{i_j} satisfies the following condition: FK'_{i_j} consists of a single attribute, A, such that A is allowed to have null values, and A does not appear in any other foreign or primary key of R'_i.

Proof Sketch. The proof follows the specification of ***Rmerge***. ■

As an illustration of the proposition above, consider foreign-key A_{6_1} and A_{7_2} associated with relation-schemes R'_5 and R'_4, respectively, of the relational schema of figure 4.2.

V. EER-ORIENTED DESIGN VS RELATIONAL NORMALIZATION

Relational normalization and EER-oriented design are two basic approaches to relational database design. In this section we contrast these two approaches.

5.1 Relational Normalization.

Relational normalization assumes that all the semantics are captured by dependencies expressed over a universal set of attributes, and that users do not need to be aware of how the attributes are grouped into relations. Normalization is underlied by certain (*Universal–Relation*) assumptions (cf. [16]) concerning the semantics conveyed by the names of relational attributes. Normalization procedures are based on the specification of a set of dependencies which are converted automatically into a normalized schema. However, specifying properly the set of dependencies is an extremely difficult task; thus, overlooking a single dependency or deviating, even slightly, from the assumptions underlying normalization, could radically, and unreasonably, change the result of normalization, that is, the organization of attributes into relation-schemes. Furthermore, even a correct dependency specification cannot guarantee the generation of an intuitively meaningful relational schema. A functional dependency, $X \rightarrow Y$, besides saying that every X can be associated with a unique Y, also intends to specify the *relationship* between X and Y [25]. Yet, functional dependencies may represent not the presence or absence of such relationships between attributes, but rather a constraint with no influence on the way data should be structured. The following well known example [25] shows that normalization does not necessarily result in a schema that reflects our intuition about how information should be organized in

relations. Consider the set of functional dependencies shown in figure 5.1(i). A possible result of normalization is the relational schema shown in figure 5.1(ii) [25]. It is obvious that it would make more sense to keep information about the courses that students take at given hours (*CHS*) instead of the rooms in which students can be found at given hours, as represented by relation-scheme R_4 of figure 5.1(ii).

The core of the problem, as noted in [1], is the impossibility of deciding automatically which dependency describes a relationship and which does not. Because of this limitation, normalization procedures do not necessarily separate object-sets that could be perceived as independent. For example consider the merging of relation-schemes associated with identical or equivalent keys involved in some normalization procedures [25]: equivalent keys may indicate the presence of multiple attributes of an object-set, that should be grouped together in the same relation-scheme, but may also represent independent object-sets which should be represented by separate relation-schemes. The issue of merging relations in relational schemas is discussed in more detail in [22].

5.2 EER–Oriented Design of Relational Schemas.

It is generally agreed that the concepts of objects and their properties are natural in designing databases. The EER object concepts (entities and relationships) correspond to the structures naturally occurring in information systems, thus enhancing the ability of designers to describe accurately database applications. The EER-oriented design of relational schemas is an object-oriented alternative to the data-oriented relational normalization, that simplifies the design process by reducing the number of elements under consideration (objects instead of attributes). The main cause for lack of normalization is *embedding* data about independent object-sets into one relation. EER-oriented design favors the realization of many normalization goals, by simplifying

(i) Functional Dependencies	Information Represented
$C \rightarrow T$	- each course has only one teacher;
$HR \rightarrow C$	- only one course can meet in a room at one time
$HT \rightarrow R$	- a teacher can be only in one room at one time
$CS \rightarrow G$	- each student has only one grade in each course
$HS \rightarrow R$	- a student can be only in one room at one time
(ii) Normalized Relation-Schemes	
R_1 (CSG)	- grades for students in courses
R_2 (CT)	- teacher of each course
R_3 (HRC)	- hours and rooms where each course meet
R_4 (HRS)	- rooms in which students can be found at given hours

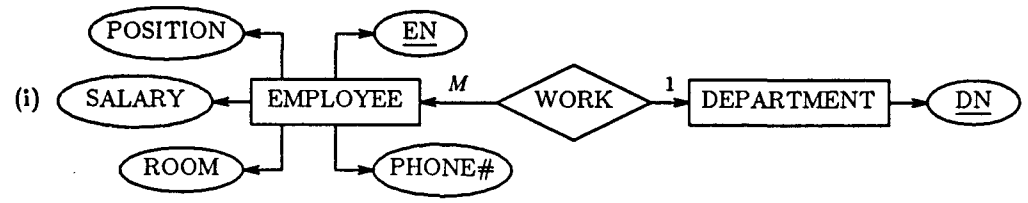
Abbreviations: C=COURSE, G=GRADE, H=HOUR, R=ROOM, S=STUDENT, T=TEACHER

Figure 5.1 A Normalization Example.

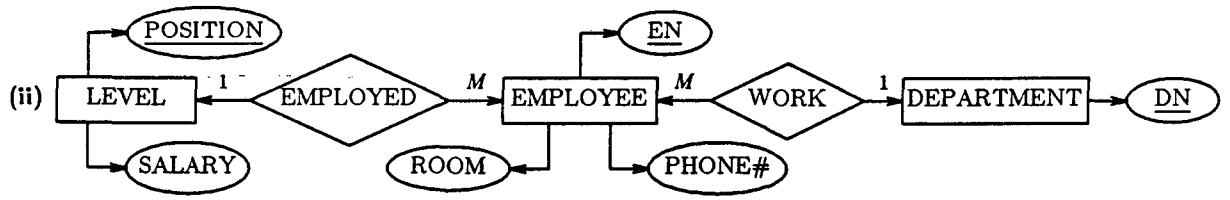
and facilitating the task of representing separately independent object-sets.

The degree of relational normalization ensured by EER-oriented design depends on the set of dependencies taken into account. The dependencies directly derivable from the EER schema are the functional dependencies of either non-key attributes on key attributes, or among primary and foreign key attributes, and inclusion dependencies representing EER object-set interactions. We have shown in [19] that for this set of dependencies an (unrestricted) well-defined EER schema can be represented by a BCNF relational schema. Note that this result contradicts [12] (see [19] for details), and is different from similar results in [5] (for ER schemas) and in [15] (for a different version of EER schemas), where inclusion dependencies are not taken into account. Some authors (e.g. [5], [15], [24]) allow the specification of additional functional dependencies between non-key attributes, and propose to use relational normalization as a complement of the EER-oriented design ([5], [24]). Applying normalization on relational schema translations of EER schemas, however, is not simple and could lead to erroneous designs if the assumptions underlying normalization are not taken into account [19]. Moreover, relational normalization must be extended in order to include inter-relation (e.g. inclusion) dependencies.

The need for specifying a functional dependency between non-key attributes can indicate the existence of an embedded object-set. The specification of a new object-set, however, cannot be done automatically since it must agree with a certain perception of the application. Consider the following functional dependency specified for relation-scheme EMPLOYEE of figure 5.2(i):



EMPLOYEE (EN, POSITION, SALARY), WORK (EN, DN), DEPARTMENT (DN)



LEVEL (POSITION, SALARY), EMPLOYED(EN, POSITION),
EMPLOYEE(EN, PHONE#, ROOM), WORK(EN, DN), DEPARTMENT(DN)

Note : Inclusion dependencies are not specified; keys and identifiers are underlined.

Figure 5.2 Relational Schemas and their Associated EER Schemas.

POSITION→SALARY. This functional dependency can be derived from the EER schema only if POSITION is converted into an entity-set, and a new relationship-set associating POSITION with EMPLOYEE is specified, as shown in figure 5.2(ii). Such a conversion has to agree with the designer's view of POSITION, since the functional dependency above can be also perceived as an incidental (numerical), rather than structural, constraint. Consider an additional functional dependency specified for relation-scheme EMPLOYEE of figure 5.2(i): PHONE#→ROOM; this dependency causes relation-scheme EMPLOYEE to be only in second normal form. Here decomposing relation-scheme EMPLOYEE in order to achieve a higher normal form is important only from a data representation point of view, and should be transparent to users.

Normalization disregards the inter-relation constraints necessary for representing object interactions. This deficiency of normalization has been addressed in [17], where a new design methodology has been proposed as an alternative to normalization. The design methodology of [17] takes into account inclusion dependencies in addition to functional dependencies, and is an interactive, rather than automatic, process. The inclusion dependency properties pursued by the methodology of [17] (acyclicity and key basing) result directly from mapping EER schemas into relational schemas, as shown in section 4. Moreover, EER-oriented design allows the expression of such desirable inclusion dependencies naturally, as EER existence constraints inherent to the EER constructs, without requiring the iterative and interactive process of [17].

VI. Mapping Relational Schemas into EER Schemas

In this section we investigate the problem of identifying EER object structures in relational schemas, in order to derive an EER schema from a given relational schema. We consider relational schemas involving key dependencies and key-based inclusion dependencies (referential integrity constraints).

We have shown in section 4 that EER schemas can be mapped (by ***Rmap***) into relational schemas of the form $(R, F \cup I)$, where R, F, and I, denote sets of relation-schemes, key dependencies, and key-based inclusion dependencies, respectively, and where every relation-scheme corresponds to a unique object-set. The properties of such relational schemas were captured in proposition 4.1, and are used below for the identification of EER object structures in relational schemas. First, we present some relational schema transformations involved in the derivation of EER schemas from relational schemas.

6.1 Relational Schema Transformations.

The relational schema translation of an EER schema in which every object-set corresponds to a unique relation, involves an acyclic set of inclusion dependencies (proposition 4.1(i)). The transformation specified below is adapted from [3], and removes certain cyclic inclusion dependency structures (i.e. inclusion dependencies that correspond to a directed cycle in the inclusion dependency graph).

Definition 6.1 — *Fold*.

Input: a relational schema $RS = (R, F \cup I)$, and a subset of I, $\overline{I}$, such that the inclusion dependencies of $\overline{I}$ form a cycle, and involve only foreign-keys that are equal to primary-keys;

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

Let $\overline{R}$ denote the set of relation-schemes involved in the inclusion dependencies of $\overline{I}$; the relation-schemes of $\overline{R}$ are associated with pairwise compatible primary-keys.

Fold $(\overline{I})$ applied on RS generates $RS' = (R', I' \cup F')$ as follows:

- R' results by replacing in R the relation-schemes of $\overline{R}$ with a relation-scheme, $R_0(X_0')$, where $R_0(X_0)$ is a relation-scheme of $\overline{R}$, and $X_0' := K_0 \bigcup_{R_i(X_i) \in \overline{R}} (X_i - K_i)$;
- F' results by replacing all the key dependencies involving primary-keys associated with the relation-schemes of $\overline{R}$, with key dependency $R_0 : K_0 \rightarrow X_0'$;
- I' results by removing from I the inclusion dependencies of $\overline{I}$, and by replacing R_i with R_0 and K_i with K_0, in every inclusion dependency of $(I - \overline{I})$ that involves a relation-scheme R_i of $\overline{R}$. ■

A simple example of folding is shown in figure 6.1(ii). The folding transformation defined above is similar to mapping ***Rmerge*** defined in section 4, and can be associated with the *state mappings* involved in the definition of ***Rmerge***. The proof that folding preserves the information-capacity of the relational schemas on which it is applied, is straightforward (see also [3]).

In section 4 we have shown that certain relation-schemes can be merged without affecting the normal form and information-capacity of relational schemas. Such mergings generate relation-schemes that correspond to multiple, rather than one, EER object-sets. In order to identify the object-sets corresponding to a relation-scheme, mergings are reversed by the transformation specified below.

Definition 6.2 — *Split*.

Input: a relational schema $RS = (R, F \cup I)$, a relation-scheme $R_i(X_i)$ of R, and a foreign-key of R_i, FK_{i_j}, consisting of a single attribute, A, such that A is allowed to have null values, and A does not appear in any foreign or primary key of R_i;

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

Split (R_i, A) applied on RS generates $RS' = (R', I' \cup F')$ as follows:

R' results by removing A from the attribute-set of R_i, and by defining a new relation-scheme $R_s(X_s)$, such that $X_s = A\ K_s$, where K_s is in a one-to-one correspondence with K_i, and the attributes of X_s are not allowed to have null values;

F' results by replacing in F the key dependency over R_i with key dependency $R_i : K_i \rightarrow (X_i - A)$, and by adding to F key dependency $R_s : K_s \rightarrow X_s$;

I' results by replacing in I the inclusion dependency $R_i[A] \subseteq R_j[K_j]$, by inclusion dependencies $R_s[K_s] \subseteq R_i[K_i]$ and $R_s[A] \subseteq R_j[K_j]$. ■

The split transformation defined above is the reverse of mapping ***Rmerge*** defined in section 4, and it can be associated with the two *state mappings* (in reverse order) defined for ***Rmerge***. The proof that this transformation is information-capacity preserving follows from the analogous property of ***Rmerge***. Similar to folding, split can also remove cyclic inclusion dependency structures. For example, consider the relational schema of figure 6.1(ii), where inclusion dependencies $R_1[D] \subseteq R_3[A]$ and $R_3[A] \subseteq R_1[A]$ form a cycle; this cycle is removed by ***Split***(R_1, D) (see the

Transformation	Relational Schema
(i) ***Input***	$R = \{R_1(\underline{A}, D^{null}, G),\ R_2(\underline{AB}),\ R_3(\underline{A}),\ R_4(\underline{ABC}),\ R_5(\underline{A}, H)\}$ $I = \{ R_1[D] \subseteq R_3[A],\ R_1[A] \subseteq R_5[A],\ R_2[A] \subseteq R_1[A],\ R_3[A] \subseteq R_1[A],$ $R_4[C] \subseteq R_1[A],\ R_4[AB] \subseteq R_2[AB],\ R_4[A] \subseteq R_5[A],\ R_5[A] \subseteq R_1[A]\}$
(ii) ***Fold***	$R = \{R_1(\underline{A}, D^{null}, G, H),\ R_2(\underline{AB}),\ R_3(\underline{A}),\ R_4(\underline{ABC})\}$ $I = \{ R_1[D] \subseteq R_3[A],\ R_2[A] \subseteq R_1[A],\ R_3[A] \subseteq R_1[A],$ $R_4[C] \subseteq R_1[A],\ R_4[AB] \subseteq R_2[AB],\ R_4[A] \subseteq R_1[A]\}$
(iii) ***Split*** (R_1, D)	$R = \{R_1(\underline{A}, G, H),\ R_2(\underline{AB}),\ R_3(\underline{A}),\ R_4(\underline{ABC}),\ R_6(\underline{A}, D)\}$ $I = \{ R_2[A] \subseteq R_1[A],\ R_3[A] \subseteq R_1[A],\ R_4[A] \subseteq R_1[A],\ R_4[AB] \subseteq R_2[AB],$ $R_4[C] \subseteq R_1[A],\ R_6[A] \subseteq R_1[A],\ R_6[D] \subseteq R_3[A]\}$
(iv) ***Remove*** $R_4[A] \subseteq R_1[A]$	$R = \{R_1(\underline{A}, G, H),\ R_2(\underline{AB}),\ R_3(\underline{A}),\ R_4(\underline{ABC}),\ R_6(\underline{A}, D)\}$ $I = \{ R_2[A] \subseteq R_1[A],\ R_3[A] \subseteq R_1[A],\ R_4[AB] \subseteq R_2[AB],$ $R_4[C] \subseteq R_1[A],\ R_6[A] \subseteq R_1[A],\ R_6[D] \subseteq R_3[A]\}$

Figure 6.1 Relational Schema Transformation Examples.

relational schema of figure 6.1(iii)).

The last relational schema transformation involved in the derivation of EER schemas from relational schemas, consists of removing extraneous foreign-keys, which entails removing the inclusion dependencies involving such foreign-keys.

Definition 6.3 – *Remove*.

Let $RS = (R, F \cup I)$ be a relational schema, let $R_i(X_i)$ be a relation-scheme of R, and let FK_{i_j} and FK_{i_k} be two foreign-keys of R_i, involved in inclusion dependencies, $R_i[FK_{i_j}] \subseteq R_j[K_j]$ and $R_i[FK_{i_k}] \subseteq R_k[K_k]$, respectively, such that $FK_{i_j} \subseteq FK_{i_k}$. Let Z be the subset of K_k corresponding (via $R_i[FK_{i_k}] \subseteq R_k[K_k]$) to FK_{i_j}. FK_{i_j} is said to be *extraneous* iff inclusion dependency $R_k[Z] \subseteq R_j[K_j]$ is implied by $(I \cup F)$. An extraneous foreign-key FK_{i_j} is *removed* from R_i by removing inclusion dependency $R_i[FK_{i_j}] \subseteq R_j[K_j]$ from I. ■

For example, consider the relational schema of figure 6.1(iii); the foreign-key involved in inclusion dependency $R_4[A] \subseteq R_1[A]$ is extraneous, and therefore this inclusion dependency can be removed. Clearly, the removal of extraneous foreign-keys is an information-capacity preserving transformation.

6.2 Mapping Relational Schemas into EER Schemas.

Given a relational schema of the form $RS = (R, F \cup I)$ the procedure defined below first transforms RS into an information-capacity equivalent schema, RS', by applying repeatedly ***Fold*** and ***Split***, and by removing extraneous foreign-keys, and then derives a *candidate* EER schema from RS', and verifies its well-definedness.

Definition 6.4 – *Rmap*^{R}

Input: a relational schema RS of the form $(R, F \cup I)$, where

F is a set of key dependencies (involving primary and candidate keys), and

I is a set of key-based inclusion dependencies (referential integrity constraints);

Output: a well-defined EER schema or a ***fail*** message;

1. *Folding*. For every subset $\overline{I}$ of I, consisting of inclusion dependencies that satisfy the input conditions of definition 6.1, apply ***Fold*** $(\overline{I})$ on RS.
2. *Reverse Merging*. For every relation-scheme R_i of R and every foreign-key FK_{i_j} of R_i that satisfy the conditions of definition 6.2, apply ***Split*** (R_i, FK_{i_j}) on RS.

3. *Remove Extraneous Foreign—Keys.* For every relation-scheme R_i of R and every extraneous foreign-key FK_{i_j} of R_i, remove the inclusion dependency involving FK_{i_j}.

4. *Examine Transformed Relational Schema.* The relational schema $RS' = (R', F' \cup I')$ resulted from applying transformations (1) through (3) above on RS, is examined in order to detect whether it satisfies the properties for relational schema translations of EER schemas:

   (i) if I' is not acyclic then issue a **warning** message;

   (ii) For every relation-scheme $R'_i(X'_i)$ of R', and every foreign-key of R'_i, FK'_{i_j}, verify that:

   (a) the attributes of FK'_{i_j} are not allowed to have null values;

   (b) FK'_{i_j} is either included in, or disjoint with, the primary-key of R'_i;

   (c) FK'_{i_j} is either equal to, or disjoint with, any other foreign-key of R'_i;

   (d) if R'_i has foreign-keys, then K'_i includes at least one foreign-key;

   if any of the conditions above is not satisfied then issue a **warning** message.

5. *Inclusion Dependency Characterization.* Determine for every inclusion dependency of I' the corresponding type of object-interaction (see proposition 4.1(ii.d)).

6. *Derive Candidate EER Schema.* Derive a candidate EER schema from RS' as follows:

   Object—Sets Every relation-scheme $R'_i(X'_i)$ of R' is mapped into an object-set O_i of type

   Relationship—Set iff every inclusion dependency of I' that involves R'_i in its left-hand side, corresponds to the interaction of a relationship with an object-set;

   Entity—Set iff every inclusion dependency of I' that involves R'_i in its left-hand side, corresponds to the interaction of a (specialization or weak) entity-set with another entity-set;

   Undefined otherwise.

   Attributes Let O_i correspond to relation-scheme $R'_i(X'_i)$; O_i is associated with a set of attributes W_i, such that: (i) W_i is in a one-to-one correspondence with the set of non foreign-key attributes of R'_i (i.e. $(X'_i - FK'_i)$), and (ii) the subset of W_i corresponding to the set of primary-key attributes of R'_i that do not appear in any foreign-key of R'_i, (i.e. $(K'_i - FK'_i)$) constitutes the identifier of O_i.

   Connections Every inclusion dependency of I' of the form $R'_i[FK'_{i_j}] \subseteq R'_j[K'_j]$, is mapped into the connection of object-sets O_i and O_j, where O_i and O_j

correspond to R'_i and R'_j, respectively, and the type of the interaction (weak entity-set—entity-set, specialization entity-set—generic entity-set, relationship-set—object-set, or undefined) and cardinality (M or 1) are derived from the object interaction type associated with $R'_i[FK'_{i,j}] \subseteq R'_j[K'_j]$ in step (5) above.

7. *Examine Well—Definedness.* The well-definedness of the candidate EER schema derived in step (6) above is examined (that is, it is verified whether it satisfies the constraints of definition 3.1), and an appropriate message is issued. ■

For example, the results of various stages of applying ***Rmap***^{R} on the relational schema of figure 6.1(i) are shown in figures 6.1(ii), 6.1(iii), and 6.1(iv), and the well-defined EER schema eventually generated from the schema of figure 6.1(iv) is shown in figure 6.2. Relational schemas that fail to be mapped into EER schemas by ***Rmap***^{R} are shown in figure 6.3.

Extended Entity-Relationship (EER) *convertibility* for relational schemas is defined below.

Definition 6.5. A relational schema $RS = (R, F \cup I)$ is said to be *EER—convertible* if there exists a well-defined EER schema, ER, such that the relational schema generated by applying ***Rmap*** on ER has equivalent information-capacity with RS.

Proposition 6.1. Let $RS = (R, F \cup I)$ be a relational schema involving key dependencies and key-based inclusion dependencies. If mapping ***Rmap***^{R} applied on RS succeeds to generate a well-defined EER schema, then RS is an EER-convertible relational schema.

Proof Sketch. Let ER be the EER schema generated by ***Rmap***^{R} from RS. We must prove that ***Rmap*** applied on ER generates a relational schema, RS'', that has equivalent information-capacity with RS. Let RS' be the relational schema resulted from applying steps (1) through (3) of ***Rmap***^{R} on RS. Since these steps preserve the information-capacity of RS, it is sufficient to prove that RS'' has equivalent information-capacity with RS'. It can be verified that RS' and RS'' are isomorphic, with corresponding attributes and relation-schemes having assigned different names (note that ***Rmap*** assigns only internal names to relational attributes). ■

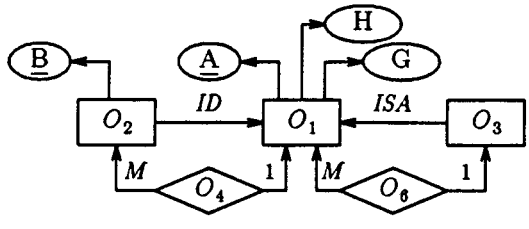


Figure 6.2 EER Schema Generated by ***Rmap***^{R} from Relational Schema of Figure 6.1(i).

Several remarks concerning ***Rmap***^{R} are in order:

(i) The ***Split*** transformation is based on proposition 4.2. The split transformation can be extended so that it will capture more powerful merging procedures such as those described in [22]; such an extension requires the introduction of special *null constraints* [22]. If the target of splitting is a relational schema that does not include such constraints, then an extended transformation could also detect the missing constraints. For example, an extended split transformation could be applied on the relational schema of figure 6.3(i), in order to detect that attributes B and C should be constrained to have both either null or non-null values in relations associated with R_1.

(ii) The set of dependencies in the relational schemas considered above consist of key dependencies and key-based inclusion dependencies. The implication problem for such sets of dependencies, involved in step (3) of ***Rmap***^{R}, is discussed in [2] and [7].

(iii) Steps (4) and (5) of ***Rmap***^{R} are based on proposition 4.1.

(iv) If the set of inclusion dependencies is not acyclic in step (4.i), then ***Rmap*** cannot generate a well-defined EER schema. However, it is preferable to continue the mapping in order to get a message expressed in terms of the EER structure, in step (7).

(v) If foreign-key attributes are allowed to have null values in step (4.ii.a) then a warning can be issued requiring a correction before going on.

(vi) If the foreign-key attributes of a relation-scheme R'_i do not satisfy the conditions specified in step (4.ii) then ***Rmap***^{R} will not be able to determine the EER type of the object-set in step (6) and R'_i will be mapped into an object-set of an *undefined* type. However, continuing the mapping can help the user in selecting the type for such object-sets, and then backtrack (using ***Rmap***) to find out what was wrong with the foreign-keys.

	Relational Schema	***Rmap* Step**	**Problem**
(i)	$R = \{R_1(\underline{A}, B^{null}, C^{null}),\ R_2(\underline{BC}, D)\}$ $I = \{R_1[BC] \subseteq R_2[BC]\}$	2	Missing Null Constraint
(ii)	$R = \{R_1(\underline{ABC}),\ R_2(\underline{AB}),\ R_3(\underline{AC}),\ R_4(\underline{A})\}$ $I = \{R_1[AB] \subseteq R_2[AB],\ R_1[AC] \subseteq R_3[AC]$ $R_2[A] \subseteq R_4[A],\ R_3[A] \subseteq R_1[A]\}$	4(ii.c)	$FK_{1_2} \cap FK_{1_3} \neq \emptyset$ and $FK_{1_2} \neq FK_{1_3}$
(iii)	$R = \{R_1(\underline{AB}, C),\ R_2(\underline{BC})\}$ $I = \{R_1[BC] \subseteq R_2[BC]\}$	4(ii.b)	$FK_{1_2} \not\subseteq K_1$ and $FK_{1_2} \cap K_1 \neq \emptyset$

Figure 6.3 Relational Schemas that cannot be mapped into EER Schemas.

(vii) The failure of $\boldsymbol{Rmap}^R$ can be caused by missing referential integrity constraints, or uncareful name assignments to relational attributes. While ***Rmap*** is independent of a specific name assignment for relational attributes, $\boldsymbol{Rmap}^R$ is applied on relational schemas whose attributes have assigned names. In [19] we have discussed the effect of name assignments on relational translations of EER schemas, and showed that an inaccurate assignment of names leads to relational schemas that are inconsistent with the semantics of the corresponding EER schemas. For example, in the relational schema of figure 6.3(ii) foreign-keys FK_{1_2} and FK_{1_3} both include an attribute named A, which may be the result of such an erroneous name assignment. If, for instance, A of FK_{1_2} is renamed D, then the relational schema of figure 6.3(ii) can be translated into an EER schema.

(viii) Note that $\boldsymbol{Rmap}^R$ applied on a relational schema generated by ***Rmap***, returns the EER schema input of ***Rmap***. However, if ***Rmap*** is applied on an EER schema generated by $\boldsymbol{Rmap}^R$ from a relational schema RS, then the result is a relational schema, RS', that has equivalent information-capacity with , but is not identical to, RS. Consequently, $\boldsymbol{Rmap}^R$ is only the *left inverse* of ***Rmap***. This asymmetry is caused by folding, splitting, and the removal of extraneous foreign-keys involved in $\boldsymbol{Rmap}^R$. While there are good reasons for merging relations (the reverse of splitting), there seems to be no reason for introducing extraneous foreign-keys, or for vertically splitting relations (the reverse of folding).

VII. AN OBJECT STRUCTURE NORMAL FORM FOR RELATIONAL SCHEMAS

Primary-keys in relational databases have two conflicting functions: to identify objects and to describe (i.e. represent attributes of) objects. As object identifiers, primary-keys cannot have null values, and updates of primary-key values cause both discontinuities in the identification of objects [14] and multiple tuple updates (i.e. have side-effects). As object descriptors, primary-key attributes should behave exactly as non primary-key attributes, that is, should be allowed to have null values, and their update should not affect either the identity or the interaction of objects represented in the database (i.e. should be side-effect free). In order to avoid such conflicting functions, Codd proposed in [6] to employ *surrogate* values as object identifiers (see also [14] for a discussion on the desirability of using surrogates). The adaptation of ***Rmap*** to generate primary-key surrogate attributes is straightforward and is specified below. An exemplification of this adapted mapping applied on the EER schema of figure 3.1, is shown in figure 7.1.

Definition 7.1 – *Smap*.

Input: a well-defined EER schema;

Output: a relational schema of the form $(R, F \cup I)$.

1. *Value–Sets*. Every value-set is mapped into a relational domain.

2. *Independent Entity–Sets*. An independent entity-set, E_i, is mapped into a relation-scheme, $R_i(\#_i\ X_i)$, where $\#_i$ is a surrogate attribute, and X_i is in a one-to-one correspondence with the EER attributes of E_i; every attribute A of X_i is assigned the domain corresponding to the value-set of the EER attribute of E_i corresponding to A. The primary-key of R_i consists of (surrogate) attribute $\#_i$, and key dependencies $R_i : \#_i \rightarrow X_i$ and $R_i : Z_i \rightarrow \#_i$ are added to F, where Z_i is the subset of X_i that is in a one-to-one correspondence with the identifier of E_i.

3. *Aggregation Object–Sets*. Let object-set O_i be the aggregation of object-sets O_{i_j}, $1 \leq j \leq m$, and let object-sets O_{i_j} correspond to relation-schemes $R_{i_j}(Y_{i_j})$, $1 \leq j \leq m$, respectively.

   Then O_i is mapped into relation-scheme $R_i(\#_i\ X_i)$, and inclusion dependencies $R_i[FK_{i_j}] \subseteq R_{i_j}[K_{i_j}]$, $1 \leq j \leq m$, are added to I; X_i is the union of two disjoint sets of attributes, X'_i and X''_i, such that: (i) X'_i is in a one-to-one correspondence with the EER attributes of O_i, where the correspondence is specified as in (2) above; (ii) $X''_i = \bigcup_{j=1}^{m} FK_{i_j}$, is a set of foreign-key (surrogate) attributes, where every FK_{i_j} corresponds to the (surrogate) primary-key of R_{i_j}, $1 \leq j \leq m$. The primary-key of R_i consists of $\#_i$, and key dependency $R_i : \#_i \rightarrow X_i$ is added to F.

   If O_i is a *weak entity-set* and Z_i is the subset of X_i which is in a one-to-one correspondence with the identifier of E_i, then key dependency $R_i : Z_i X''_i \rightarrow \#_i$ is added to F.

   If O_i is a *relationship–set* then if all the cardinalities of the object-sets involved in O_i are *many*, then the key dependency $R_i : X''_i \rightarrow \#_i$ is added to F; else for every object-set O_{i_j} which has cardinality *one* in O_i, key dependency $R_i : (X''_i - FK_{i_j}) \rightarrow \#_i\ FK_{i_j}$ is added to F.

4. *Specialization Entity–Sets*. Let entity-set E_i be the specialization of entity-sets E_{i_j}, $1 \leq j \leq m$, and E_k be the (unique) generalization-source of E_i. Let E_k correspond to relation-scheme $R_k(Y_k)$ and entity-sets E_{i_j} correspond to relation-schemes $R_{i_j}(Y_{i_j})$, $1 \leq j \leq m$, respectively. Then E_i is mapped into relation-scheme $R_i(X_i)$, and inclusion dependencies $R_i[FK_{i_j}] \subseteq R_{i_j}[K_{i_j}]$ $1 \leq j \leq m$, are added to I. X_i is the union of two disjoint sets of attributes, X'_i and X''_i, such that: (i) X'_i is in a one-to-one correspondence with the EER

attributes of E_i, where the correspondence is specified as in (2) above; (ii) X''_i corresponds to the primary-key of R_k; and (iii) every foreign-key FK_{i_j}, $1 \leq j \leq m$, is equal to X''_i. X''_i is the primary-key of R_i, and key dependency $R_i : X''_i \rightarrow X_i$ is added to F. ■

Like ***Rmap***, ***Smap*** generates BCNF relational schemas. However, the relational schemas generated by ***Smap*** provide a simpler and clearer representation of EER object structures, as discussed below.

Suppose that relational schema RS is generated by ***Rmap***, and that relation-scheme $R_i(X_i)$ corresponds to entity-set E_i. Changing the entity-identifier of E_i implies changes not only in X_i, but also in all the relation-schemes of RS that involve foreign-key attributes corresponding to the primary-key attributes of R_i. Moreover, such changes cause discontinuities in the identification of the objects represented in the relations associated with the relation-schemes affected by these changes [14]. In relational schemas generated by ***Smap*** entity-identifier changes affect only the relation-scheme corresponding to the entity-set whose entity-identifier is redefined, and have no effect on the identity of objects represented in the database.

In relational schemas generated by ***Rmap***, foreign-key attributes correspond to EER attributes. Such attributes need a special name assignment in order to avoid name conflicts [20]; and special procedures for preserving existing references between tuples (representing existing object interactions) when primary-key values are updated. In relational schemas generated by ***Smap***, foreign-key attributes are surrogate attributes, and therefore the name assignment problem becomes trivial, because only a single (rather than multiple) relational attribute corresponds to

Object–Set	Relation–Scheme (keys are underlined)	Attribute : ER Attribute		Inclusion Dependencies
PERSON	R_1 ($\underline{\#_1}$, A_{1_1}, A_{1_2})	A_{1_1} : SSN	A_{1_2} : ADDRESS	$R_4[A_{4_2}] \subseteq R_1[\#_1]$
DEPARTMENT	R_2 ($\underline{\#_2}$, A_{2_1}, A_{2_2}, A_{2_3})	A_{2_1} : NAME	A_{2_2} : ADDRESS	$R_5[A_{5_1}] \subseteq R_2[\#_2]$
COURSE	R_3 ($\underline{\#_3}$, A_{3_1})	A_{3_1} : NAME		$R_5[A_{5_2}] \subseteq R_3[\#_3]$
FACULTY	R_4 ($\underline{A_{4_2}}$, A_{4_1})	A_{4_1} : RANK		$R_6[A_{6_1}] \subseteq R_4[A_{4_2}]$
OFFER	R_5 ($\underline{\#_5}$, A_{5_1}, A_{5_2})			$R_6[A_{6_2}] \subseteq R_5[\#_5]$
TEACH	R_6 ($\underline{\#_6}$, A_{6_1}, A_{6_2})			$R_7[A_{7_1}] \subseteq R_4[A_{4_2}]$
SUPERVISE	R_7 ($\underline{\#_7}$, A_{7_1}, A_{7_2})			$R_7[A_{7_2}] \subseteq R_3[\#_3]$
SCHOOL	R_8 ($\underline{\#_8}$, A_{8_1}, A_{8_2})	A_{8_1} : NAME	A_{8_2} : ADDRESS	$R_2[A_{2_3}] \subseteq R_8[\#_8]$

Additional Key Dependencies

$R_1 : A_{1_1} \rightarrow \#_1$ $R_2 : A_{2_1}A_{2_3} \rightarrow \#_2$ $R_3 : A_{3_1} \rightarrow \#_3$ $R_5 : A_{5_2} \rightarrow \#_5\, A_{5_1}$

$R_6 : A_{6_2} \rightarrow \#_6\, A_{6_1}$ $R_7 : A_{7_1} \rightarrow \#_7\, A_{7_2}$ $R_8 : A_{8_1} \rightarrow \#_8$

Figure 7.1 EER/OSNF Relational Schema Corresponding to the EER Schema of Figure 3.1.

every EER attribute. Surrogate values carry no information, and users can cause their deletion or generation, but cannot update them [6]; consequently, no special update procedures must be specified for preserving references between tuples in databases associated with relational schemas generated by ***Smap***.

The primary and foreign keys in the relational schemas generated by ***Smap*** consist of single attributes. This simplifies the specification and maintenance of referential integrity constraints in relational database management systems.

We define below a new normal form for relational schemas representing EER object structures. Databases associated with relational schemas in this normal form have two desirable properties: association with an object-oriented description whose semantics facilitate interfacing with the database, and better object identification.

Definition 7.2 – *EER Object Structure Normal Form*.

A relational schema of the form $RS = (R, F \cup I)$ is said to be in *EER Object Structure Normal Form* (EER/OSNF) iff: (i) F is a set of key dependencies, (ii) I is a set of key-based inclusion dependencies (i.e. referential integrity constraints), (iii) primary and foreign key attributes are surrogate attributes, and (iv) RS is EER-convertible. ■

EER/OSNF can be adapted to other object-oriented data models, by replacing EER-convertibility in condition (iv) of definition 7.2, with convertibility to other object-oriented structures. Related normal forms have been defined in [3], [5] and [15]; the normal form of [3] is similar to our definition, but concerns only ER structures and does not involve the use of surrogate attributes; the normal form of [5] does not take into account the inter-relation constraints (essential for an accurate representation of ER and EER schemas), and the normal form of [15] regards EER schemas, rather than relational translations of EER schemas.

Clearly, ***Smap*** generates EER/OSNF schemas. The normalization procedure for an EER-convertible relational schema, RS, consists of applying ***Smap*** on the result of $\boldsymbol{Rmap}^R(RS)$. Note that $\boldsymbol{Rmap}^R$ can be easily extended in order to handle EER/OSNF relational schemas, and that merging can be applied on EER/OSNF schemas, but requires the introduction of null constraints such as those described in [22]. Regarding the RDBMS definition of relational schemas in EER/OSNF, only the latest release of INGRES (6.3) supports the specification of surrogate attributes, while referential integrity is supported by several systems, such as INGRES 6.3, SYBASE 4.0, and IBM's DB2.

VIII. SUMMARY

We have investigated the problem of identifying EER object structures corresponding to relational schemas consisting of relation-schemes, key dependencies, and key-based inclusion dependencies. Relational schemas for which such an identification succeeds are said to be *EER-convertible*. We have developed a procedure that derives EER schemas from relational schemas. This procedure can be extended by allowing relational schemas to involve null constraints in addition to key and inclusion dependencies. This procedure can be used both for deriving EER schemas from relational schemas, and for detecting and correcting relational constructs that prevent such derivations.

We have proposed a normal form for relational schemas representing EER object structures, the *EER Object Structure Normal Form* (EER/OSNF), which entails using only surrogate attributes in the definition of primary and foreign keys. For EER-convertible relational schemas we have presented a procedure for transforming EER-convertible schemas into EER/OSNF schemas. EER/OSNF ensures an improved object identification in relational databases, a simplified maintenance of referential integrity constraints, side-effect free attribute modifications, and allow the expression of concise queries [14]. The definition of EER/OSNF can be adapted to other object-oriented data models.

ACKNOWLEDGEMENT

The authors thank Nimrod Rotics for his contribution to [18], from which the present paper evolved, and the referees for their helpful comments.

REFERENCES

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

[2] M.A. Casanova and V.M.P. Vidal, "Towards a sound view integration methodology", in Proc. of the *Second ACM Symposium on Principles of Database Systems (PODS)*, pp. 36-47, 1983.

[3] M.A. Casanova and J.E. Amaral de Sa, "Mapping uninterpreted schemes into entity-relationship diagrams: two applications to conceptual schema design", *IBM Journal of Research and Development*, vol. 28, no. 1, pp. 82-94, January 1984.

[4] P.P. Chen, "The entity-relationship model- towards a unified view of data", *ACM Trans. on Database Systems*, vol. 1, no. 1, pp. 9-36, March 1976.

[5] I. Chung, F. Nakamura, and P.P. Chen, "A decomposition of relations using the entity-relationship approach", in *Entity-Relationship Approach to Information Modeling and Analysis*, P.P. Chen (ed),

Elsevier Science Publishers B.V., pp. 149-171, 1981.

[6] E.F. Codd, "Extending the relational database model to capture more meaning",*ACM Trans. on Database Systems* vol. 4, no. 4, pp. 397-434, December 1979.

[7] S.S. Cosmadakis and P.C. Kanellakis, "Equational Theories and Database Constraints", in Proc. of the *Seventeenth ACM Symposium on Theory of Computing (STOC)*, pp. 273-284, 1985.

[8] K.H. Davis and A.K. Arora, "Converting a relational database model into an entity-relationship model", in Proc. of the *Sixth International Conference on Entity-Relationship Approach*, S. March (ed), Elsevier Science Publishers B.V., pp. 271-285, 1987.

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

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

[11] R. Hull and R. King, "Semantic database modeling: Survey, applications, and research issues", *Computing Surveys* vol. 19, no. 3, pp. 201-260, September 1987.

[12] S. Jajodia, P.A. Ng, and F.N. Springsteel, "Entity-relationship diagrams which are in BCNF", *International Journal of Computer and Information Sciences*, vol. 12, no. 4, pp. 269-283, 1983.

[13] A. Klug, "Entity-relationship views over uninterpreted enterprise schemas", in *Entity-Relationship Approach to Systems Analysis and Design* , P.P. Chen (ed), Elsevier Science Publishers B.V., pp. 39-59, 1980.

[14] S.N. Khoshafian and G.P. Copeland, "Object identity", in Proc. of the *ACM Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, SIGPLAN Notices, vol. 21, no. 11, pp. 417-423, 1986.

[15] T.W. Ling, "A normal form for entity-relationship diagrams", in Proc. of the *Fourth International Conference on Entity-Relationship Approach*, P.P. Chen (ed), IEEE Computer Society Press, pp. 24-35, 1985

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

[17] H. Mannila and K-J. Raiha, "Inclusion dependencies in database design", in Proc. of the *Second Conference on Data Engineering*, pp. 713-718, 1986.

[18] J.A. Makowsky, V.M. Markowitz, and N. Rotics, "Entity-relationship consistency for relational schemas", *Lecture Notes in Computer Science* , vol.243, G. Ausiello and P. Atzeni (eds), Springer-Verlag, pp. 306-322, 1986.

[19] V.M. Markowitz and A. Shoshani, "On the correctness of representing extended entity-relationship structures in the relational model", in Proc. of the *1989 SIGMOD Conference*, pp. 430-439, 1989.

[20] V.M. Markowitz and A. Shoshani, "Name assignment techniques for relational schemas representing extended entity-relationship structures", in Proc. of the *Eighth International Conference on Entity-Relationship Approach*, Toronto, Canada, 1989.

[21] V.M. Markowitz and A. Shoshani, "Representing object structures in relational databases: A modular approach", submitted for publication, also appeared as Technical Report LBL-28482, March 1990.

[22] V.M. Markowitz, "Merging relations in relational databases", submitted for publication, also appeared as Technical Report LBL-27842, January 1990.

[23] V.M. Markowitz, "Referential integrity in relational databases", submitted for publication, also appeared as Technical Report LBL-27841, February 1990.

[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*, vol. 18, no. 2, pp. 197-222, June 1986.

[25] J.D. Ullman, *Principles of Database Systems*, Computer Science Press, 1982.

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
INFORMATION RESOURCES DEPARTMENT
1 CYCLOTRON ROAD
BERKELEY, CALIFORNIA 94720