Title
A Relation Merging Technique for Relational Databases

Author
Markowitz, V.M.

Publication Date
1992-02-01

LBL-27842

A Relation Merging Technique for Relational Databases*

Victor M. Markowitz†

Computer Science and Research
Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

February 1992

Published in the Proceedings of the Eighth International Conference on Data Engineering, 1992

* 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:(510) 486-683

LBL--27842

DE92 017120

A Relation Merging Technique for Relational Databases *

Victor M. Markowitz

Computer Science Research and Development
Information and Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road, Berkeley, CA 94720

Abstract

Relation merging is employed in relational databases in order to reduce the need for joining relations. Merging, however, can create unnormalized relations. In this paper we propose a merging technique that preserves the high (Boyce-Codd) normal form of relational schemas consisting of relation-schemes, key dependencies, referential integrity constraints, and null constraints. The additional constraints generated by this merging technique can be effectively maintained using the mechanisms provided by several relational database management systems.

1. Introduction

Relational schema design usually pursues the development of *normalized* schemas. Normalization leads to decreased data redundancy and therefore implies simpler procedures for maintaining database consistency and better update performance. The normalization process tends to increase the number of relations by splitting unnormalized relations into smaller, normalized, relations. Conversely, decreasing the number of relations in a database by *merging* relations reduces the need for joining relations, and usually results in a better access performance. The process of merging relations, however, may conflict with normalization by creating unnormalized relations. Ideally, the design process should result in a relational schema which is both normalized and has as few as possible relation-schemes. This goal is pursued by both normalization (e.g. see [1]) and *Entity–Relationship* oriented methodologies for designing relational schemas (e.g. see [14]). We examine briefly the shortcomings of the merging techniques involved in these methodologies.

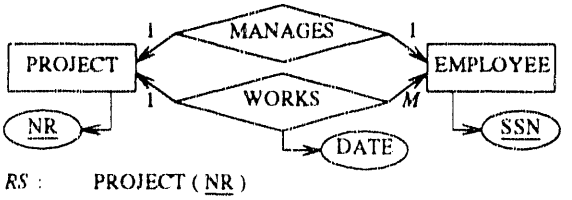
Relation merging was first used in *synthesis* normalization algorithms. Thus, the synthesis normalization algorithm presented in [1] involves a step of merging relations with *equivalent keys*. Consider, for example, two relation-schemes, TEACH (COURSE, FACULTY) and OFFER (COURSE, DEPARTMENT), both having COURSE as key. Following the synthesis algorithm of [1], these relation-schemes can be merged into a new relation-scheme whose key is also COURSE: ASSIGN (COURSE, FACULTY, DEPARTMENT). Suppose that the attributes of TEACH and OFFER are not allowed to have null values. Then ASSIGN has equivalent information-capacity [8] with TEACH and OFFER, only if attributes FACULTY and DEPARTMENT are allowed to have null values in ASSIGN, such that in every ASSIGN tuple at least one of these attributes has a non-null value. Such restrictions, defining the way in which nulls should appear in relations, were disregarded in the early normalization algorithms. These restrictions can be explicitly expressed using *null constraints* [9], or implicitly expressed using the *Universal Relation* assumptions [10].

An alternative approach to relational schema design has been proposed by the proponents of data models that have more semantic intuition, such as the *Entity-Relationship* (ER) [2] and *Extended Entity-Relationship* (EER) ([11], [14]) models. The ER and EER models are widely used for designing relational schemas: first, an ER or EER schema is specified, and then the ER or EER schema is translated into a relational schema. In [11] we have shown that ER and EER schemas can be represented by relational schemas in *Boyce-Codd Normal Form* (BCNF), consisting of relation-schemes, key dependencies, referential integrity constraints, and null constraints. For example, following [11], the ER schema of figure 1(i) is represented by the BCNF schema *RS* of figure 1(ii). Informally, object-sets are represented by relation-schemes, existence dependencies implied by object connections are represented by referential integrity constraints, and null-value restrictions on EER attributes are expressed by null (*nulls-not-allowed*) constraints. Regarding the goal of reducing the number of relations in a database, the question is whether a single relation-scheme can be used for representing multiple object-sets.

* Issued as technical report LBL-27842. 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.

Most ER and EER-oriented design methodologies recommend using a single relation-scheme for representing a binary *many-to-one* relationship-set and the entity-set involved in that relationship-set with a *many* cardinality. We have shown in [11] that methodologies such as that of [14] result in relational schemas that are inconsistent with the semantics of the corresponding ER or EER schema. Consider relational schema *RS'* of figure 1(iii), which, following [14], represents the ER schema of figure 1(i). Then, contrary to the semantics of the ER structure of figure 1(i), *RS'* allows a WORKS relation to include tuples representing employees having a non-null assignment DATE, even if these employees are not working on any PROJECT (i.e. with a null NR value). Consequently, additional constraints need to be specified in order to ensure that the database is consistent with the semantics of the corresponding ER schema. In relational schema *RS'*, for example, relational attribute DATE should be constrained to have a null value whenever attribute NR has a null value in a WORKS relation in order to represent accurately the association of ER attribute DATE with relationship-set WORKS; such restrictions can be expressed using *null constraints*.

In this paper we propose a relation merging technique for relational schemas consisting of relation-schemes, key dependencies, referential integrity constraints, and nulls-not-allowed constraints. We do not consider alternative relational representations based on the *Universal Relation* assumptions [10], because their capability of expressing constraints is limited, and because their underlying assumptions cannot be maintained using commercial relational database management systems (DBMS). We define a merging procedure that preserves the information-capacity and normal form of relational schemas. We show that in general merging requires the introduction of additional *null constraints* for restricting the way in which null values appear in merged relations. For example, relation-scheme WORKS of figure 1(iii) must be associated with null constraint DATE $\stackrel{Ex}{\rightarrow}$ NR (read 'non-null DATE requires non-null NR') in order to ensure that in a WORKS relation attribute DATE has a null value whenever attribute NR has a null value. It turns out that in certain cases only *nulls-not-allowed* constraints (i.e. constraints that restrict attributes to non-null values) are required. For example, relation-schemes EMPLOYEE and MANAGES of relational schema *RS* of figure 1(ii) can be merged into relation-scheme EMPLOYEE' (SSN, NR), where attribute SSN is not allowed to have null values, attribute NR is allowed to have null values, and no other null constraints need to be specified.



(ii) *RS* :
PROJECT (NR)
EMPLOYEE (SSN)
WORKS (SSN, NR, DATE*)
MANAGES (SSN, NR)
WORKS [NR] ⊆ PROJECT [NR]
WORKS [SSN] ⊆ EMPLOYEE [SSN]
MANAGES [NR] ⊆ PROJECT [NR]
MANAGES [SSN] ⊆ EMPLOYEE [SSN]

(iii) *RS'* :
PROJECT (NR)
WORKS (SSN, NR*, DATE*)
MANAGES (SSN, NR)

* : *Nulls Allowed*

Fig. 1. Relational Schemas Representing an ER Schema.

Employing effectively the merging procedure proposed in this paper depends on the capabilities of the underlying relational DBMS. An increasing number of commercial relational DBMSs support key dependencies, referential integrity constraints (which are key-based inclusion dependencies), and nulls-not-allowed constraints. For such systems we examine under what conditions the merging procedure does not require the introduction of additional constraints. In general, however, the merging technique presented in this paper may involve more complex null constraints and non key-based inclusion dependencies. Then the merging procedure can be employed only if the underlying DBMS provides a mechanism for maintaining such constraints and dependencies, such as the *triggers* mechanism of SYBASE 4.0 [13] and the *rules* mechanism of INGRES 6.3 [6].

As mentioned above, relational schemas consisting of relation-schemes, key dependencies, referential integrity constraints, and null constraints may represent EER schemas [11]. Applying the merging procedure developed in this paper on relational schema translations of EER schemas shows that multiple object-sets are amenable for representation by a single relation-scheme not only for the standard binary many-to-one relationship-set case, but for more complex structures as well.

The paper is organized as follows. In section 2 we introduce the relational concepts and notations used in this paper. The background for the merging technique is discussed in section 3. The merging technique is developed in section 4. In section 5 we discuss several aspects of applying our merging technique to schemas of databases developed using commercial relational DBMSs, and to relational schema translations of EER schemas. We conclude the paper with a summary.

2. Preliminary Definitions

We review briefly below the relational concepts used in this paper; details concerning these concepts can be found in textbooks such as [9].

A *relation-scheme* is a pair $R_i(X_i)$, where R_i is a relation-scheme name and X_i is a set of attributes. Every attribute is associated with a *domain*, and every relation-scheme is associated with a *relation* consisting of *tuples*.

We denote by t a tuple, and by $t[W]$ the subtuple of t corresponding to the attributes of W. A null value is denoted *null*, and a tuple consisting of k null values is denoted $null^k$. A tuple is said to be *total* iff it has only non-null values. 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 set of tuples $\{t[W_i] \mid t \in r_i\}$. The *total projection* of r_i on W is denoted $\pi\downarrow_W(r_i)$, and is equal to the subset of *total* tuples of $\pi_W(r_i)$. *Renaming* W in r_i to a set of attributes Y compatible with W is denoted *rename* $(r_i; W \leftarrow Y)$, and generates a relation associated with attribute set $(X_i - W)Y$, that is equal to set of tuples $\{t' \mid t \in r_i,\ t'[X_i - W] = t[X_i - W]$, 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 set of tuples $\{t \mid t[X_i] \in r_i,\ t[X_j] \in r_j$, and $t[Y] = t[Z]\}$. The *outer-equi-join* of r_i and r_j on $(Y = Z)$ is denoted $r_i \underset{Y=Z}{\overset{\Omega}{\bowtie}} r_j$, and is equal to the union of three relations, r_1, r_2, and r_3, where: $r_1 = r_i \underset{Y=Z}{\bowtie} r_j$, $r_2 = \{t \mid t[X_i] = null^{k_i},\ t[X_j] \in r_j$, and $\nexists t' \in r_i$ s.t. $t'[Y] = t[Z]\}$, and $r_3 = \{t \mid t[X_i] \in r_i,\ t[X_j] = null^{k_j}$, 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 every 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 every 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 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 $\pi\downarrow_Y(r_i) \subseteq \pi\downarrow_Z(r_j)$. If 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 usually called *referential integrity constraints* [4].

A *relational schema* RS is a pair (R, Δ), where R is a set of relation-schemes and Δ is a set of dependencies and constraints over R. A *database state* r of (associated with) RS consists of the relations associated with the relation-schemes of R; state r is said to be *consistent* iff it satisfies the dependencies and constraints of Δ.

It is well known in database design that the same data can be structured in different ways, that is, represented by different schemas, provided that these schemas have *equivalent information-capacities* [8]. 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 [8].

Definition 2.1. 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'. ■

Informally, a schema RS' has equivalent information-capacity with a 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.

[†] 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.

3. Background for Merging Relations

In this section we examine the main aspects of the merging technique developed in this paper, and introduce the null constraints involved in this technique.

In a relational database, real-world objects are represented by tuples and are identified by primary-key values. We assume that every relation in a database represents a homogeneous set of objects, and that relations associated with compatible primary-keys represent semantically compatible sets of real-world objects. Accordingly, in order to avoid creating relations that may represent heterogeneous sets of semantically incompatible objects, we consider for merging only relation-schemes that are associated with pairwise compatible primary-keys.

Let $\bar{R}$ denote a set of relation-schemes targeted for merging, and let $\bar{r}$ denote the set of relations associated with the relation-schemes of $\bar{R}$. Merging must preserve the tuples contained in the relations of $\bar{r}$, and therefore it involves outer-equi-joining on (compatible) primary-keys the relations of $\bar{r}$. However, instead of being joined directly, the relations of $\bar{r}$ are outer-equi-joined with a *key–relation* that contains all the primary-key values appearing in the relations of $\bar{r}$. The result of an outer-equi-join may contain redundant attribute values that must be subsequently removed. Consequently, the merging technique developed in the next section involves:

1. outer-equi-joining a key-relation with the relations involved in merging; and
2. projecting out the redundant attribute values from the result of the outer-equi-join above.

The *key–relation* is defined below.

Definition 3.1. Let $RS = (R, \Delta)$ be a relational schema, and let $\bar{R}$ be a subset of R consisting of relation-schemes that have pairwise compatible primary-keys. A relation-scheme $R_k(X_k)$ is said to be a *key–relation* of $\bar{R}$ iff (i) R_k has a primary-key, K_k, that is pairwise compatible with each of the primary-keys of the relation-schemes in $\bar{R}$, and (ii) for every database state associated with RS, the relation associated with R_k, r_k, satisfies the following condition: $\pi_{K_k}(r_k) = \bigcup_{R_i \in \bar{R}} (\textit{rename} (\pi_{K_i}(r_i), K_i \leftarrow K_k))$. ■

We consider in this paper relational schemas consisting of relation-schemes, key dependencies, key-based inclusion dependencies, and null constraints. We show below that for such schemas the key-relation of $\bar{R}$ can be one of the relation-schemes of $\bar{R}$.

Proposition 3.1. Let $RS = (R, F \cup I \cup N)$ be a relational schema, where R, F, I, and N denote sets of relation-schemes, key dependencies, key-based inclusion dependencies, and null constraints, respectively. Let $\bar{R}$ be a subset of R consisting of relation-schemes that have pairwise compatible primary-keys. Let R_o belong to $\bar{R}$, and let sets $\textit{Refkey}(R_o, \bar{R})$ and $\textit{Refkey}^*(R_o, \bar{R})$ be defined recursively as follows:

- $\textit{Refkey}(R_o, \bar{R}) = \{R_i \mid R_i \in \bar{R}, R_i[K_i] \subseteq R_o[K_o] \in I\}$;
- $\textit{Refkey}^*(R_o, \bar{R}) = \textit{Refkey}(R_o, \bar{R}) \cup_{R_i \in \textit{Refkey}(R_o, \bar{R})} \textit{Refkey}^*(R_i, \bar{R})$.

Then $\pi_{K_o}(r_o) = \bigcup_{R_i \in \bar{R}} (\textit{rename} (\pi_{K_i}(r_i), K_i \leftarrow K_o))$ for every database state associated with RS, iff $\bar{R} = \{R_o\} \cup \textit{Refkey}^*(R_o, \bar{R})$.

Proof Sketch. Straightforward, following the definition of inclusion dependencies. ■

Note that if a set of relation-schemes $\bar{R}$ defined as above does not contain a key-relation, then a new relation-scheme, $R_k(K_k)$, can be specified as a key-relation of $\bar{R}$, that is, so that K_k is pairwise compatible with each of the primary-keys of the relation-schemes of $\bar{R}$, and so that for every database state associated with relational schema RS, R_k is associated with relation relation r_k, where $r_k := \bigcup_{R_i \in \bar{R}} (\textit{rename} (\pi_{K_i}(r_i), K_i \leftarrow K_k))$.

For example, let $\bar{R}$ consist of relation-schemes OFFER and TEACH of figure 2, and let r_O and r_T be relations associated with OFFER and TEACH, respectively. A relation-scheme R_C can be a key-relation of $\bar{R}$ if the primary-key of R_C, say CN, is pairwise compatible with O.CN and T.CN, and if the relation associated with R_C, r_C, satisfies the following condition: $\pi_{CN}(r_C) = \textit{rename} (\pi_{O.CN}(r_O), O.CN \leftarrow CN) \cup \textit{rename} (\pi_{T.CN}(r_O), T.CN \leftarrow CN)$. Then merging relation-schemes OFFER and TEACH into a new relation-scheme ASSIGN (see figure 2) involves a state mapping defining the relation associated with ASSIGN: $r_A := r_C \overset{}{\bowtie}_{CN = O.CN} r_O \overset{}{\bowtie}_{CN = T.CN} r_T$. Note that if relation-schemes OFFER and TEACH are not involved in any inclusion dependency, then attributes O.CN and T.CN are redundant in ASSIGN; however, if OFFER and TEACH are involved in the right-hand sides of two distinct inclusion dependencies, then these attributes are not redundant in ASSIGN. If relation-schemes OFFER and TEACH are involved in inclusion dependency TEACH[T.CN] $\subseteq$ OFFER[O.CN] then, following proposition 3.1, OFFER is a key-relation of $\bar{R}$, and merging OFFER with TEACH involves outer-equi-joining relations r_O and r_T.

$\bar{R}$ = {OFFER (<u>O.CN</u>, O.DN) , TEACH (<u>T.CN</u>, T.FN)}

– ***Merge*** → ASSIGN (<u>CN</u>, O.CN, O.DN, T.CN, T.FN)

<u>Abbreviations</u>: CN=COURSE NUMBER, DN=DEPT NUMBER, FN=FACULTY NAME, O=OFFER, T=TEACH

Fig. 2. Relation-Scheme Examples (*keys are <u>underlined</u>*).

The result of an outer-equi-join is a relation, r_m, that usually contains null values; these null values must be restricted in order to ensure that the joined relations can be reconstructed from r_m without losing or adding information. Such restrictions are called *null constraints* and are defined below. A *null constraint* is a single-tuple restriction on where and how nulls should appear in a relation [9]. In the following definitions $R_i(X_i)$ denotes a relation-scheme, and r_i denotes a relation associated with $R_i(X_i)$.

A *null-existence constraint* is a statement of the form $R_i : Y \overset{Ex}{\rightarrow} Z$, where Y and Z are subsets of X_i; $R_i : Y \overset{Ex}{\rightarrow} Z$ is *satisfied* by 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 $R_i : \emptyset \overset{Ex}{\rightarrow} Z$, that is satisfied iff every subtuple $t[Z]$ of r_i is total. In relations associated with relation-scheme ASSIGN of figure 2, for example, ASSIGN: T.CN $\overset{Ex}{\rightarrow}$ O.CN does not allow tuples that contain null O.CN values together with non-null T.CN values, while ASSIGN: $\emptyset \overset{Ex}{\rightarrow}$ O.CN does not allow tuples containing null O.CN values.

A *null-synchronization* set is a set of null-existence constraints, of the form $\{R_i : A_j \overset{Ex}{\rightarrow} Y \mid A_j \in Y\}$, denoted $R_i : NS(Y)$; $R_i : NS(Y)$ is *satisfied* by r_i iff for every tuple t of r_i, $t[Y]$ is either total or consists entirely of null values (i.e. cannot be *partly null*). Consider relation-scheme ASSIGN of figure 2; the two null-existence constraints of ASSIGN:NS (T.CN, T.FN) do not allow in relations associated with ASSIGN tuples t containing partly null t[T.CN, T.FN] subtuples.

A *part-null constraint* is a statement of the form $R_i : PN(Y_1, \ldots, Y_m)$, where Y_j, $1 \le j \le m$, is a subset of X_i; $R_i : PN(Y_1, \ldots, Y_m)$ is *satisfied* by r_i iff for every tuple t of r_i, at least one subtuple $t[Y_j]$, $1 \le j \le m$, is total. In relations associated with relation-scheme ASSIGN of figure 2, for example, ASSIGN: PN ({O.CN, O.FN}, {T.CN, T.FN}) ensures that in every tuple t of such relations either subtuple t[T.CN, T.FN], or subtuple t[O.CN, O.FN], or both are total.

Finally, a *total-equality constraint* is a statement of the form $R_i : Y =\downarrow Z$, where Y and Z are compatible subsets of X_i; $R_i : Y =\downarrow Z$ is *satisfied* by r_i iff for every tuple t of r_i, $t[Y] = t[Z]$ whenever both $t[Y]$ and $t[Z]$ are total. Consider again relation-scheme ASSIGN of figure 2; total-equality constraint ASSIGN : T.CN $=\downarrow$ O.CN ensures that in every tuple of relations associated with ASSIGN, non-null values for attributes T.CN and O.CN are equal.

Inference axioms for null-existence constraints have the form of the inference axioms for functional dependencies (see [9]). Inference axioms for total-equality constraints are analogous to the inference axioms for the equality constraints of [7]. Null-existence, total-equality, and part-null constraints do not interact with each other.

4. A Relation Merging Technique

In this section we develop a relation merging technique that preserves the information-capacity and normal form of the relational schemas on which it is applied.

4.1 Merging Relation–Schemes

We define below a procedure for merging relation-schemes in relational schemas of the form $(R, F \cup I \cup N)$, where R, F, I, and N denote sets of relation-schemes, key dependencies, key-based inclusion dependencies, and null constraints, respectively; such a schema is shown in figure 3. We assume that the attributes are assigned globally unique names in the schema. Let $\bar{R}$ be a subset of R consisting of relation-schemes associated with pairwise compatible primary-keys; merging the relation-schemes of $\bar{R}$ implies mapping $(R, F \cup I \cup N)$ into a new schema, $(R', F' \cup I' \cup N')$, where R' results by replacing the relation-schemes of $\bar{R}$ with a new relation-scheme, R_m, and F', I', and N' consist of adjusted key dependencies, inclusion dependencies, and null constraints, respectively. For the sake of simplicity, we assume that initially the attributes associated with relation-schemes of $\bar{R}$ are not allowed to have null values.

Definition 4.1. Let $RS = (R, F \cup I \cup N)$ be a relational schema, where R, F, I, and N denote sets of relation-schemes, key dependencies, key-based inclusion dependencies, and null constraints, respectively. Let $\bar{R}$ be a subset of R consisting of relation-schemes associated with pairwise compatible primary-keys, so that every relation-scheme $R_i(X_i)$ of $\bar{R}$ is associated with nulls-not-allowed constraint $R_i : \emptyset \overset{Ex}{\rightarrow} X_i$. Let $R_k(X_k)$ be a relation-scheme

Relation-Schemes (*underlined keys*)

(1) PERSON (P.SSN)
(2) FACULTY (F.SSN)
(3) STUDENT (S.SSN)
(4) COURSE (C.NR)
(5) DEPARTMENT (D.NAME)
(6) OFFER (O.C.NR, O.D.NAME)
(7) TEACH (T.C.NR, T.F.SSN)
(8) ASSIST (A.C.NR, A.S.SSN)

Inclusion Dependencies

(1) FACULTY [F.SSN] $\subseteq$ PERSON [P.SSN]
(2) STUDENT [S.SSN] $\subseteq$ PERSON [P.SSN]
(3) OFFER [O.C.NR] $\subseteq$ COURSE [C.NR]
(4) OFFER [O.D.NAME] $\subseteq$ DEPARTMENT [D.NAME]
(5) TEACH [T.C.NR] $\subseteq$ OFFER [O.C.NR]
(6) TEACH [T.F.SSN] $\subseteq$ FACULTY [T.SSN]
(7) ASSIST [A.C.NR] $\subseteq$ OFFER [O.C.NR]
(8) ASSIST [A.S.SSN] $\subseteq$ STUDENT [S.SSN]

Null (*nulls–not–allowed*) **Constraints**

(1) PERSON: $\emptyset \overset{Ex}{\rightarrow}$ P.SSN
(2) FACULTY: $\emptyset \overset{Ex}{\rightarrow}$ F.SSN
(3) STUDENT: $\emptyset \overset{Ex}{\rightarrow}$ S.SSN
(4) COURSE: $\emptyset \overset{Ex}{\rightarrow}$ C.NR
(5) DEPARTMENT: $\emptyset \overset{Ex}{\rightarrow}$ D.NAME
(6) OFFER: $\emptyset \overset{Ex}{\rightarrow}$ O.C.NR, O.D.NAME
(7) TEACH: $\emptyset \overset{Ex}{\rightarrow}$ T.C.NR, T.F.SSN
(8) ASSIST: $\emptyset \overset{Ex}{\rightarrow}$ A.C.NR, A.S.SSN

Abbreviations: A=ASSIST, C=COURSE, D=DEPARTMENT, F=FACULTY, O=OFFER, S=STUDENT, T=TEACH

Fig. 3. A Relational Schema.

defined as follows: if $\bar{R}$ contains a key-relation, R_o, then $R_k := R_o$, $X_k := X_o$, and $K_k := K_o$; otherwise $X_k = K_k$, where K_k is disjoint with the attribute sets associated with the relation-schemes of $\bar{R}$, and for every database state associated with RS, R_k is associated with relation r_k, where $r_k := \bigcup_{R_i \in \bar{R}} (rename(\pi_{K_i}(r_i), K_i \leftarrow K_k))$.

Merge $(\bar{R})$ applied on RS generates relational schema $RS' = (R', F' \cup I' \cup N')$ as follows:

1. R' results by replacing in R the relation-schemes of $\bar{R}$ with a new relation-scheme, $R_m(X_m)$, such that $K_m := K_k$ and $X_m := X_k \cup_{R_i(X_i) \in \bar{R}} X_i$;
2. F' results by replacing the key dependencies involving primary-keys associated with the relation-schemes of $\bar{R}$ with key dependency $R_m : K_m \rightarrow X_m$;
3. N' is generated as follows:
   a. the nulls-not-allowed constraints associated with the relation-schemes of $\bar{R}$ are replaced with nulls-not-allowed constraint $R_m : \varnothing \overset{Ex}{\rightarrow} X_k$;
   b. for every relation-scheme $R_i(X_i)$ of $\bar{R}$, if $K_i \neq K_m$, where K_i is the primary-key of R_i, then total-equality constraint $R_m : K_m =\downarrow K_i$ is added to N';
   c. for every relation-scheme $R_i(X_i)$ of $(\bar{R} - \{R_k\})$, if X_i consists of more than one attribute, then the null-existence constraints of null-synchronization set $R_m : NS(X_i)$ are added to N';
   d. if R_k does not belong to $\bar{R}$, then part-null constraint $R_m : PN(X_1, \ldots, X_n)$ involving the attribute sets associated with relation-schemes $R_i(X_i)$ of $\bar{R}$, $1 \leq i \leq n$, is added to N';
   e. for every inclusion dependency of I of the form $R_j[Z] \subseteq R_i[K_i]$, where R_i and R_j belong to $\bar{R}$, if $K_i \neq K_m$ then null-existence constraint $R_m : X_j \overset{Ex}{\rightarrow} X_i$ is added to N';
4. I' results by (a) replacing R_i with R_m in every inclusion dependency of I involving a relation-scheme R_i of $\bar{R}$; (b) replacing K_i with K_m in every inclusion dependency of I', of the form $R_m[Z] \subseteq R_m[K_i]$; and (c) removing from I' inclusion dependencies of the form $R_m[K_i] \subseteq R_m[K_m]$, where K_i is the primary-key of a relation-scheme of $\bar{R}$.

Merge $(\bar{R})$ 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 database state r' of RS' into a database state $\bar{r}$ of RS, as follows:

η is *identity* for relations of r associated with relation-schemes of $(R - \bar{R})$; and maps set of relations $\bar{r} = \{r_i \mid r_i \in r, r_i$ is associated with R_i of $\bar{R}\}$ into r_m as follows: (i) $r_m := r_k$; (ii) for each R_i of $(\bar{R} - \{R_k\})$ do $r_m := r_m \overset{\otimes}{}_{K_m = K_i} r_i$;

η' is *identity* for relations of $(r' - r_m)$; and maps relation r_m into relations $\bar{r}_i := \pi\downarrow_{X_i}(r_m)$, where X_i is the attribute set of a relation-scheme $R_i(X_i)$ of $\bar{R}$. ■

Merging the relation-schemes of $\bar{R}$ into a new relation-scheme, R_m, involves defining the relation associated with R_m, r_m, as the outer-equi-join of a key-relation with the relations of $\bar{r}$, where $\bar{r}$ consists of relations associated with relation-schemes of $\bar{R}$. The set of dependencies and constraints generated by merging ensure that the relations of $\bar{r}$ can always be reconstructed (by total projection) from r_m. Thus, the total-equality constraints require the tuples of r_m to satisfy the join conditions involved in the definition of r_m; attributes that are not associated with the key-relation are allowed to have null values in r_m; the null-existence constraints of a given null-synchronization set ensure that in every tuple of r_m, a subtuple (of the form $t[X_i]$) corresponding to a tuple of a relation of $\bar{r}$ cannot have both null and non-null values; the part-null constraint ensures that in every tuple of r_m there is at least one total subtuple corresponding to a tuple of a relation of $\bar{r}$; the (inner-relational) null-existence constraints generated in step 3(e) express the inter-relational existence constraints implied by the inclusion dependencies involving pairs of relation-schemes of $\bar{R}$. Finally, the key dependencies removed in step 2 and the inclusion dependencies removed in step 4(c), are implied by the new key dependency and total-equality constraints, respectively by the total-equality and null-existence constraints (generated in step 3(e)), and therefore are redundant.

Two examples of applying ***Merge*** on the relational schema of figure 3 are shown in figures 4 and 5; in both examples the key-relation is relation-scheme COURSE. While in the example shown in figure 4 merging involves only relation-schemes COURSE, OFFER, and TEACH, in the example shown in figure 5 merging involves an additional relation-scheme, ASSIST. The following proposition shows that ***Merge*** preserves the information-capacity (in the

Relation-Schemes 4, 6, and 7 are replaced by
COURSE' (C.NR, O.C.NR, O.D.NAME, T.C.NR, T.F.SSN)

Inclusion Dependencies 3 to 7 are replaced by:
(9) : COURSE' [O.D.NAME] ⊆ DEPARTMENT [D.NAME]
(10) : COURSE' [T.F.SSN] ⊆ FACULTY [F.SSN]
(11) : ASSIST [A.C.NR] ⊆ COURSE' [O.C.NR]

Null Constraints 4, 6, and 7 are replaced by the following null constraints for COURSE':
(9) $\varnothing \overset{Ex}{\rightarrow}$ C.NR;
(10) NS (O.C.NR, O.D.NAME); (11) NS (T.C.NR, T.F.SSN);
(12) T.C.NR,T.F.SSN $\overset{Ex}{\rightarrow}$ O.C.NR,O.D.NAME;
(13) C.NR =↓ O.C.NR ; (14) C.NR =↓ T.C.NR

Fig. 4. Applying on Relational Schema of Fig. 3 ***Merge*** (COURSE, OFFER, TEACH).

sense of definition 2.1) and the normal form of the schemas on which it is applied.

Proposition 4.1. Let RS, RS', $\bar{R}$, and ***Merge*** be defined as in definition 4.1, so that $RS' = (R', F' \cup I' \cup N')$ is the result of applying ***Merge*** $(\bar{R})$ on $RS = (R, F \cup I \cup N)$. 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 2.1, and regards only the relations affected by merging; for the first two conditions, the proof follows the definition of η and η'; for the third condition, the proof follows from the fact that ***Merge*** preserves the primary-keys associated with the relation-schemes of $\bar{R}$, and that the outer-equi-joins involved in η are on primary keys; the last condition is obviously satisfied. (ii) The proof that all functional dependencies implied by $(F' \cup I' \cup N')$ are key dependencies is based on the fact that the closure of F can be computed independently of I (see [3]), and on the inference axioms for total-equality constraints and functional dependencies (see [7]). ■

The attributes involved in the total-equality constraints generated by ***Merge*** seem to be removable without any effect on the information-capacity of the relational schema. Since ***Merge*** preserves the normal form (BCNF) of relational schemas, these potentially redundant attributes are not a source of *update anomalies* [9]. Removing redundant attributes, however, simplifies the set of null constraints associated with merged relation-schemes, as well as reduces the size of the relations associated with merged relation-schemes. A procedure for removing redundant attributes is specified below.

4.2 Removing Redundant Attributes

We define below the conditions characterizing redundant attributes in relation-schemes generated by ***Merge***, and then specify a procedure for removing such attributes.

Relation-Schemes 4, 6, 7, and 8 are replaced by :
COURSE″(C.NR, O.C.NR, O.D.NAME, T.C.NR, T.F.SSN, A.C.NR, A.S.SSN)

Inclusion Dependencies 3 to 8 are replaced by :
(9) : COURSE″ [O.D.NAME] ⊆ DEPARTMENT [D.NAME]
(10) : COURSE″ [T.F.SSN] ⊆ FACULTY [F.SSN]
(11) : COURSE″ [A.S.SSN] ⊆ STUDENT [S.SSN]

Null Constraints 4, 6, 7, and 8 are replaced by the following null constraints for COURSE″:
(9) $\emptyset \xrightarrow{Ex}$ C.NR; (10) NS (O.C.NR, O.D.NAME);
(11) NS (T.C.NR, T.F.SSN); (12) NS (A.C.NR, A.S.SSN);
(13) T.C.NR,T.F.SSN $\xrightarrow{Ex}$ O.C.NR,O.D.NAME;
(14) A.C.NR,A.S.SSN $\xrightarrow{Ex}$ O.C.NR,O.D.NAME;
(15) C.NR $=\!\downarrow$ O.C.NR; (16) C.NR $=\!\downarrow$ T.C.NR; (17) C.NR $=\!\downarrow$ A.C.NR

Fig. 5. Applying on Relational Schema of Fig. 3 ***Merge*** (COURSE, OFFER, TEACH, ASSIST).

Definition 4.2. Let RS, RS', $\bar{R}$, and ***Merge*** be defined as in definition 4.1, so that $RS' = (R', F' \cup I' \cup N')$ is the result of applying ***Merge*** $(\bar{R})$ on RS, and $R_m(X_m)$ is the result of merging the relation-schemes of $\bar{R}$. Let X_i be a subset of X_m, such that X_i is associated with relation-scheme R_i of $\bar{R}$, and let Y_i be a subset of X_i involved in a total-equality constraint associated with R_m, such that $Y_i \neq K_m$. Then Y_i is said to be *removable* in R_m if the following conditions are satisfied:

(1) $\mid X_i - Y_i \mid \geq 1$.
(2) Y_i is not involved in the right-hand side of I' inclusion dependencies of the form $R_j[Z] \subseteq R_m[Y_i]$, $R_j \neq R_m$.
(3) If Y_i is involved in the left-hand side of an I' inclusion dependency of the form $R_m[Y_i] \subseteq R_j[K_j]$, $R_j \neq R_m$, then for every subset of X_m, W, involved in a total-equality constraint associated with R_m, I' includes an inclusion dependency of the form $R_m[W] \subseteq R_j[K_j]$.
(4) Y_i does not overlap with other foreign-keys of R_m, that is, if attributes of Y_i are involved in the left-hand side of an I' inclusion dependency of the form $R_m[W] \subseteq R_j[K_j]$, $R_j \neq R_m$, then $W \equiv Y_i$. ■

For example, O.C.NR, T.C.NR, and A.C.NR are removable attributes in relation-scheme COURSE″ of figure 5.

Note that an attribute that is removable in a merged relation-scheme associated with attribute set X_m, is not necessarily removable in a merged relation-scheme associated with a proper subset of X_m. Let $R_m(X_m)$ and $R'_m(X'_m)$ result by applying ***Merge*** $(\bar{R})$ and ***Merge*** $(\bar{R}')$, respectively, on a relational schema RS, where $\bar{R}'$ is a subset of $\bar{R}$. Then X'_m is a subset of X_m, but because of condition (2) of definition 4.2, a common subset of X_m and X'_m can be removable in R_m, but not in R'_m. For example, while attribute O.C.NR is removable in relation-scheme COURSE″ of figure 5, O.C.NR is not removable in relation-scheme COURSE′ of figure 4.

Definition 4.3. Let RS, RS', $\bar{R}$, and ***Merge*** be defined as in definition 4.1, so that $RS' = (R', F' \cup I' \cup N')$ is the result of applying ***Merge*** $(\bar{R})$ on RS, and $R_m(X_m)$ is the result of merging the relation-schemes of $\bar{R}$. Let Y_i be a subset of X_m, removable in R_m. ***Remove*** (Y_i) applied on RS' generates $RS'' = (R'', F'' \cup I'' \cup N'')$ as follows:

1. R'' results by removing the attributes of Y_i from attribute set X_m associated with R_m;
2. F'' results by replacing in key dependencies of F' every attribute A of Y_i with an attribute of K_m that corresponds to A in a total-equality constraint of N';
3. I'' results by replacing Y_i with K_m in inclusion dependencies of I' of the form $R_m[Y_i] \subseteq R_j[K_j]$;
4. N'' results by removing (a) the attributes of Y_i from the part-null and null-existence constraints of N', and

(b) total-equality constraint $R_m : K_m = \downarrow Y_i$ from N'.

Remove (Y_i) is associated with two *state mappings*, μ and μ', where μ maps a database state r' of RS' into a database state r'' of RS'', and μ' maps a database state r'' of RS'' into a database state r' of RS', as follows:

- μ is *identity* for relations of r' associated with relation-schemes of $(R' - \{R_m\})$; and maps relation r'_m associated with R_m into $r''_m := \pi_{(X_m - Y_i)}(r'_m)$;
- μ' is *identity* for relations of r'' associated with relation-schemes of $(R'' - \{R_m\})$; and maps relation r''_m associated with R_m, into $r'_m := r''_m \underset{K_m = Y_i}{\bowtie} (rename(\pi_{K_m}(\pi\downarrow_{K_m (X_i - Y_i)}(r''_m)), K_m \leftarrow Y_i))$. ■

An example of applying ***Remove*** is shown in figure 6, where attributes O.C.NR, T.C.NR, and A.C.NR are successively removed from relation-scheme COURSE'' of figure 5. The following proposition shows that ***Remove*** preserves the information-capacity (in the sense of definition 2.1) of the relational schemas on which it is applied.

Proposition 4.2. Let RS, RS', $\bar{R}$, and ***Merge*** be defined as in definition 4.1, so that $R_m(X_m)$ in RS' is the result of merging the relation-schemes of $\bar{R}$. Let RS'', Y_i, and ***Remove*** be defined as in definition 4.3, so that Y_i is a removable subset of X_m, and RS'' is the result of applying ***Remove*** (Y_i) on RS'. Then RS' and RS'' have equivalent information-capacities.

Proof Sketch. The proof regards only the relations (associated with R_m) affected by removal, and refers to the conditions of definition 2.1. The last condition is obviously satisfied. For the first two conditions, the proof follows the definition of the state mappings; note that replacing inclusion dependencies of the form $R_m[Y_i] \subseteq R_j[K_j]$ with $R_m[K_m] \subseteq R_j[K_j]$ can be accomplished because of conditions (3) and (4) of definition 4.2. For the third condition, the proof follows from the fact that the primary-key of R_m is not affected by ***Remove***; note that condition (1) of definition 4.2 is essential for satisfying this condition, and that if condition (2) of definition 4.2 is removed and the replacement of Y_i with K_m in inclusion dependencies of the form $R_j[Z] \subseteq R_m[Y_i]$ is allowed, then the third condition of definition 2.1 would not be satisfied. ■

Remove is applied on COURSE'' of figure 5 for:
O.C.NR, T.C.NR, and A.C.NR

Relation-Scheme COURSE'' is replaced by
COURSE'' (C.NR, O.D.NAME, T.F.SSN, A.S.SSN)

Inclusion Dependencies involving COURSE'' are unchanged

Null Constraints involving COURSE'' are replaced by :
$\emptyset \xrightarrow{E} C.NR$, $T.F.SSN \xrightarrow{E} O.D.NAME$, $A.S.SSN \xrightarrow{E} O.D.NAME$

Fig. 6. Applying ***Remove*** on Relational Schema of Fig. 5.

5. Applications of the Merging Technique

In this section we discuss several aspects of applying our relation merging technique to relational databases that are implemented using a commercial relational database management system (DBMS), or to relational schemas developed using an EER-oriented design methodology.

5.1 Relation Merging for Relational DBMSs

The relation merging technique developed in section 4 involves merging relation-schemes into a new (merged) relation-scheme, and removing redundant attributes from the merged relation-scheme; a merged relation-scheme is associated with various null constraints, may be involved in non key-based inclusion dependencies (that are not referential integrity constraints), and may be associated with candidate keys that are allowed to be null. Some relational DBMSs do not have mechanisms for maintaining general null constraints, candidate keys that are allowed to be null, and non key-based inclusion dependencies; for such DBMSs our merging technique can be applied only when such constraints and dependencies are not generated.

Non key-based inclusion dependencies are not supported by DBMSs such as IBM's DB2 [5], but can be maintained in DBMSs such as SYBASE 4.0 [13] (using the *triggers* mechanism) and INGRES 6.3 [6] (using the *rules* mechanism). However, even in SYBASE and INGRES non key-based inclusion dependencies are harder to maintain then key-based inclusion dependencies. Keys that are allowed to be null cannot be maintained in DBMSs (e.g. SYBASE, INGRES) that consider all null values as identical. The conditions ensuring that ***Merge*** generates only key-based inclusion dependencies and keys consisting only of attributes that are not allowed to have null values, are given below (the type of inclusion and key dependencies is not affected by ***Remove***).

Proposition 5.1. Let RS, RS', $\bar{R}$, and ***Merge*** be defined as in definition 4.1, so that $RS' = (R', F' \cup I' \cup N')$ is the result of applying ***Merge*** $(\bar{R})$ on RS, and $R_m(X_m)$ is the result of merging the relation-schemes of $\bar{R}$. Then (i) I' contains only key-based inclusion dependencies iff every relation-scheme $R_i(X_i)$ of $\bar{R}$ that is not a key-relation, is not involved in the right-hand side of inclusion dependencies of the form $R_j[Z] \subseteq R_i[K_i]$, $R_j \notin \bar{R}$; and (ii) the key attributes associated with R_m are not allowed to have null values iff every relation-scheme $R_i(X_i)$ of $\bar{R}$ that is not a key-relation, is associated with a unique (primary) key.

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

Some DBMSs provide mechanisms for maintaining general null constraints. For example, the *validproc* mechanism of DB2, the *triggers* mechanism of SYBASE 4.0, and the *rules* mechanism of INGRES 6.3 can be used

to maintain null constraints. However, these mechanisms require tedious and error-prone specifications of procedures, and therefore are difficult to use. Conversely, all relational DBMSs support declarative (non procedural) specifications for nulls-not-allowed constraints. The following proposition gives the conditions ensuring that the set of null constraints generated by ***Merge*** and simplified by ***Remove***, consists only of nulls-not-allowed constraints.

Proposition 5.2. Let RS, RS', $\bar{R}$, and ***Merge*** be defined as in definition 4.1, so that RS' is the result of applying ***Merge*** $(\bar{R})$ on $RS = (R, F \cup I \cup N)$, and $\underline{R}_m(X_m)$ is the result of merging the relation-schemes of $\bar{R}$. Let ***Remove*** and RS'' be defined as in definition 4.3, so that the result of removing all removable attributes from $R_m(X_m)$ by applying ***Remove*** is $RS'' = (R'', F'' \cup I'' \cup N'')$. Then N'' contains only nulls-not-allowed constraints if $\bar{R}$ contains a relation-scheme $\underline{R}_k(X_k)$ such that for every relation-scheme $R_i(X_i)$ of $\bar{R}$, $R_i \neq R_k$, the following conditions are satisfied:

(1) $R_i[K_i] \subseteq R_k[K_k]$ belongs to I.

(2) $|Z| = 1$, where $Z = X_i - K_i$, (i.e. R_i has exactly one non primary-key attribute).

(3) R_i is not involved in the right-hand side of any inclusion dependency of I.

(4) In addition to $R_i[K_i] \subseteq R_k[K_k]$, R_i can be involved only in the left-hand sides of inclusion dependencies of I having the form $R_i[Z] \subseteq R_j[K_j]$ or $R_i[K_i] \subseteq R_j[K_j]$, where $R_j \notin \bar{R}$; however, if $R_i[K_i] \subseteq R_j[K_j]$ belongs to I then $R_k[K_k] \subseteq R_j[K_j]$ also belongs to I.

Proof Sketch. Note that condition (1) implies that relation-scheme R_k is a key-relation of $\bar{R}$. For part-null and total-equality constraints the proof follows from the definitions of ***Merge*** and ***Remove***. Regarding null-synchronization sets, null-existence constraints with empty right-hand sides are trivially satisfied. Finally, if all inclusion dependencies of I involving relation-schemes of $\bar{R}$ in both their left-hand and right-hand sides, are of the form specified in (1) above, then ***Merge*** generates only null-existence constraints that are either nulls-not-allowed constraints or belong to a null-synchronization set. ■

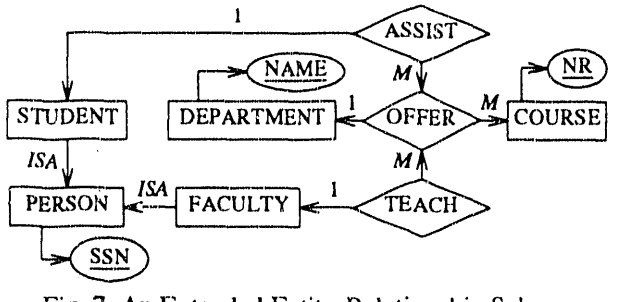


Fig. 7. An Extended Entity-Relationship Schema.

5.2 Relation Merging for Relational Schema Translations of EER Schemas

Relational schemas consisting of relation-schemes, key dependencies, key-based inclusion dependencies, and null constraints may represent EER schemas. The relational schema of figure 3, for example, represents the EER schema of figure 7. Translations of EER schemas into relational schemas are discussed in detail in [11]. Note that if in relational schema translations of EER schemas every relation-scheme represents a single EER object-set, then the set of null constraints consists only of nulls-not-allowed constraints involving primary-keys and foreign-keys [11] (e.g. see figure 3). These constraints express the usual restriction of not allowing EER entity-identifier attributes to have null values, and comply with the simplifying assumption in the definition of ***Merge***.

ER and EER-oriented design methodologies for relational schemas recommend using a single relation-scheme for representing a binary *many-to-one* relationship-set and the entity-set involved in that relationship-set with a *many* cardinality [14]. The result of applying the merging procedure developed in this paper on relational schema translations of EER schemas, shows that a single relation-scheme can be used for representing more complex structures as well (see figure 8). Such compact representations are especially useful for representing large generalization hierarchies whose specialization entity-sets are not involved in relationship-sets (see figure 8(i)). In most cases these compact representations require only additional null constraints. For example, in each of

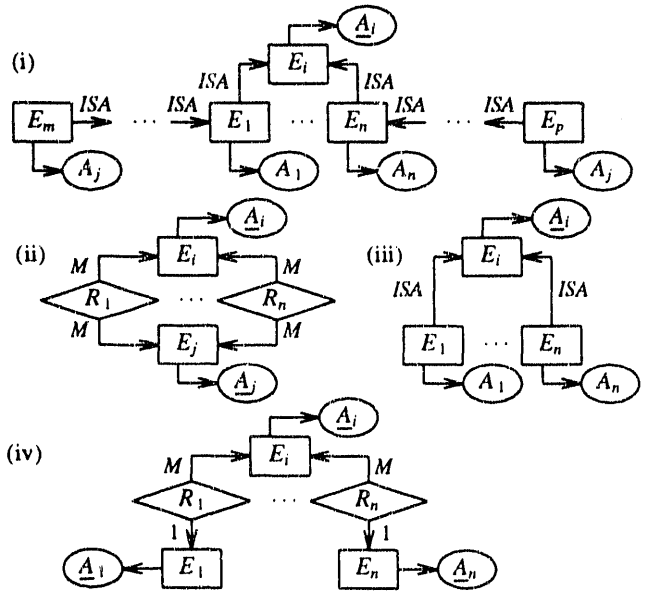


Fig. 8. EER Structures Amenable for Representations Involving a Single Relation.

the EER structures shown in figure 8 multiple object-sets can be represented using a single relation-scheme; however, while for the EER schemas of figures 8(i) and 8(ii) this representation involves general null constraints, for the EER schemas of figures 8(iii) and 8(iv) this representation involves only nulls-not-allowed constraints. The conditions of proposition 5.2 imply that multiple object-sets can be represented by a single relation involving only nulls-not-allowed constraints, only if these multiple object-sets consist of:

(1) 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 in relationship-sets or weak entity-sets, and (c) have exactly one (not inherited) attribute of their own (see figure 8(iii)); or

(2) 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 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 (e.g. see figure 8(iv)).

Consider, for example, the EER schema of figure 7. Entity-set COURSE together with relationship-sets OFFER, TEACH, and ASSIST do not satisfy the conditions above, and, indeed, these object-sets can be represented using a single relation-scheme (such as relation-scheme COURSE" of figure 6) only if this relation-scheme is associated with null-existence constraints. Conversely, relationship-sets OFFER, TEACH, and ASSIST, satisfy conditions (2.a), (2.b), and (2.c), and therefore can be represented using a single relation-scheme that is associated only with nulls-not-allowed constraints.

6. Summary

We have presented in this paper a merging technique for relational schemas consisting of relation-schemes, key dependencies, referential integrity constraints, and null constraints. We have examined the conditions required for using this technique with relational DBMSs that provide different mechanisms for maintaining null and referential integrity constraints. For relational schemas developed using an EER-oriented design methodology, we have shown that a relation-scheme can be used for representing multiple object-sets not only for the standard binary *many-to-one* relationship-set structure, but for more complex structures as well.

Variations of the merging technique presented in this paper have been implemented as part of a database *Schema Definition and Translation* tool (***SDT***) [12]. Given an EER schema, ***SDT*** generates the corresponding schema definition for various relational DBMSs, such as DB2, SYBASE 4.0, and INGRES 6.3. ***SDT*** provides the options of (i) establishing a one-to-one correspondence between the relation-schemes in the relational schema and the object-sets in the EER schema (i.e. not using merging), or (ii) using merging for reducing the number of relation-schemes in the relational schema.

References

[1] C. Beeri, P.A. Bernstein, and N. Goodman, "A sophisticate's introduction to database normalization theory", Proc. of the *4th VLDB Conference*, 1978, pp. 113-124.

[2] 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.

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

[4] C.J. Date, "Referential integrity", in *Relational Database-Selected Writings*, Addison-Wesley, 1986.

[5] IBM Corporation, "IBM DATABASE 2 Referential Integrity Usage Guide", June 1989.

[6] Ingres, Inc., "INGRES/SQL Reference Manual", Release 6.3, Alameda, California, Nov. 1989.

[7] A. Klug, "Calculating Constraints on Relational Expressions", *ACM Trans. on Database Systems* **5**,3 (Sep. 1980), pp. 260-290.

[8] R. Hull, "Relative information capacity of simple relational database schemata", in Proc. of the *3rd PODS Symposium*, 1984, pp. 97-109.

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

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

[11] V.M. Markowitz and A. Shoshani, "Representing extended entity-relationship structures in relational databases: A modular approach", to appear in *ACM Trans. on Database Systems*.

[12] V.M. Markowitz and W. Fang, "***SDT*** 4.1. Reference manual", TR LBL-27843, May 1991.

[13] Sybase, Inc., "Transact-SQL User's Guide", Release 4.0, Emeryville, California, Oct. 1989.

[14] 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.

END

DATE FILMED

8/31/92